Tokaj-Hegyalja Egyetem

Hálózati Architektúrák és Protokollok

5. Adatkapcsolati réteg

Mileff Péter - Miskolci Egyetem

Adatkapcsolati réteg

- Az adatkapcsolati réteg feladata a nyers bitfolyam megbízható keretekbe (frames) történő csomagolása és továbbítása a fizikai réteg felett.
- Ez a réteg biztosítja, hogy a fizikai közegen átküldött adatok hibamentesen és a megfelelő sorrendben érkezzenek a célállomásra.

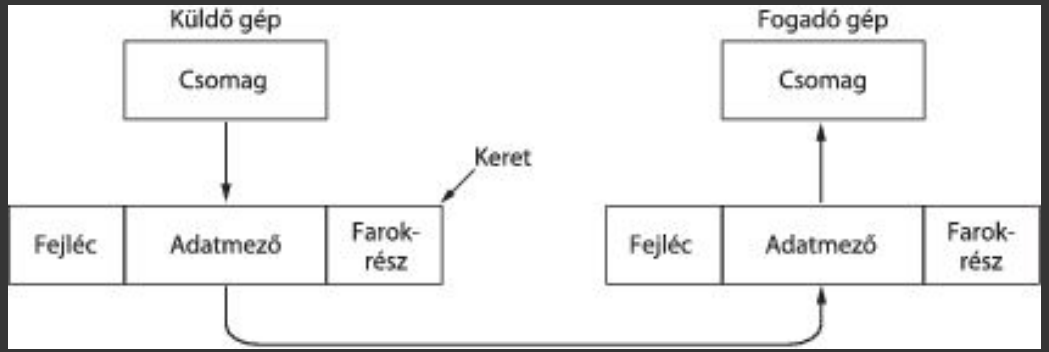
- **Hidat képez**
  - a fizikai réteg, amely az elektromos vagy optikai jelek küldéséért felelős
  - és a hálózati réteg között, amely az adatok csomagokba szervezéséért és útvonalválasztásáért felelős

