Tokaj-Hegyalja Egyetem

Hálózati Architektúrák és Protokollok

7. Hálózati réteg

Mileff Péter - Miskolci Egyetem

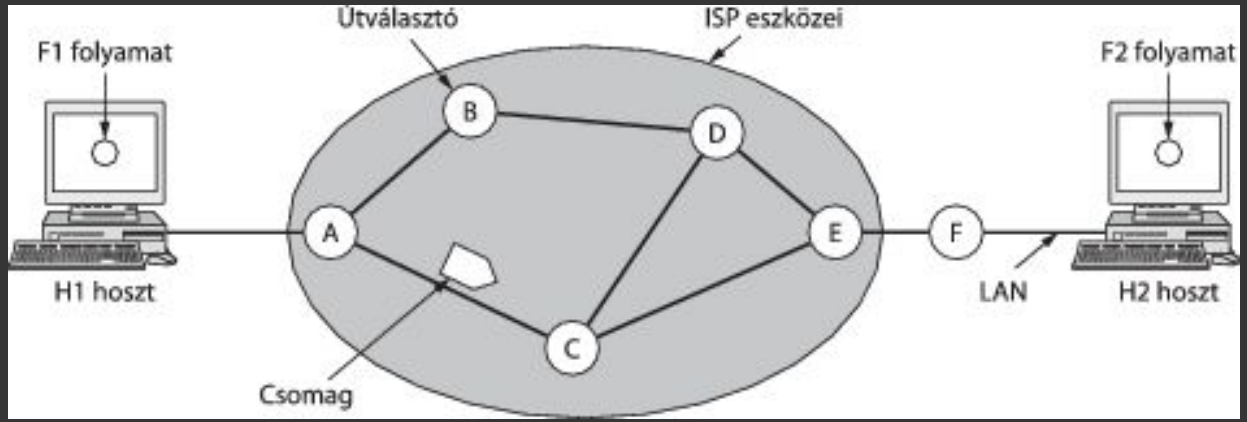
Hálózati réteg

- A hálózati réteg a rétegzett hálózati modell harmadik szintje
- **Elsődleges feladata az adatok továbbítása különböző hálózatok között**
  - a hálózati réteg lehetővé teszi, hogy az adatok több, egymástól független hálózaton keresztül jussanak el a forrástól a célállomásig
- A modern számítógépes hálózatok – különösen az internet – számos különálló hálózat összekapcsolásából épülnek fel
  - A hálózati réteg biztosítja ezen hálózatok közötti kommunikációt
  - valamint meghatározza az adatok útvonalát a hálózaton keresztül
- E célok elérése érdekében a hálózati rétegnek ismernie kell a hálózat (vagyis az útválasztók és az adatkapcsolatok halmazának) topológiáját
  - megfelelő útvonalakat kell találnia azon keresztül, még nagy kiterjedésű hálózatok esetén is.
- Arra is ügyelnie kell, hogy úgy válassza ki az útválasztókat, hogy elkerülje néhány kommunikációs vonal és útválasztó túlterhelését

A hálózati réteg tervezési kérdései...

Tárol-és-továbbít típusú csomagkapcsolás

- Példa hálózat



Tárol-és-továbbít típusú csomagkapcsolás

- **A berendezések működése a következő:**
  - A hosztok az elküldeni kívánt csomagokat a saját LAN-on vagy a szolgáltató felé vezető kétpontos (point-to-point) kapcsolaton keresztül a legközelebbi útválasztóhoz továbbítják
  - Az útválasztó tárolja a csomagot, amíg az teljes egészében be nem érkezik, hogy ki lehessen számítani az ellenőrző összeget.
  - Ezután a csomag mindig a soron következő útválasztóhoz kerül, míg el nem éri a címzett hosztot.
  - Ezt hívják **tárol-és-továbbít** (**store-and-forward**) típusú csomagkapcsolásnak

Tárol-és-továbbít típusú csomagkapcsolás

- A hálózati réteg interfészen nyújtja szolgáltatásait a szállítási rétegnek
- A hálózati réteg tervezésénél a következő vezérelveket tartották szem előtt:
  - A szolgáltatásoknak függetleneknek kell lenniük az útválasztók kialakításától
  - A szállítási réteg elől el kell takarni a jelenlevő útválasztók számát, típusát és topológiáját
  - A szállítási réteg rendelkezésére bocsátott hálózati címeknek egységes számozási rendszert kell alkotniuk, még LAN-ok és WAN-ok esetén is
- A hálózati réteg tervezői nagy szabadságot élveznek a szállítási rétegnek nyújtandó szolgáltatások részletes specifikációinak elkészítése során
- A szabadság két szembenálló csoportot eredményez:
  - A vita középpontjában az áll, hogy vajon a hálózati réteg **összeköttetés-alapú** vagy **összeköttetés nélküli szolgáltatás**t nyújtson-e.

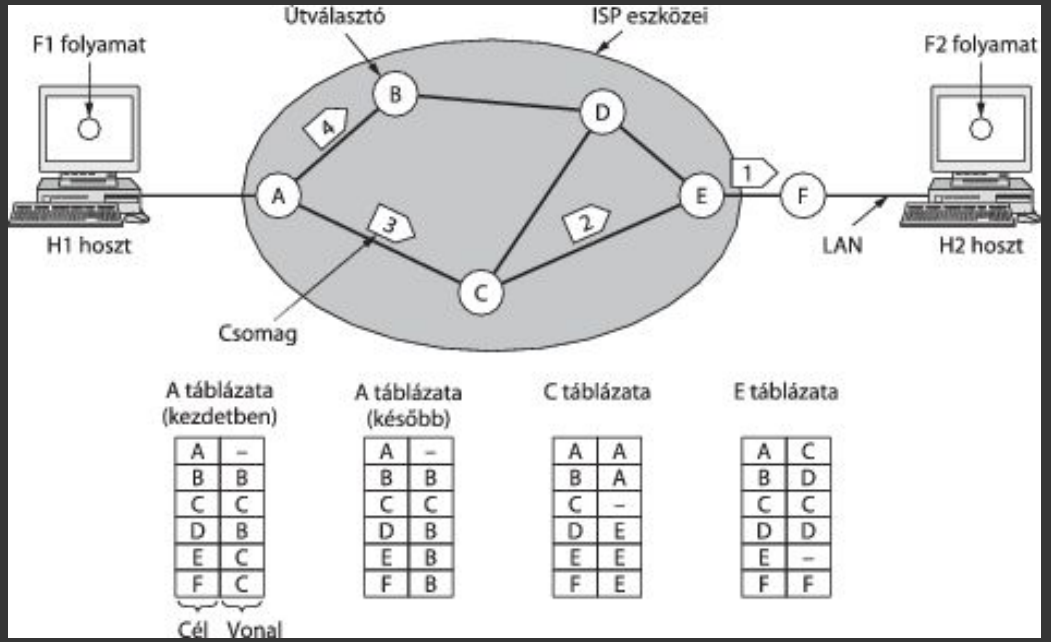
Összeköttetés nélküli szolgáltatás...

Összeköttetés nélküli szolgáltatás...

- Az összeköttetés nélküli szolgáltatás esetében a hálózatba érkező csomagok egyenként és egymástól függetlenül kerülnek továbbításra
  - előzetes összeköttetés-felépítésre nincs szükség
- Ebben az összefüggésben:
  - a csomagokat gyakran **datagram**oknak (datagrams, DG),
  - a hálózatot pedig **datagramalapú hálózat**nak (datagram network) is nevezik a távirat (telegram) kifejezés mintájára.
- Ha **összeköttetés-alapú szolgáltatás**t használunk, a forrás és a cél útválasztó között előre ki kell építeni egy útvonalat
  - mielőtt egyetlen adatcsomagot is elküldenénk
- Ezt a kapcsolatot **virtuális áramkör**nek (virtual circuit, VC) nevezzük
  - a telefonhálózat fizikai áramköreinek mintájára,
  - a hálózat neve pedig ebben az esetben **virtuálisáramkör-alapú hálózat** (virtual circuit subnet).

Összeköttetés nélküli szolgáltatás...

- Tegyük fel, hogy F1 folyamat egy hosszú üzenetet szeretne küldeni az F2 folyamat számára.
- F1 átadja az üzenetet a szállítási rétegének azzal az utasítással, hogy továbbítsa azt a H2 hoszt F2 folyamatának.
- A szállítási réteg kódja a H1 hoszton fut, tipikusan az operációs rendszeren belül.
- Ez az üzenet elejéhez fűz egy szállítási fejrészt és továbbítja azt a hálózati rétegnek,
- amelyik valószínűleg szintén egy eljárás az operációs rendszeren belül.



Összeköttetés nélküli szolgáltatás...

- Tegyük fel, hogy az üzenet négyszer olyan hosszú, mint a maximális csomagméret:
  - a hálózati rétegnek négy csomagra kell bontania,
  - és mindegyiket sorban az útválasztóhoz továbbítania valamilyen kétpontos protokoll,
    - például a PPP felhasználásával
- Ezen a ponton lép be a képbe a szolgáltató:
  - A hálózat minden útválasztójának van egy belső táblázata,
    - minden lehetséges cél esetére megadja, hogy merrefelé kell továbbítani a csomagokat.
  - A táblázat bejegyzései olyan kettősök:
    - amelyek a címzett útválasztó-azonosítóját és a címzetthez vezető kimeneti vonal azonosítóját tartalmazzák

Összeköttetés nélküli szolgáltatás...

- Csak közvetlen kapcsolatban álló összeköttetéseket lehet használni.
- A fenti ábrán az útválasztónak csak két kimeneti vonala van
  - egy a B és egy a C felé
  - így minden bejövő csomagot a két útválasztó közül valamelyiknek kell továbbküldeni,
  - még akkor is, ha a címzett egy ezektől különböző útválasztó.
- Az A kezdeti útválasztó táblázatát a képen a „kezdetben” megjelölésű oszlop mutatja.
- Négy csomag útját követhetjük végig az ábrán

Összeköttetés nélküli szolgáltatás...

- Az A útválasztó a beérkezett 1., 2. és 3. csomagot rövid ideig tárolja, miután azok beérkeztek a bejövő adatkapcsolaton és kiszámítja az ellenőrző összegüket
- Ezután a táblázatának megfelelően továbbítja a C-hez egy új keretben.
- Az 1. csomag ezután az E-hez, majd az F-hez kerül.
- F-hez érve beágyazódik egy keretbe, és a LAN-on keresztül eljut H2-höz.
- A 2. és a 3. csomag ugyanezt az utat járja be.

Összeköttetés nélküli szolgáltatás...

- A 4. csomaggal azonban valami más történik
- Az A-tól a B útválasztóhoz kerül, annak ellenére, hogy az F volt a címzett.
- Valami miatt az A úgy döntött, hogy a 4. csomagot más úton továbbítja, mint az első hármat.
- Értesülhetett például egy, az ACE út mentén lévő torlódásról, és módosíthatta az útválasztó táblázatát, amint azt az ábra „később” megjelölésű oszlopa mutatja
- **Azt az algoritmust, mely a táblázatok karbantartását végzi és meghozza az útválasztó döntéseket, útválasztó algoritmusnak (routing algorithm) nevezzük**
- Ezen algoritmusok képezik fejezetünk egyik legfontosabb tárgyát. Ahogy látni fogjuk, számos különböző algoritmus létezik.

Összeköttetés nélküli szolgáltatás...

- Az internetprotokoll (IP, Internet Protocol), amely a teljes internet alapját képezi, meghatározó példája az összeköttetés nélküli hálózati szolgáltatásnak.
- Minden csomag tartalmazza a címzett IP-címet, amelyet az útválasztók használnak az egyes csomagok egyesével történő továbbításához.
- Az IPv4-csomagok címe 32 bites, az IPv6-csomagok címe pedig 128 bites.

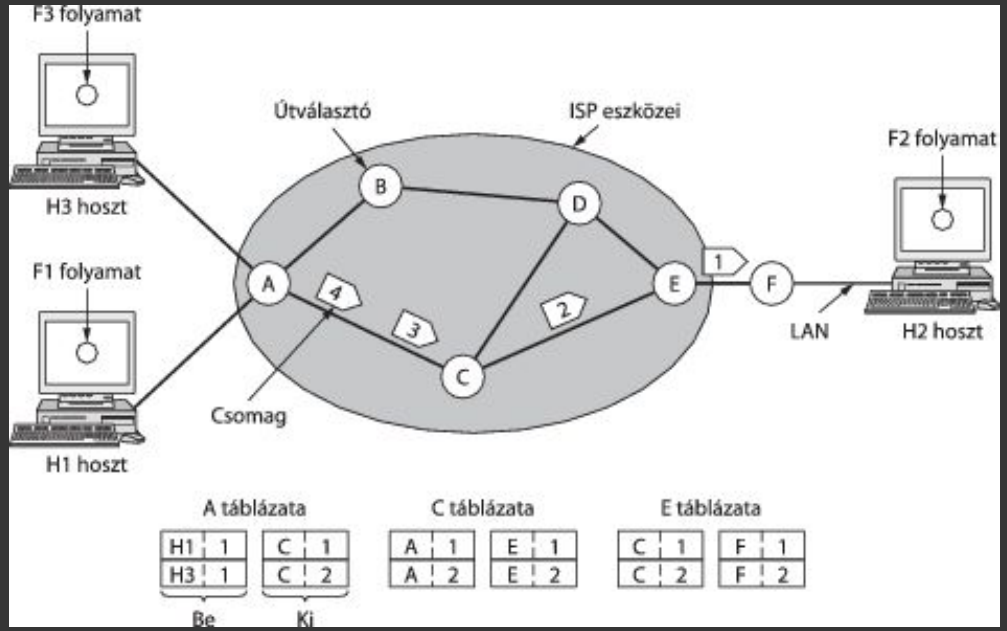
Összeköttetés alapú szolgáltatás...

Összeköttetés alapú szolgáltatás...

- Az összeköttetés-alapú szolgáltatáshoz szükségünk van egy **virtuálisáramkör-alapú hálózat**ra.
- **A virtuális áramkörök alapötlete:** elkerülik azt, hogy minden egyes csomag számára újra és újra útvonalat kelljen választani
- **Ehelyett:**
  - már az összeköttetés felépítésekor kiválasztanak a küldő és a címzett hoszt között egy utat
  - amelyet az útválasztók az összeköttetés kiépítése keretében tárolnak a táblázataikban.
  - Ezt az utat használják azután a kapcsolat teljes forgalmának lebonyolítására, pontosan úgy, ahogy a telefonhálózat esetében is.
  - Amikor az összeköttetés megszűnik, a virtuális áramkör is bomlik.
- Az összeköttetés-alapú szolgáltatás esetén minden csomag tartalmaz egy azonosítót
  - megmondja, hogy a csomag melyik virtuális áramkörhöz tartozik.

Összeköttetés alapú szolgáltatás...

- Itt a H1 hoszt kialakított egy összeköttetést H2-vel.
- Ezt jelzi az útválasztó táblázatok első bejegyzése.
- Az A útválasztó táblázatának első sora szerint, ha egy 1-es azonosítót hordozó csomag érkezik a H1 felől, akkor azt a C útválasztó felé kell továbbítani 1-es azonosítóval.
- Hasonlóképpen, C első bejegyzése az E-hez továbbítja a csomagot, szintén 1-es összeköttetés-azonosítóval.



Összeköttetés alapú szolgáltatás...

- Mi történik, ha H3 is szeretne összeköttetést létesíteni H2-vel
- Összeköttetés-azonosítónak az 1-et választja
  - mivel ez kezdeményezi az összeköttetést és ez az egyetlen összeköttetése
  - és arra utasítja a hálózatot, hogy hozza létre a virtuális áramkört.

- Ez eredményezi a második sort a táblázatokban.
- **Vegyük észre, hogy ez ütközéshez vezet:** mert bár A könnyen meg tudja különböztetni a H1-ből és a H3-ból érkező, egyaránt 1-es azonosítójú csomagokat, C már nem képes erre.
- Ezért A új összeköttetés-azonosítót rendel a második összeköttetés kimenő forgalmához.
- Az ilyen konfliktushelyzetek elkerüléséhez tehát az szükséges:
  - hogy az útválasztók képesek legyenek megváltoztatni az összeköttetés-azonosítókat a kimenő csomagokban.

Összeköttetés alapú szolgáltatás...