Adatkapcsolati réteg feladatai

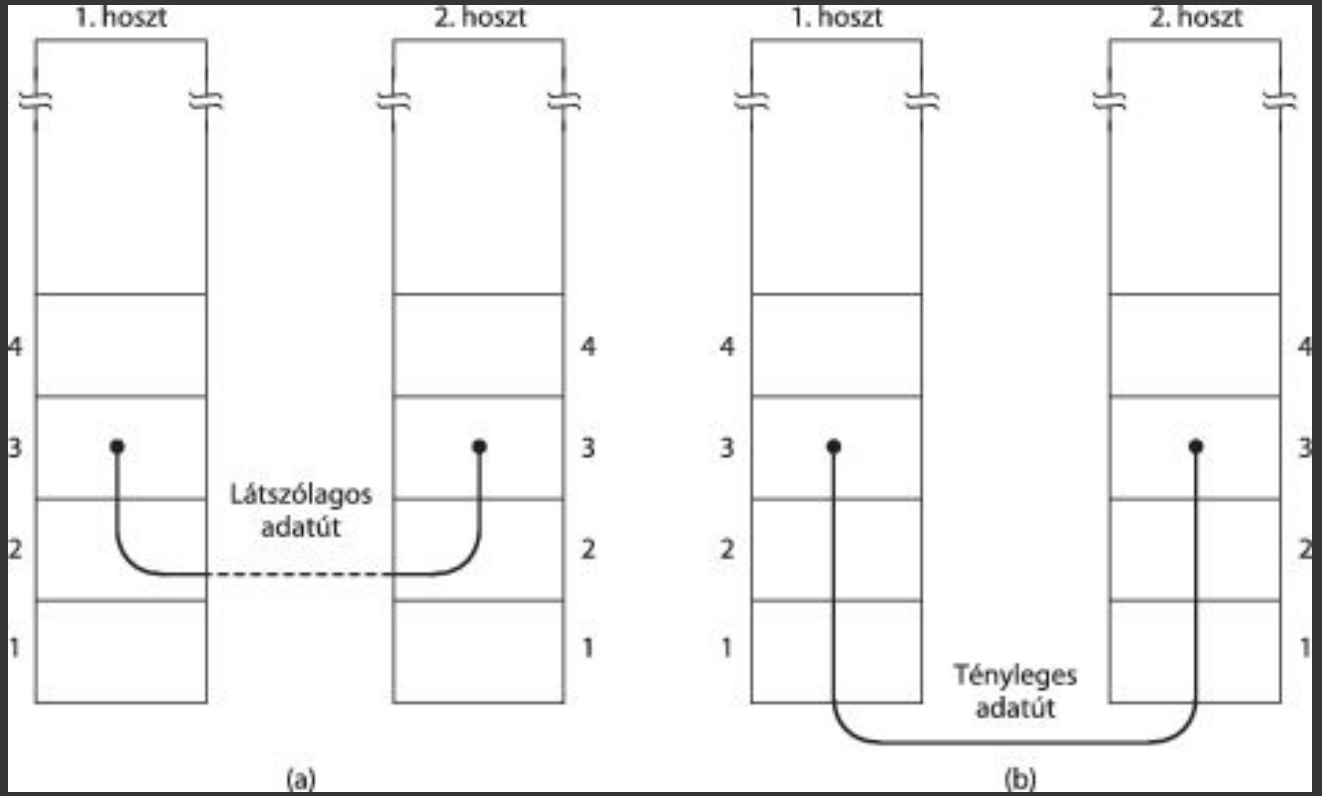
- **Keretezés (Framing)** – A nyers bitfolyam keretekbe (frame) történő bontása.
- **Címzés (Addressing)** – Fizikai (MAC) címek használata a célállomás azonosítására.
- **Hibaellenőrzés (Error Detection and Correction)** – Az átvitt adatok integritásának biztosítása hibadetektáló kódokkal.
- **Áramlásvezérlés (Flow Control)** – Megakadályozza, hogy a gyorsabb küldő túlterhelje a lassabb vevőt.
- **Hozzáférés-vezérlés (Access Control)** – Többszörös eszköz esetén biztosítja, hogy ne ütközzenek az adatok a közös közegen.

Adatkapcsolati réteg

- A célok eléréséhez az adatkapcsolati réteg a hálózati rétegtől kapott csomagokat **keret**ekbe (frame) ágyazza be az átvitel idejére.
- Minden keret tartalmaz egy keret**fejrész**t, egy **adatmező**t a csomag számára, valamint egy keret**farokrész**t.
- A keretek kezelése az alapja mindannak, amit az adatkapcsolati réteg véghezvisz.



Adatkapcsolati réteg



Adatkapcsolati réteg

- Az adatkapcsolati réteget különféle szolgáltatások megvalósítására készíthetik fel.
- A ténylegesen megvalósított szolgáltatások rendszerről rendszerre változhatnak.

- Három megvalósított lehetőség:

    - Nyugtázatlan összeköttetés nélküli szolgáltatás.
    - Nyugtázott összeköttetés nélküli szolgáltatás.
    - Nyugtázott összeköttetés-alapú szolgáltatás.

Adatkapcsolati réteg

- **Nyugtázatlan összeköttetés nélküli szolgáltatás:**
  - a forrásgép egymástól független kereteket küld a célgép felé, amely nem nyugtázza a keretek megérkezését.
    - Az Ethernet jó példa ilyen típusú protokollra.
  - Semmiféle kapcsolatot nem építenek fel előzetesen, illetve nem bontanak le az átvitel után.
  - Ha egy keret a vonali zaj miatt elveszik, nem történik kísérlet a hiba felfedezésére és helyreállítására az adatkapcsolati rétegben.
  - Ez a szolgáltatási osztály abban az esetben megfelelő, ha a hibaarány nagyon alacsony,
    - így a hibák javítása a felsőbb rétegekre hagyható,
  - valamint valós idejű forgalom esetén (például beszédátvitel),
    - amikor a későn érkező adat rosszabb, mint a hibás adat.