- Egyes esetekben ezt a működést **címkekapcsolás**nak (label switching) is nevezik
- Összeköttetés-alapú hálózati szolgáltatásra példa a többprotokollos címkekapcsolás (MultiProtocol Label Switching, MPLS)
    - Ezt használják az ISP-hálózatok az interneten

- **Működése:**
    - Ebben az esetben az IP-csomagba beágyaznak egy 20 bites összeköttetés-azonosítót vagy összeköttetés-címkét tartalmazó MPLS-fejrészt
    - Az MPLS gyakran rejtve van az ügyfelek elől, mint amikor az ISP hosszú távú összeköttetéseket alakít ki nagy forgalomhoz,
        - de egyre inkább használják akkor is, ha a szolgáltatás minősége fontos, illetve egyéb ISP forgalomkezelési feladatokhoz is alkalmazzák.

Összeköttetés alapú szolgáltatás...

- Egyes esetekben ezt a működést **címkekapcsolás**nak (label switching) is nevezik
- Összeköttetés-alapú hálózati szolgáltatásra példa a többprotokollos címkekapcsolás (MultiProtocol Label Switching, MPLS)
    - Ezt használják az ISP-hálózatok az interneten

- **Működése:**
    - Ebben az esetben az IP-csomagba beágyaznak egy 20 bites összeköttetés-azonosítót vagy összeköttetés-címkét tartalmazó MPLS-fejrészt
    - Az MPLS gyakran rejtve van az ügyfelek elől, mint amikor az ISP hosszú távú összeköttetéseket alakít ki nagy forgalomhoz,
        - de egyre inkább használják akkor is, ha a szolgáltatás minősége fontos, illetve egyéb ISP forgalomkezelési feladatokhoz is alkalmazzák.

A virtuálisáramkör- és a datagramalapú hálózatok...

Virtuálisáramkör- és a datagramalapú hálózatok

- A hálózaton belül sokfajta kompromisszum lehetséges a virtuális áramkörök és a datagramok között
- Az egyik kompromisszum az **összeköttetés-felépítési** és a **címfeldolgozási idő** közt kereshető.
- A virtuális áramkörök használata megkövetel egy összeköttetés-felépítési fázist, amely idő- és erőforrás-igényes.
- Ha azonban az összeköttetés felépült, akkor könnyű eldönteni, mi legyen a csomagokkal:
  - az útválasztó az áramkör számát indexként használja a táblázatokhoz a csomag továbbítása érdekében
- Datagramalapú hálózatban nincs szükség az összeköttetés felépítésére
  - azonban sokkal bonyolultabb az a keresőeljárás, amely a célcímhez tartozó bejegyzés meghatározásához szükséges.

Virtuálisáramkör- és a datagramalapú hálózatok

- Másik probléma, hogy a datagramalapú hálózatokban használt célcím hosszabb a virtuális áramkörökben használt áramkörszámnál
  - mivel a célcím globális jelentéssel rendelkezik.
- Ha a csomagok hossza rövid, akkor a minden csomagban jelenlevő teljes célcím jelentős mértékű többletterhet jelent,
  - ez sávszélesség-veszteséggel jár
- További kérdés, hogy mennyi helyet foglalnak el a táblázatok az útválasztók memóriájában
  - Egy datagramalapú hálózatban minden lehetséges címzett számára fenn kell tartani egy bejegyzést,
  - míg a virtuálisáramkör-alapú hálózatban csak az egyes áramkörök számára kell egy-egy bejegyzés
  - **Megjegyzendő:** a virtuális áramkörök ezen előnye sem ennyire egyértelmű,
    - mert az összeköttetést felépítő csomagokat is irányítani kell, ezek pedig ugyanúgy tartalmazzák a célcsomagok címét, mint a datagramok.

Virtuálisáramkör- és a datagramalapú hálózatok

- A virtuális áramkörök a szolgáltatásminőségi garanciák és a hálózaton belüli torlódáskezelés területén rendelkeznek némi előnnyel:
  - az erőforrásokat (puffereket, sávszélességet, processzoridőt) előre, az összeköttetés felépítésekor le tudják foglalni.
  - Mire a csomagok megérkeznek, a szükséges sávszélesség és útválasztó-kapacitás már rendelkezésre fog állni.
- Datagramalapú hálózatokban a torlódások elkerülése bonyolultabb kérdés

Virtuálisáramkör- és a datagramalapú hálózatok

- A tranzakciófeldolgozó rendszereknél:
  - egy virtuális áramkör felállításához és kitörléséhez szükséges többletteher könnyen felülmúlhatja az áramkör valódi hasznát
  - Ha a forgalom nagy része várhatóan ilyen típusú lesz, a hálózaton belüli kapcsolt virtuális áramkörök használatának nincs sok értelme
- Másfelől viszont hasznosnak bizonyulhatnak:
  - az olyan hosszú ideig működő virtuális áramkörök, mint amilyen a VPN két cég központja között,
  - amelyeket kézzel hoznak létre és hónapokig vagy évekig fennállnak.

Virtuálisáramkör- és a datagramalapú hálózatok

- **A virtuális áramkörök azonban sebezhetők is**
  - Ha egy útválasztó összeomlik, és elveszti memóriájának tartalmát az összes rajta keresztülhaladó virtuális áramkört meg kell szakítani
    - még ha egy másodperccel később újraindul is
  - Ezzel szemben, ha egy datagramalapú útválasztó megy tönkre:
    - csak azok a felhasználók szenvednek kárt, amelyeknek a csomagjai éppen ebben az útválasztóban álltak sorban,
    - sőt talán nem is mindegyik, mivel a küldő valószínűleg rövid időn belül újraküldi azokat

Virtuálisáramkör- és a datagramalapú hálózatok

- Egy kommunikációs vonal elvesztése végzetes a rajta keresztülhaladó virtuális áramkörök számára
  - de könnyen ellensúlyozható datagramok használata esetén
- A datagramok azt is lehetővé teszik az útválasztóknak, hogy kiegyenlítsék a hálózat terhelését,
  - mivel az útvonalak egy hosszabb csomagsorozat közben is módosíthatók

Útválasztó algoritmusok...

Útválasztó algoritmusok

- A hálózati réteg fő feladata, hogy a csomagokat a forrásgéptől a célgépig irányítsa.
- A legtöbb hálózatban a csomagoknak ehhez több útválasztón kell keresztülhaladni, több ugrást kell megtenni.
- Az egyetlen figyelemre méltó kivétel az adatszóró hálózatok esete, de az útválasztás itt is téma lehet, ha a forrás és a cél nem ugyanazon hálózaton van
- Azok az algoritmusok, amelyek az útvonal meghatározását szolgálják,
    - és azok az adatstruktúrák, amelyeket az algoritmusok használnak, a hálózati réteg tervezésének egyik legfontosabb területét alkotják.

Útválasztó algoritmusok

- Az **útválasztó algoritmus** (routing algorithm):
  - a hálózati réteg szoftverének azon része, amely azért a döntésért felelős, hogy egy bejövő csomag melyik kimeneti vonalon kerüljön továbbításra
- Ha a hálózat belül datagramokat használ, ezt a döntést újra meg újra meg kell hozni minden beérkező adatcsomagra,
  - mivel lehet, hogy a legjobb útvonal a legutóbbi meghatározás óta változott
- Ha a hálózat belül virtuális áramköröket használ, akkor útválasztó döntéseket csak új virtuális áramkör felépítésekor kell meghozni.
  - Ezután az adatcsomagok már csak az előzőleg kialakított útvonalat követik.
- Ezt az utóbbi esetet néha **viszony-útválasztásnak** (session routing) is nevezik,
  - mivel az útvonal érvényben marad a teljes felhasználói viszony (mint például a VPN-en keresztüli bejelentkezés) alatt.

Útválasztó algoritmusok

- Néha célszerű megkülönböztetnünk az útválasztást, azaz a választandó útvonalról való döntést és a továbbítást,
  - ami a csomag beérkezésekor történik
- Képzeljük azt, hogy az útválasztó két folyamatot tartalmaz:
  - Az egyik kezeli a beérkező csomagokat: mindegyik számára kikeres egy kimeneti vonalat az útválasztó táblázatokból.
    - Ez a folyamat a **továbbítás** (forwarding).
  - A másik folyamat az útválasztó táblázatok feltöltéséért és karbantartásáért felelős
    - itt kerülnek be a képbe az útválasztó algoritmusok

Útválasztó algoritmusok

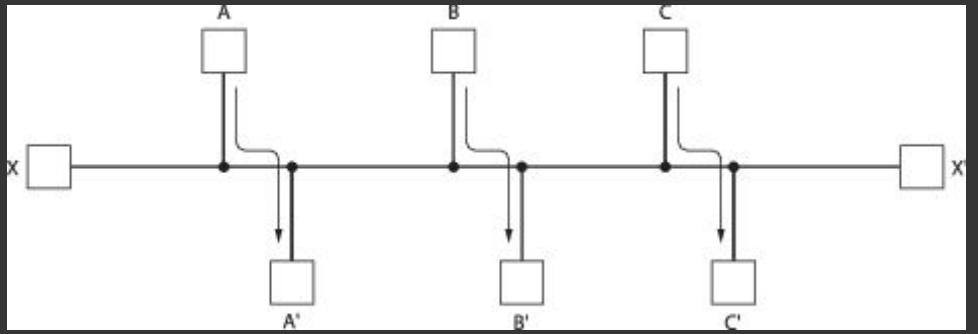
- Bizonyos tulajdonságok, amelyek kívánatosak egy útválasztó algoritmus esetében:
  - helyesség, egyszerűség, robusztusság, stabilitás, igazságosság, optimalitás és hatékonyság.
- A helyesség és az egyszerűség magától értetődő
- **Robosztusság:**
  - Amikor egy nagyobb hálózatot üzembe helyeznek, elvárják, hogy az évekig rendszerszintű hibáktól mentesen működjön.
  - Ez alatt az idő alatt mindenféle hardver- és szoftverhiba adódik.
    - Hosztok, útválasztók és vonalak többször is tönkremennek és megjavulnak, és a topológia is sokszor fog változni.
- Az útválasztó algoritmusnak képesnek kell lennie:
  - hogy megbirkózzon a topológiában és a forgalomban bekövetkező változásokkal anélkül, hogy az összes hoszton futó összes munkát meg kellene szakítani.
    - Mekkora probléma lenne az, ha a hálózatot mindig újra kellene indítani, ahányszor csak valamelyik útválasztó összeomlik.

Útválasztó algoritmusok

- A stabilitás szintén az útválasztó algoritmus fontos célja
- Léteznek algoritmusok, amelyek soha nem közelítik meg az egyensúlyi állapotot (egy fix útvonalhalmazt), bármilyen hosszan fussanak is.
- Egy stabil algoritmus eléri az egyensúlyt, és meg is őrzi azt.
- Ennek gyorsan kell történnie, mivel a kommunikáció megszakadhat, mire az útválasztó algoritmus elérné az egyensúlyi állapotot.

Útválasztó algoritmusok

- Az igazságosság és hatékonyság ugyancsak nyilvánvalónak tűnik
  - de ezek gyakran egymásnak ellentmondó célok
- Erre a konfliktusra példa: Tegyük fel, hogy elegendő forgalom van A és A', B és B', illetve C és C' között ahhoz, hogy telítődjenek a vízszintes adatkapcsolatok.
- A teljes adatfolyam maximalizálása érdekében az X és X' közti forgalmat teljesen be kellene szüntetni.
- Sajnos elképzelhető, hogy X és X' ezt nem így látja.
- Ezért kompromisszumot kell kötni a globális hatékonyság és az egyes összeköttetések azonos elbírálása között.



Útválasztó algoritmusok

- Mielőtt megpróbálnánk kompromisszumot keresni az igazságosság és hatékonyság között, el kell határoznunk, mit is akarunk optimalizálni.
- A hálózati forgalom hatékony küldésénél
  - az egyik nyilvánvaló jelölt az átlagos csomagkésleltetés minimalizálása,
  - de ugyanígy a teljes hálózati átbocsátóképesség maximalizálása is.
- Ez a két cél is ütközik:
  - ha bármely sorbanállási rendszert a teljesítőképessége határán működtetünk,
  - az hosszú sorbanállási késleltetést von maga után.
- Kompromisszumként sok hálózatban a csomagok által megtett utat, illetve az ugrások számát próbálják meg minimalizálni,
  - mivel ez csökkenti a csomagonként felhasznált sávszélességet, és ezáltal az átbocsátóképesség is javul.

Útválasztó algoritmusok

- Az útválasztó algoritmusok két nagy osztályba sorolhatók: **adaptív** és **nem adaptív algoritmusok**
- A nem adaptív algoritmusok (nonadaptive algorithms) döntéseikben nem támaszkodnak mérésekre vagy becslésekre az aktuális forgalomról és topológiáról
- Ehelyett az I-től J-ig használandó útvonalat (minden I-re és J-re) előre, offline módon számolják ki, és a hálózat indításakor letöltik az útválasztókba
- Ezt az eljárást néha **statikus útválasztás**nak (static routing) is szokták nevezni.
- Mivel a statikus útválasztás nem reagál a hibákra, főként olyan helyzetekben hasznos, ahol az útválasztás egyértelmű.

Útválasztó algoritmusok

- **Az adaptív algoritmusok (adaptive algorithms):** úgy változtatják útválasztó döntéseiket, hogy azok tükrözzék a topológiában és néha a forgalomban is történt változásokat.
- Ezek a dinamikus útválasztó algoritmusok abban különböznek egymástól:
  - hogy honnan kapják az információt (például helyileg, a szomszédos útválasztóktól vagy az összes útválasztótól),
  - mikor változtatják meg az útvonalakat (például amikor a topológia változik, vagy minden másodpercben, amikor a terhelés változik),
  - és milyen metrikát használnak az optimalizáláshoz (például a távolságot, az ugrások számát vagy a becsült áthaladási időt).

Optimalitási elv

- Az útválasztó algoritmusok működésének egyik alapja az optimalitási elv,
  - kimondja, hogy **egy optimális útvonal részútvonalai is optimálisak**
- Ez az elv lehetővé teszi, hogy a hálózat hatékonyan építse fel az optimális útvonalakat kisebb egységekből.

Rövid és leghosszabb út problémája

- **A hálózat egy gráfként modellezhető**, ahol a csomópontok (routerek) a gráf csúcsai, a köztük lévő kapcsolatok pedig élek
- Minden élhez egy költség (súly) rendelhető, amely az adott kapcsolat „árát” fejezi ki
- **Mit jelent a „legrövidebb út”?**
- Fontos megjegyezni, hogy a „legrövidebb út” nem feltétlenül a fizikai távolságot jelenti.
- A gyakorlatban egy költségfüggvényt minimalizálunk, amely különböző tényezőket vehet figyelembe.

Rövid és leghosszabb út problémája

- A leggyakrabban használt költségek:
  - késleltetés (latency)
  - sávszélesség (bandwidth)
  - terheltség (load)
  - megbízhatóság
  - ugrások száma (hop count)
- A hálózat tervezője határozza meg, hogy melyik tényező milyen súllyal szerepel a költségszámításban.
- Számos algoritmus ismert egy gráf két csomópontja közti legrövidebb út kiszámítására. Az egyik legismertebb Dijkstrától [1959] származik

Rövid és leghosszabb út problémája

Példa szemléltetés

- Tegyük fel, hogy egy hálózatban a következő útvonalak léteznek:

$$A \rightarrow B \rightarrow D \rightarrow C$$

$$A \rightarrow E \rightarrow C$$

- Ha az első út költsége 10, a másodiké pedig 6, akkor a hálózat a második útvonalat választja, mivel annak teljes költsége kisebb.
- Ez a döntés minden routerben lokálisan történik, de az algoritmusok biztosítják, hogy globálisan is optimális útvonal alakuljon ki.