Adatkapcsolati réteg

- **Nyugtázott összeköttetés nélküli szolgáltatás:**
  - A következő lépés a megbízhatóság irányába
  - Ilyen szolgáltatás esetén sincs felépített kapcsolat, de minden egyes elküldött keret megérkezését nyugtázza a címzett állomás,
    - a küldő értesül arról, hogy a keret megérkezett-e, vagy sem.
  - Ha egy keret nem érkezik meg meghatározott időn belül, újra lehet küldeni.
  - Ez a szolgáltatás megbízhatatlan csatornák (például vezeték nélküli rendszerek) esetén hasznos.
    - A 802.11 Wi-Fi is ilyen típusú szolgáltatást alkalmaz.

Adatkapcsolati réteg

- A nyugtázás megvalósítása az adatkapcsolati rétegben sohasem elvárás, csak optimalizáció.
- A hálózati réteg mindig küldhet egy csomagot és megvárhatja a távoli partner által küldött nyugtát.
- Ha a nyugta az időzítő lejárta előtt nem érkezik meg, a küldő újraküldheti az egész üzenetet.
- A probléma ezzel a stratégiával az, hogy nem hatékony:
  - Az adatkapcsolatok rendszerint hardveres okokból általában szigorúan korlátolt hosszúságú keretekkel és ismert terjedési késleltetéssel rendelkeznek.
  - Ezeket a paramétereket a hálózati réteg nem ismeri.
    - Pl. küldünk egy nagy csomagot, amely 10 keretből áll, és ezek közül átlagosan kettő elvész, nagyon sokáig tarthat, amíg a csomag hibátlanul megérkezik.
    - Ha minden keretet egyenként nyugtázunk, és szükség esetén újraküldünk, sokkal gyorsabban jut át az egész üzenet.
    - Megbízható csatornákon, mint például az üvegszál, szükségtelen bonyolult adatkapcsolati protokoll használata, de vezeték nélküli átviteli csatornákon a csatorna megbízhatatlansága miatt nagyon is megéri.

Adatkapcsolati réteg

- A legkifinomultabb szolgáltatás, amit az adatkapcsolati réteg a hálózati rétegnek nyújthat: **az összeköttetés-alapú szolgáltatás**
- **Működése:**
  - a forrás- és a címzett számítógép felépít egy összeköttetést, mielőtt az adatátvitelt megkezdenék.
  - Minden elküldött keret sorszámozott, és az adatkapcsolati réteg garantálja:
    - a keretek valóban meg is érkezzenek,
    - minden keret pontosan egyszer és a megfelelő sorrendben érkezzen meg.
  - Az összeköttetés-alapú szolgáltatás így megbízható bitfolyamot biztosít a hálózati réteg folyamatai számára.
  - kifejezetten előnyös olyan megbízhatatlan csatornákon, mint a műholdas összeköttetések vagy nagy távolságú telefonvezetékek.
  - ha nyugtázott összeköttetés nélküli szolgáltatást használnánk:
    - az elveszett nyugták miatt a kereteket többször újra kellene küldeni és venni,
    - ami jelentős sávszélesség-veszteséget okozna.

Keretezés...

Keretezés

- Az adatkapcsolati réteg a fizikai réteg szolgáltatását veszi igénybe.
- A fizikai réteg nem tesz mást, mint a kapott bitsorozatot megpróbálja továbbítani a célhoz.
- Ha a csatorna zajos a fizikai réteg, hogy a hibák számát csökkentse, redundanciát ad a kimenő jelekhez,
  - de az érkező bitsorozat hibamentességét a fizikai réteg nem garantálja.
- A vett bitek száma lehet kevesebb, azonos vagy több, mint az elküldötteké, és a bitek értéke is különbözhet az eredetitől
- Az adatkapcsolati réteg feladata, hogy jelezze, illetve – ha szükséges – kijavítsa a hibákat.