A leghosszabb út problémája

- Hasonló módon fogalmazható meg: a cél, hogy a maximális összköltségű útvonalat találjuk meg.
- Ez a probléma általános gráfok esetén lényegesen bonyolultabb, és gyakran nem is alkalmazható közvetlenül hálózati útválasztásban, mivel:
  - ciklusok esetén végtelen hosszúságú utak is létezhetnek
  - nem biztosít hatékony adatátvitelt
- A hálózati rétegben ezért elsősorban a legrövidebb út problémájával foglalkozunk. Két legismertebb megközelítés:
  - **távolságvektor algoritmusok:** iteratív költségfrissítés
  - **kapcsolatállapot algoritmusok:** teljes gráf + legrövidebb út számítás
- Ezek az algoritmusok biztosítják, hogy a hálózat hatékonyan és megbízhatóan továbbítsa az adatokat

Távolságvektor alapú útválasztás

- A hálózati réteg egyik alapvető dinamikus útválasztási módszerét jelentik
- Ezek az algoritmusok **a hálózat csomópontjai közötti távolságok iteratív meghatározásán alapulnak**
  - lehetővé teszik az optimális útvonalak fokozatos kialakulását
- A távolságvektor-alapú útválasztás (distance vector routing) alapja:
  - minden útválasztónak egy táblázatot (vagyis egy vektort) kell karbantartania,
  - ebben minden célhoz szerepel a legrövidebb ismert távolság, és annak a vonalnak az azonosítója, amelyiken a célhoz lehet eljutni.
  - A táblázatokat a szomszédokkal való információcsere útján frissítik.
  - Végül minden útválasztó tudni fogja a legjobb adatkapcsolatot minden célcím megtalálásához.

Távolságvektor alapú útválasztás

- **Működés lépésről lépésre**
  - **Inicializáció:** Minden csomópont ismeri a közvetlen szomszédaihoz vezető utak költségét. A saját magához vezető távolság nulla.
  - **Információcsere:** A csomópontok elküldik aktuális távolságvektorukat a szomszédos csomópontoknak.
  - **Frissítés (relaxáció):** A beérkező információk alapján minden csomópont újraszámolja a saját távolságvektorát.
  - A frissítés alapja a következő összefüggés:
    - új_költség = szomszédig vezető költség + szomszéd által jelentett költség a célhoz
  - **Iteráció:** A folyamat addig ismétlődik, amíg a hálózat stabil állapotba nem kerül (konvergencia)

Távolságvektor alapú útválasztás

Matematikai háttér:

- A távolságvektor algoritmus a Bellman–Ford elven alapul. Az egyenlet a következőképpen írható fel:

$$D_x(y) = \min_v \{c(x,v) + D_v(y)\}$$

ahol:

- $D_x(y)$: az x csomópont becsült távolsága y-hoz
- $c(x,v)$: az x és v közötti közvetlen kapcsolat költsége
- $D_v(y)$: a szomszéd (v) által ismert távolság y-hoz

Távolságvektor alapú útválasztás

- **Példa működés**
  - Tegyük fel, hogy egy router nem ismeri közvetlenül egy távoli cél elérési útját
  - Egy szomszédjától azonban kap információt, amely szerint az adott cél elérhető bizonyos költséggel.
  - A router hozzáadja a szomszédhoz vezető saját költségét ehhez az értékhez, és ha ez kedvezőbb, mint a korábbi becslés, frissíti a tábláját.
  - Így a hálózatban az információ fokozatosan „terjed”, és minden csomópont egyre pontosabb képet kap az optimális útvonalakról

Konvergencia: A konvergencia azt az állapotot jelenti, amikor minden csomópont stabil és helyes útválasztási információval rendelkezik.

- A távolságvektor algoritmus idővel konvergál, azonban ez a folyamat lassabb lehet, különösen nagy hálózatokban.

Távolságvektor alapú útválasztás

- **Problémák és korlátok**
  - A távolságvektor algoritmus egyik legismertebb problémája a **count-to-infinity jelenség**.
  - Ez akkor fordul elő, amikor egy útvonal megszűnik, de a routerek egymástól kapott hibás információk alapján továbbra is elérhetőnek hiszik azt, és fokozatosan növelik a költséget.
  - Ez a folyamat lassú konvergenciához és hibás útválasztáshoz vezethet.
- **Problémakezelési módszerek:** A gyakorlatban több technikát alkalmaznak a problémák csökkentésére:
  - **Split horizon:** egy router nem hirdet vissza információt arra az irányra, ahonnan azt kapta
  - **Poison reverse:** a hibás útvonalakat végtelen költséggel jelölik
  - **Hold-down timer:** ideiglenesen figyelmen kívül hagyják a változásokat

Kapcsolatállapot alapú útválasztás

- A kapcsolatállapot alapú útválasztó algoritmusok a hálózati réteg fejlettebb dinamikus útválasztási módszerei közé tartoznak
- Ezek az algoritmusok a hálózat teljes topológiájának ismeretére épülnek,
  - lehetővé teszik az optimális útvonalak gyors és pontos meghatározását
- **A módszer alapelve:**
  - minden csomópont részletes információt gyűjt a közvetlen kapcsolatairól,
  - majd ezt az információt eljuttatja a hálózat többi csomópontjához.
  - Ennek eredményeként minden router ugyanazzal a teljes hálózati képpel rendelkezik.

Kapcsolatállapot alapú útválasztás

A kapcsolatállapot-alapú útválasztás mögötti öt logikai lépés:

1. Szomszédfelderítés: A router meghatározza, hogy mely csomópontokkal áll közvetlen kapcsolatban. Ez általában speciális üzenetek (hello üzenetek) segítségével történik.

2. Költségmeghatározás: beállítani a távolság vagy a költség értékét a minden szomszédjáig mért késleltetés alapján. A router minden közvetlen kapcsolatához költséget rendel. Ez lehet például:

- késleltetés
- sávszélesség
- terhelés

Kapcsolatállapot alapú útválasztás

3. Információ terjesztése (flooding): A router elkészíti a saját kapcsolatállapot leírását (Link-State Advertisement, **LSA**), és ezt elküldi a hálózat többi csomópontjának.

- A flooding mechanizmus biztosítja, hogy minden router megkapja az összes szükséges információt.

4. Topológia felépítése: Minden router a beérkezett információk alapján felépíti a hálózat teljes gráfját

5. Útvonal számítása: A router a gráfon legrövidebb út algoritmust alkalmaz (általában Dijkstra-algoritmust), hogy meghatározza az optimális útvonalakat minden célállomás felé.

- Lényegében a teljes topológia eljut az összes útválasztóhoz.
- Ezután a Dijkstra-algoritmust futtatják az útválasztók, hogy megtalálják a legrövidebb utat az összes többi útválasztóhoz.

Kapcsolatállapot alapú útválasztás

A Dijkstra-algoritmus szerepe: A kapcsolatállapot alapú útválasztás kulcseleme a legrövidebb út kiszámítása. Ehhez a leggyakrabban alkalmazott módszer a Dijkstra-algoritmus.

- Az algoritmus lépésről lépésre építi fel a legrövidebb út fát a forráscsomópontból kiindulva, mindig a legkisebb költségű csomópontot választva.
- Ennek eredményeként minden router meghatározza:
  - a legjobb útvonalat
  - a következő ugrást minden célhoz

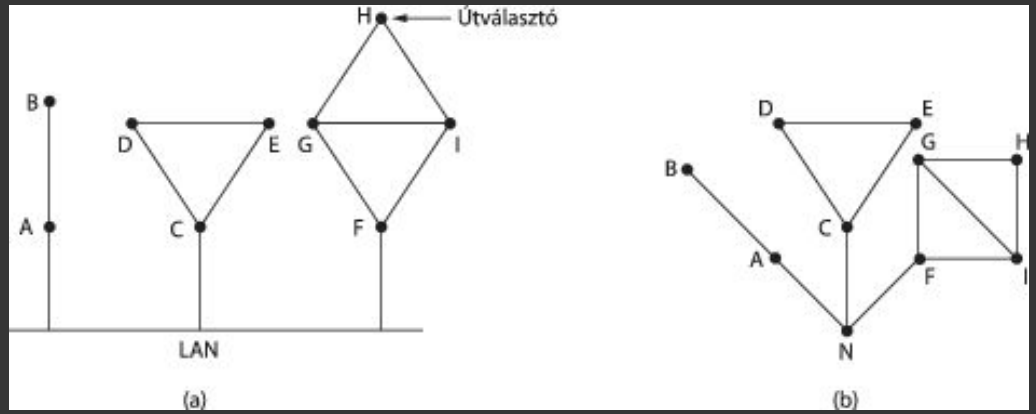
Kapcsolatállapot alapú útválasztás

Szomszédok megismerése:

- Amikor egy útválasztó beindul, az első feladata, hogy megtudja, kik a szomszédai.
- Ezt a célt úgy éri el, hogy egy speciális hello csomagot küld ki minden hozzá kapcsolódó kétpontos vonalon.
- A másik végen levő útválasztótól elvárja, hogy küldjön vissza egy választ, amelyben közli azonosítóját.
- **Ezeknek a neveknek globálisan egyedieknek kell lenniük:**
    - mert amikor később egy távoli útválasztó azt hallja, hogy három útválasztó az F-hez csatlakozik, el kell döntenie, hogy mind a három ugyanaz az F vagy sem

Kapcsolatállapot alapú útválasztás

- Amikor két vagy több útválasztót üzenetszórásos adatkapcsolat köt össze (például kapcsoló, gyűrű vagy klasszikus Ethernet), a helyzet kicsivel bonyolultabb.
- A következő ábra egy üzenetszórásos LAN-t mutat, amelyhez három útválasztó: A, C és F kapcsolódik közvetlenül.
- Mindegyik útválasztóhoz egy vagy több további útválasztó kapcsolódik



Kapcsolatállapot alapú útválasztás

- Az üzenetszórásos LAN minden csatlakoztatott útválasztópár között kapcsolatot létesít
- A LAN több kétpontos típusú adatkapcsolatként történő modellezése azonban növeli a topológia méretét és üzenetek elvesztéséhez vezet.
- Jobb módszer, ha a LAN-t magát egy csomópontnak tekintjük, amint azt az ábra mutatja.
- Itt egy új, mesterséges csomópontot vezettünk be, az N-t, amihez A, C és F kapcsolódik.
- Kiválasztunk a LAN-on egy kijelölt útválasztót (designated router), amely az N szerepét fogja játszani az útválasztó protokollban.
- A tényt, hogy lehetséges A-tól C-ig eljutni a LAN-on, itt az ANC út mutatja.

Az adatkapcsolat költségének beállítása

- **Követelmény:** minden adatkapcsolat rendelkezzen a legrövidebb út megtalálására vonatkozó távolság vagy költség mérőszámmal.
- A szomszédok elérésének költsége beállítható automatikusan, vagy azt beállíthatja a hálózat operátora.
- Általános választási lehetőség az adatkapcsolat költségének beállítására, hogy **a költség fordítottan arányos az adatkapcsolat sávszélességével.**
    - Pl. az 1 Gb/s-os Ethernet költsége 1, a 100 Mb/s-os Etherneté pedig 10.
    - Ez a nagyobb kapacitású utakat jobb választásnak tekinti

Az adatkapcsolat költségének beállítása

- **Ha a hálózat földrajzilag kiterjedt:** az adatkapcsolatok késleltetését is figyelembe lehet venni a költség kiszámításánál
- Így a rövidebb adatkapcsolaton lévő útvonalak jobb választásnak bizonyulnak.
- **A késleltetés meghatározásának legközvetlenebb módja egy speciális echo csomag kiküldése a vonalon**
  - amelyet a másik oldalnak azonnal vissza kell küldenie
- A körülfordulási idő megmérésével és kettővel osztásával a küldő útválasztó elfogadható becslést kaphat a késleltetés mértékéről.

A kapcsolatállapot-csomagok összeállítása

- Az információcseréhez szükséges adatok összegyűjtése után a következő lépés:
  - minden útválasztó összeállít egy csomagot, amely az összes adatot tartalmazza
  - A csomag a feladó azonosítójával kezdődik, amit egy sorszám és egy korérték végül pedig a szomszédok listája követ.
  - Minden szomszédhoz megadják a feléje tapasztalható költséget is.

A kapcsolatállapot-csomagok összeállítása

- A kapcsolatállapot-csomagok összeállítása egyszerű
- A nehézséget annak eldöntése jelenti, hogy mikor állítsuk össze azokat.
- **Az egyik lehetőség:** periodikusan, vagyis szabályos időközönként.
- **Egy másik lehetőség:** amikor valami jelentős esemény történt
  - például egy vonal vagy szomszéd meghibásodott, megjavult vagy megváltoztatta a tulajdonságait
- **A kapcsolatállapot-csomagok szétosztása általában az elárasztást használjuk**
- Hogy az elárasztást kézben tartsák, minden csomag tartalmaz egy sorszámot, amely minden egyes csomagküldésnél eggyel növekedik.

Kapcsolatállapot alapú útválasztás

- Az útválasztók számon tartanak minden (forrásútválasztó, sorszám) párt, amit csak látnak
- Amikor egy új kapcsolatállapot-csomag érkezik, összevetik a már látott csomagok listájával
- Ha új, eddig nem látott csomag érkezett, akkor továbbítják minden vonalon, kivéve azon, amelyiken érkezett.
  - Ha másodpéldány, eldobják
- Ha egy csomag olyan sorszámmal érkezik, amely kisebb, mint a legnagyobb már látott sorszám, akkor az útválasztó elavult csomagnak tekinti
  - és nem továbbítja,
  - hiszen az útválasztó rendelkezik már ennél frissebb információval is.

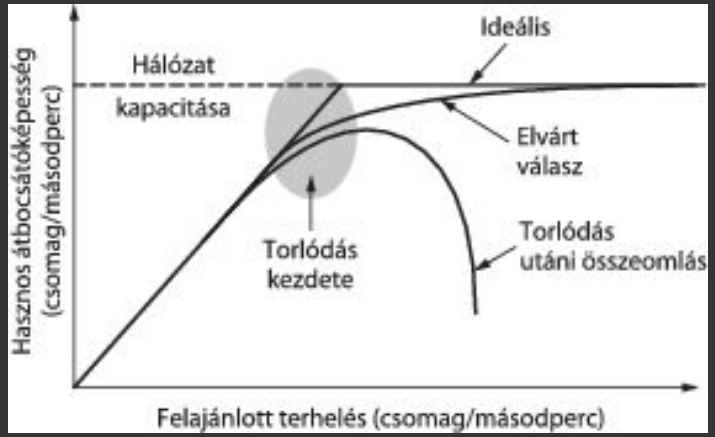
Kapcsolatállapot alapú útválasztás

- Ennek az algoritmusnak van néhány problémája, de ezek kezelhetők
- 1) Ha a sorszámok átfordulnak, zűrzavar fog eluralkodni:
  - Megoldás, hogy 32 bites sorszámokat kell használni.
  - Másodpercenként egy kapcsolatállapot-csomag elküldését feltételezve, az átfordulás 137 év múlva következne be
- 2) Ha egy útválasztó valamikor összeomlik, elfelejti, melyik sorszámnál tartott
  - Ha újra 0-ról kezdi, a következő általa küldött csomagot másodpéldánynak fogják tekinteni, és nem továbbítják.
- 3) Ha egy sorszám megsérül: 65 540 érkezik meg 4 helyett (egy 1 bites hiba),
  - az 5-től 65 540-ig tartó csomagokat elavultként visszautasítja, mivel a jelenlegi sorszámot 65 540-nek hiszi

Torlódáskezelési algoritmusok...

Torlódáskezelés

- Amikor túl sok csomag van jelen a hálózatban (vagy egy részében), a teljesítőképesség visszaesik, a csomagküldés pedig késleltetést szenved.
- Ezt a helyzetet **torlódásnak (congestion)** nevezzük
- A hálózati és a szállítási réteg megosztja a torlódáskezelés felelősségét
- Amikor torlódás lép fel a hálózatban, a hálózati réteg ezt közvetlenül érzékeli, és neki kell meghatározni végül, hogy mi történjen a többletcsomagokkal.



Torlódáskezelés

- Amikor a hosztok által a hálózatba beadott csomagok száma a hálózat átviteli kapacitásán belül marad, akkor minden csomag kézbesítésre kerül
  - a kézbesített csomagok száma arányos az elküldött csomagok számával
- Ahogy azonban az átvitelre szánt (felajánlott) terhelés eléri a hálózat átviteli kapacitását:
  - a forgalmi löket alkalmanként megtölti az útválasztók puffereit és néhány csomag elveszik
- Ezek az elveszett csomagok felemésztik a kapacitás egy részét, így a kézbesített csomagok száma az ideális szint alá csökken
  - A hálózaton torlódás alakul ki
- **Ha a hálózat nincs jól megtervezve, akkor a torlódás összeomlást okozhat**
  - amelynek hatására a teljesítőképesség gyorsan csökken, amint a terhelés túlnő a kapacitáson.