Keretezés

- Az adatkapcsolati réteg egyik legfontosabb feladata a **keretezés (framing)**
- A fizikai rétegből érkező folyamatos bitfolyamot jól elkülöníthető, strukturált egységekre, úgynevezett keretekre bontja.
- **A keretezés alapvető célja:** hogy az adatátvitel során a vevő képes legyen azonosítani a beérkező adatok szerkezetét.
    - Ezért keretek nemcsak a tényleges adatot tartalmazzák, hanem különböző vezérlőinformációkat is
- Amikor a keret megérkezik a célhoz, az ellenőrző összeget újra kiszámolja az adatkapcsolati réteg.
- Ha ez különbözik attól, amit a keret tartalmaz, a réteg tudja, hogy hiba történt, és lépéseket tesz ennek kezelésére

Keretezés

- A keretezés egyik legfontosabb problémája a keretek határainak kijelölése
  - A bitfolyam keretekre tördelése bonyolultabb feladat
- Mivel a fizikai réteg folyamatos adatáramot biztosít, szükség van valamilyen módszerre annak meghatározására
  - hogy hol kezdődik és hol végződik egy keret.
  - ennek megoldására több különböző technika alakult ki.
- Egy jó megoldásnak lehetővé kell tenni a keretek kezdetének könnyű felismerését elenyésző csatorna-sávszélesség használata mellett
- A keretek három fő részből állnak: **fejléc**ből, **adatrész**ből és **ellenőrző mező**ből.
- A fejléc tartalmazza: a forrás- és célcímeket, valamint a protokollra vonatkozó információkat.
- Az adatrész hordozza a tényleges továbbítandó információt,
- Az ellenőrző mező a hibák felismerését szolgálja

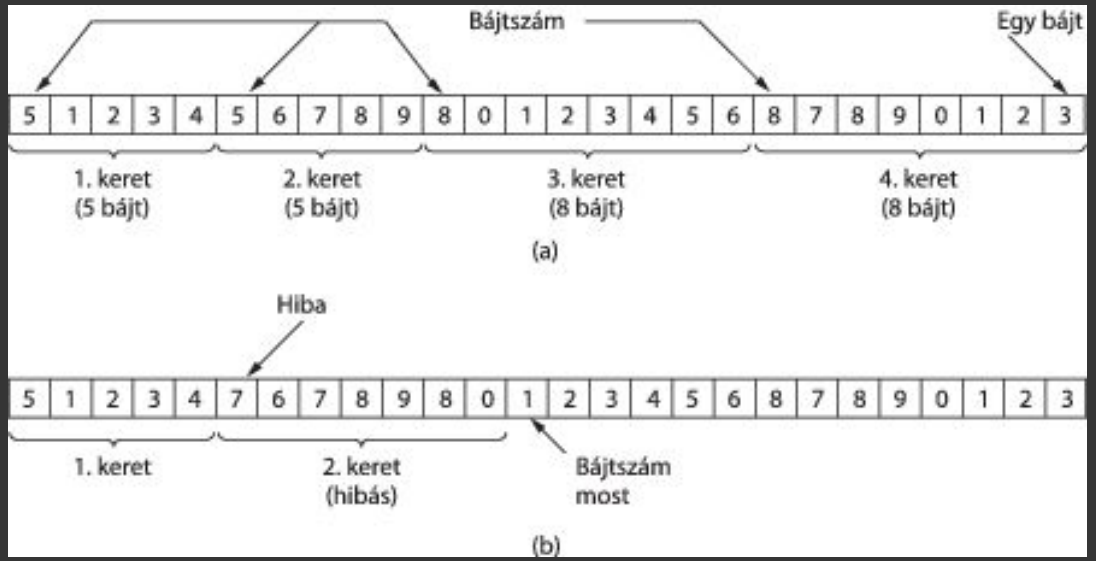
Keretezés

- Az egyik legegyszerűbb megközelítés a **fix hosszúságú keretek használata**
- Ebben az esetben minden keret azonos méretű, így a vevő egyszerűen megszámolja a biteket,
  - és ennek alapján határozza meg a kerethatárokat.
- Ez a módszer könnyen megvalósítható, azonban nem rugalmas,
- Gyakran pazarló, mivel a kisebb adatmennyiségek esetén is teljes méretű kereteket kell használni.

Keretezés

- Egy fejlettebb keretezési módszer a **hosszmező** alkalmazása
  - a keret fejlécében egy külön mező adja meg az adatrész méretét
- Amikor a címzett állomás adatkapcsolati rétege megkapja a keretben levő bájtok számát, tudni fogja:
  - mennyi bájtnak kell érkeznie,
  - hogy hol van a keret vége
- A módszer hatékonyabb, mint a fix hosszúságú keretek, azonban érzékeny a hibákra:
  - ha a hosszmező megsérül, a teljes keret feldolgozása hibássá válhat

Keretezés



Egy bájtfolyam (a) hiba nélkül (b) egy hibával

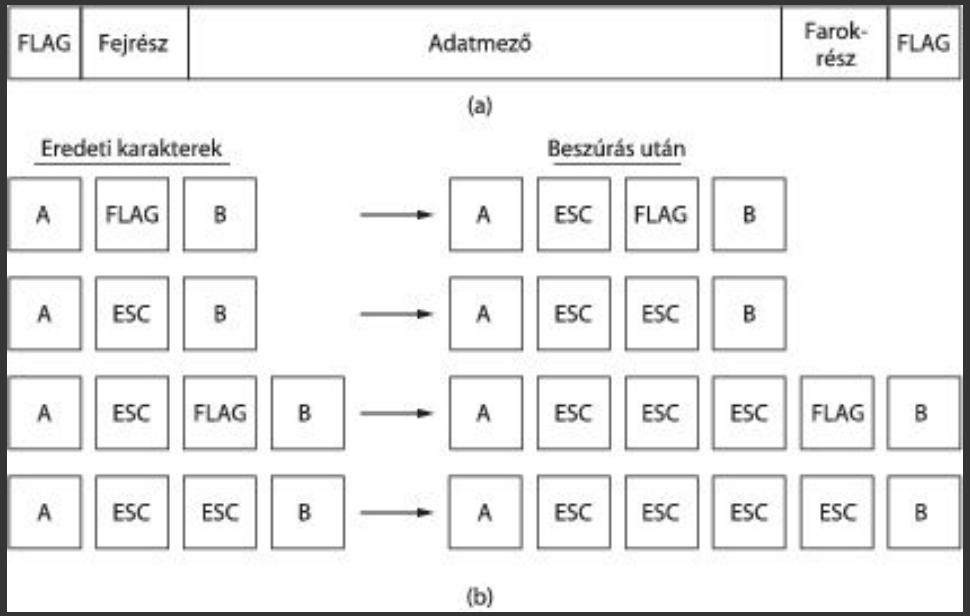
Keretezés

- Pl: ha egy 5-ös bájtszám a második keretében 7-té válik egyetlen bithiba eredményeképpen, a célállomás kiesik a szinkronból,
  - képtelen lesz megtalálni a következő keret elejét.
- Még ha az ellenőrző összeg hibás is, és a címzett tudja, hogy a keret rossz, akkor sincs mód arra, hogy megmondjuk, hol kezdődik a következő keret.
- Szintén nem segít a keret újraküldésének kérése
  - a címzett nem tudja, hogy hány bájtot kell átlépnie ahhoz, hogy az újraküldött keret elejéhez érjen.
  - Az előzőek miatt a bájtszámlálásos módszert ma már ritkán használják.