Torlódáskezelés

- Ez azért történhet meg, mert a csomagok késleltetése a hálózatban olyan nagy lehet, hogy azok már nem hasznosak, amikor elhagyják a hálózatot
- Másféle hibamód jelentkezik akkor, amikor a feladók újraküldenek nagy késleltetésű csomagokat, mert azt hiszik, hogy azok elvesztek.
    - Ebben az esetben ugyanannak a csomagnak több példányát továbbítja a hálózat, ez ismét a kapacitás pazarlásával jár.
- Az ábra y tengelye a **hasznos átbocsátóképességet (goodput)** ábrázolja
- Ez az az arány, amellyel a hálózat a hasznos csomagokat kézbesíti

Torlódáskezelés

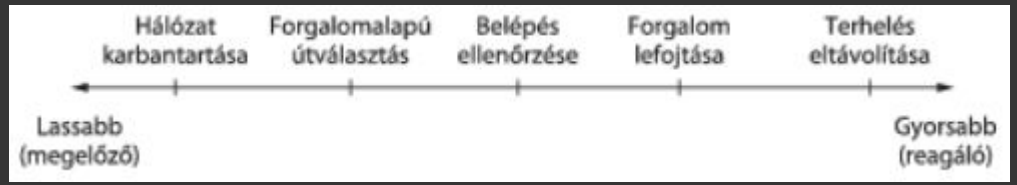
- Olyan hálózatot szeretnénk kialakítani:
  - amely ahol csak lehet, elkerüli a torlódást,
  - és nem történik a torlódás után összeomlás, amennyiben mégis torlódás alakulna ki.
- **Sajnos a torlódás nem kerülhető el teljesen:**
  - Ha hirtelen nagy mennyiségű csomag érkezik három vagy négy bejövő vonalon,
  - és ezek közül mindegyiknek ugyanarra a kimenő vonalra van szüksége, akkor felépül egy várakozási sor.
- Ha a memória kevés ahhoz, hogy mindet befogadja, akkor csomagok fognak elveszni.

Torlódáskezelés

- **Több memória hozzáadása egy adott pontig segíthet**
  - Nagle [1987] kimutatta, hogy ha az útválasztóknak végtelen kapacitású memóriája van, a torlódás nem javul, hanem rosszabbá válik
  - mivel mire a csomagok a sor elejére kerülnek, az időzítésük (akár többször is) lejár, és másodpéldányok kerülnek továbbításra
  - Ez ront a helyzeten, nem javít – torlódás utáni összeomláshoz vezet
- A kis sávszélességű adatkapcsolatok és útválasztók szintén okozhatnak torlódást
  - amelyek a vonali kapacitásnál lassabban dolgozzák fel a csomagokat
- Ebben az esetben a helyzet javítható azáltal, hogy a forgalom egy része átirányításra kerül a szűk keresztmetszetről a hálózat többi részére
  - Végül azonban a hálózat összes régiója torlódik.
- Ebben a helyzetben csak a terhelés csökkentése vagy gyorsabb hálózat kiépítése jelenthet megoldást

Torlódáskezelés alapelvei

- A torlódás kialakulása azt jelenti, hogy a terhelés (ideiglenesen) nagyobb, mint amit az erőforrások (a hálózat egy részében) kezelni tudnak.
- **Két megoldási lehetőség merül fel:**
  - az **erőforrások növelése** vagy a **terhelés csökkentése**.
- Ezek a megoldások különböző időskálán alkalmazhatók:
  - a torlódás kialakulása előtt annak megakadályozása érdekében,
  - vagy reagálhatnak a már kialakult torlódásra.



Torlódáskezelés alapelvei

- **A legalapvetőbb mód a torlódás elkerülésére, ha olyan hálózatot építünk, amely megfelel a rajta átmenő forgalomnak**.
- Ha van kis sávszélességű adatkapcsolat az útvonalon, amely mentén a legtöbb forgalom halad, akkor a torlódás nagy valószínűséggel bekövetkezik
- Néha további erőforrások dinamikusan használhatók, amikor súlyos torlódás következik be:
  - Pl. tartalék útválasztókat is üzembe lehet helyezni, melyek egyébként csak vésztartalékul szolgálnak,
  - vagy sávszélesség vásárolható a szabad piacon.
- Nagyon gyakran először a nagy kihasználtságú adatkapcsolatok és útválasztók kerülnek frissítésre
- Ezt karbantartásnak hívják és havi ütemezés szerint történik, amelyet hosszú távú forgalmi trendek vezérelnek

Torlódáskezelés alapelvei

- A meglévő hálózati kapacitás legjobb kihasználása érdekében az útvonalak a forgalmi minták alapján személyre szabhatók.
- Ezek a forgalmi minták a nap folyamán változnak, ahogy a különböző időzónában lévő felhasználók felkelnek, illetve elmennek aludni.
- **Például:** az útvonalak módosíthatók úgy, hogy a nagy kihasználtságú útvonalakról elvigyék a forgalmat a legrövidebb útvonal súlyának módosításával
- Ezt **forgalomalapú útválasztás**nak (traffic-aware routing) hívják.
- A forgalom több útvonal közötti felosztása is hasznos lehet.

Torlódáskezelés alapelvei

- Néha azonban a kapacitás egyáltalán nem növelhető
- Ekkor a torlódás leküzdésére az egyetlen mód az, hogy csökkentjük a terhelést
- Virtuálisáramkör-alapú hálózat esetén:
  - az új összeköttetések felépítése visszautasítható, ha azok a hálózat torlódását okoznák
  - Ezt **belépés-ellenőrzés**nek (admission control) hívjuk.
- Egy finomabb eljárás szerint, ha torlódás várható, akkor a hálózat visszajelzést küldhet azoknak a forrásoknak, amelyek forgalma felelős a problémáért.
- A hálózat megkérheti ezeket a forrásokat a forgalmuk csökkentésére, vagy saját maga lassíthatja le a forgalmat

Torlódáskezelés alapelvei

- Ennek a megoldásnak két nehézsége van:
  - hogyan azonosítjuk a torlódás kezdetét,
  - illetve hogyan tájékoztassuk a forrást arról, hogy lassítania kell a forgalmat.
- Az első probléma megoldása érdekében:
  - az útválasztók meg tudják figyelni az átlagos terhelést,
  - a sorbanállási késleltetést,
  - valamint a csomagvesztést
- Mindegyik esetben a növekvő szám torlódásnövekedést jelent
- A második probléma megoldása érdekében az útválasztóknak a forrásokkal egy visszacsatoló hurokban kell lenniük
- Ahhoz, hogy a séma megfelelően működjön, az időléptéket gondosan be kell állítani

Torlódáskezelés alapelvei

- Végül, ha már minden csődöt mondott, a hálózatot kényszeríteni kell a nem kézbesíthető csomagok eldobására
- Ennek általános neve **terheléslefojtás (load shedding)**
- Érdemes olyan csomagokat eldobni, amelyekkel megakadályozható a torlódás utáni összeomlás

Forgalomalapú útválasztás

- A hagyományos útválasztási megközelítések általában statikus vagy ritkán változó költségek alapján határozzák meg az adatcsomagok útvonalát.
- Ezek a költségek rendszerint a hálózat topológiájából vagy előre meghatározott paraméterekből származnak,
  - nem tükrözik a hálózat aktuális állapotát.
- A valós hálózatokban azonban a forgalom időben változik, és egy adott útvonal terheltsége jelentősen ingadozhat.
  - előfordulhat, hogy egy elvileg optimális útvonal a gyakorlatban túlterheltté válik,
  - ami megnövekedett késleltetéshez és csomagvesztéshez vezet.
- **A forgalomalapú útválasztás olyan megközelítést jelent, amely figyelembe veszi a hálózat aktuális terhelését**
  - ennek megfelelően dinamikusan módosítja az útválasztási döntéseket
- Ebben az esetben az útvonalakhoz rendelt költségek nem állandóak, hanem a hálózat pillanatnyi állapotától függenek.

Forgalomalapú útválasztás

- **A routerek folyamatosan figyelik a hálózat különböző jellemzőit:**
  - Pl. az egyes kapcsolatok kihasználtságát,
  - a sorban álló csomagok számát,
  - valamint az adatátvitel során tapasztalt késleltetést.
- Ezekből az adatokból képet alkotnak az egyes útvonalak terheltségéről, és ennek alapján módosítják a továbbítási döntéseiket.
- **A forgalomalapú útválasztás során tehát egy adott útvonal költsége növekszik, ha az adott útvonal túlterhelté válik.**
- Ennek hatására a routerek más, kevésbé terhelt útvonalakat részesítenek előnyben,
  - még akkor is, ha ezek hosszabbak vagy több ugrást tartalmaznak.
- Ez a megközelítés hozzájárul a hálózati terhelés egyenletesebb elosztásához.

Forgalomalapú útválasztás

- **Ha két lehetséges útvonal közül:**
  - az egyik rövidebb, de erősen terhelt, míg a másik hosszabb, de kevésbé kihasznált,
  - akkor a forgalomalapú útválasztás a második útvonalat választhatja.
- Ez csökkentheti az adatátvitel késleltetését és javíthatja a hálózat általános teljesítményét
- A módszer alkalmazása azonban nem mentes a problémáktól:
  - Az egyik legjelentősebb kihívás a stabilitás biztosítása
  - Ha a routerek túl gyorsan reagálnak a hálózat állapotának változásaira, akkor az útvonalak folyamatosan változhatnak.
  - Ez az úgynevezett **oszcilláció jelenség**éhez vezethet
    - amely során a forgalom ide-oda vándorol a különböző útvonalak között, ami tovább ronthatja a hálózat teljesítményét.

Forgalomalapú útválasztás

- A stabil működés érdekében a gyakorlatban különböző technikákat alkalmaznak
- Ilyen például:
  - az állapotinformációk időbeli átlagolása,
  - a frissítések gyakoriságának korlátozása,
  - valamint a döntések késleltetése.
- Ezek a módszerek csökkentik a túl gyors reakciók okozta instabilitást
- A modern hálózatokban a tisztán forgalomalapú útválasztás ritkán jelenik meg önálló formában.
- Ehelyett a hálózati protokollok gyakran kombinálják a statikus és dinamikus költségmodelleket,
  - és a terhelési információkat kiegészítő tényezőként használják az útvonalválasztás során.

Nyílt hurkú torlódáskezelés

- A **nyílt hurkú (open-loop) torlódáskezelés** célja a torlódás kialakulásának megelőzése
- Ebben a megközelítésben a hálózat működése előre meghatározott szabályok alapján történik,
  - a rendszer nem reagál közvetlenül a pillanatnyi hálózati állapotra.
- A nyílt hurkú megoldások során a hálózat tervezésekor olyan mechanizmusokat alkalmaznak, amelyek csökkentik a torlódás kialakulásának valószínűségét.
- **Ezek a mechanizmusok nem használnak visszacsatolást a hálózat aktuális állapotáról**

Nyílt hurkú torlódáskezelés

- **<u>Ilyen módszerek például:</u>**
  - forgalomszabályozás a forrás oldalon
  - csomagok ütemezése
  - csomageldobási stratégiák
  - erőforrások előzetes kiosztása
- A nyílt hurkú megközelítés előnye, hogy egyszerű és stabil, mivel nem reagál folyamatosan a hálózat változásaira.
- Ugyanakkor hátránya, hogy nem képes alkalmazkodni a váratlan forgalmi helyzetekhez

Zárt hurkú torlódáskezelés

- A **zárt hurkú (closed-loop) torlódáskezelés** ezzel szemben a hálózat aktuális állapotának folyamatos megfigyelésére és az arra adott reakciókra épül.
- Ez a megközelítés visszacsatolást alkalmaz
- Lehetővé teszi a rendszer számára, hogy érzékelje a torlódás kialakulását, és ennek megfelelően módosítsa működését

Zárt hurkú torlódáskezelés

A zárt hurkú rendszerek működése három alapvető lépésből áll:

- **Észlelés:** A hálózat felismeri a torlódás kialakulását, például megnövekedett késleltetés vagy csomagvesztés alapján.
- **Információ továbbítása:** A torlódásra vonatkozó információ eljut a megfelelő hálózati elemekhez (pl. forráshoz vagy routerekhez).
- **Beavatkozás:** A rendszer módosítja a forgalmat, például csökkenti az adatküldés sebességét vagy megváltoztatja az útvonalat.

Típusai:

- **Explicit visszacsatolás:** a hálózat közvetlenül jelzi a torlódást (pl. speciális jelzések segítségével)
- **Implicit visszacsatolás:** a torlódás közvetett módon érzékelhető (pl. csomagvesztés vagy késleltetés növekedése alapján)

Zárt hurkú torlódáskezelés

- **A zárt hurkú torlódáskezelés előnye:** képes alkalmazkodni a hálózat aktuális állapotához, és hatékonyabban kezeli a változó forgalmi viszonyokat
- Ugyanakkor bonyolultabb megvalósítást igényel, és nem megfelelő beállítás esetén instabilitást is okozhat
- Ezzel szemben a nyílt hurkú megközelítés egyszerűbb és stabilabb, de kevésbé rugalmas, és nem reagál a hálózat aktuális terhelésére

RED (Random Early Detection)

- A **RED (Random Early Detection)** egy aktív torlódáskezelési algoritmus
  - a hálózati rétegben és az útválasztók pufferkezelésében alkalmaznak a torlódás megelőzésére
- **A módszer célja:** még a puffer teljes telítődése előtt beavatkozzon, és ezzel csökkentse a hálózat túlterhelésének kialakulását
- A hagyományos megközelítésekben a routerek csak akkor dobnak el csomagokat, amikor a puffer megtelik.
- Ez a „**tail drop**” módszer azonban hirtelen csomagvesztést eredményez,
  - ami jelentős teljesítménycsökkenést okozhat, különösen olyan protokollok esetén, mint a TCP

RED (Random Early Detection)

- A RED ezzel szemben korábban, fokozatosan avatkozik be.
- **Működésének alapja:** a router nem a pillanatnyi pufferkihasználtságot figyeli, hanem annak egy átlagolt értékét
  - Ez segít kiszűrni a rövid idejű forgalmi ingadozásokat.
- A módszer három tartományt különböztet meg:
  - Alacsony terhelés: nincs csomageldobás
  - Közepes terhelés: valószínűségi csomageldobás
  - Magas terhelés: minden beérkező csomag eldobása

RED (Random Early Detection)

- A RED algoritmus folyamatosan számolja a puffer átlagos kihasználtságát. Ez az érték határozza meg a router viselkedését.
  - **1. Ha az átlagos pufferkihasználtság alacsony**
    - Amennyiben az átlagos terhelés egy alsó küszöbérték alatt van, a router minden csomagot elfogad.
    - Ebben az állapotban a hálózat nem tekinthető túlterheltnek.
  - **2. Ha a terhelés közepes**
    - Ha az átlagos pufferkihasználtság az alsó és felső küszöbérték közé esik, a router véletlenszerűen dob el csomagokat.
    - Az eldobás valószínűsége arányos a puffer telítettségével:
      - minél nagyobb a terhelés
      - annál nagyobb az eldobás valószínűsége
    - Ez a „korai jelzés” lehetővé teszi a forrás számára (például TCP esetén), hogy csökkentse az adatküldési sebességet.

RED (Random Early Detection)

- **3. Ha a terhelés magas**
  - Ha a puffer kihasználtsága meghalad egy felső küszöbértéket, a router minden beérkező csomagot eldob.
  - Ez már egy védelmi mechanizmus a teljes túlterhelés ellen.
- **Miért „random”?**
  - A RED egyik kulcsfontosságú jellemzője a véletlenszerű csomageldobás. Ez több szempontból is előnyös:
    - nem egyetlen adatfolyamot büntet
    - elkerüli a szinkronizációt (amikor minden TCP kapcsolat egyszerre lassul le)
    - egyenletesebb terheléscsökkentést biztosít
  - A RED egy tipikus zárt hurkú torlódáskezelési mechanizmus, mivel:
    - figyeli a hálózat aktuális állapotát
    - visszacsatolást ad (csomageldobás formájában)
    - befolyásolja a forrás viselkedését
  - Ezáltal a hálózat dinamikusan képes alkalmazkodni a terheléshez.

Köszönöm a figyelmet!