Keretezés

- Gyakori megoldás a **speciális határoló karakterek alkalmazása**
- A keret elejét és végét egyedi, előre meghatározott jelsorozatok jelölik
- Régebben a keretek elejét és végét különböző bájtok jelezték,
- Az elmúlt években a legtöbb protokoll már ugyanazt a bájtot használja erre a célra.
- Ezt **jelzőbájt**nak (**flag byte**) nevezik.
- Két egymást követő jelzőbájt közül az egyik a keret végét, a másik a következő keret elejét jelzi.
- Ha a vevő bármikor kiesik a szinkronból, akkor csak a jelzőbájtot kell megkeresnie ahhoz, hogy megtalálja az éppen futó keret végét.

Keretezés



(a) Egy jelzőbájtokkal határolt keret.
(b) Négy bájtsorozat bájtbeszúrás előtt és után

Keretezés

- A módszernél komoly gond jelentkezik, amikor bináris adatokat, például képeket vagy zenét kell átvinni.
- Könnyen előfordulhat, hogy a jelzőbájt bitmintája megjelenik az adatok között is
  - ez pedig általában belezavar a keretezésbe.
- **A gond orvoslásának egyik módja:**
  - a küldő fél adatkapcsolati rétege egy különleges **kivételbájt**ot (**escape byte, ESC**) helyez minden olyan jelzőbájt elé, amely „véletlenül” került az adatmezőbe,
  - így minden jelzőbájt megkülönböztethető az adatokban előforduló azonos értékű bájtoktól az előtte álló kivételbájt (ESC) megléte, vagy hiánya alapján.
  - A vevő oldal adatkapcsolati rétege eltávolítja ezt a kivétel bájtot, mielőtt az adatokat a hálózati rétegnek továbbítaná.
  - Ezt a módszert **bájtbeszúrás**nak (**byte stuffing**) nevezik.

Keretezés

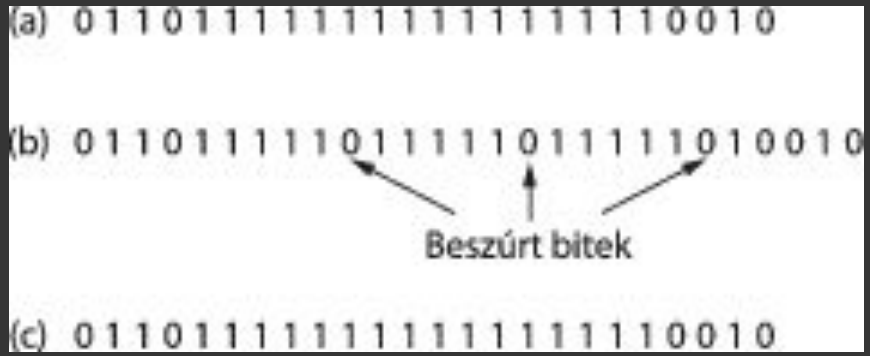
- Mi történik abban az esetben, amikor a felhasználó adatai egy kivételbájtot tartalmaznak?
  - ezt is megjelölik egy kivételbájttal.
  - Így minden egyedüli kivételbájt egy kivételbájt-sorozat (escape sequence) része
    - ahol minden kettős karakter azt jelenti, hogy egy kivételbájt volt a felhasználói adatok között.
  - A beszúrt bájtok eltávolítása után leszállított bájtsorozat minden esetben pontosan megegyezik az eredeti bájtsorozattal.
  - A kerethatárok a kivételbájtok eltávolítása nélkül továbbra is megtalálhatók két, egymás melletti jelzőbájt keresésével.

Keretezés

- **HDLC (Highlevel Data Link Control)-hoz kifejlesztett megoldás:**
  - tipikusan 8 bites bájtokat használ
  - lehetővé teszi, hogy tetszőleges számú bit legyen egy keretben, és az alkalmazott karakterkódok is tetszőleges számú bitet tartalmazzanak
- **Működése:**
  - minden keret egy speciális, jelző- (flag) bájtnak nevezett bitmintával kezdődik.
  - Ez a **01111110** (hexadecimálisan 0x7E).
  - Amikor az adó adatkapcsolati rétege öt egymást követő 1-es bitet talál az adatok között, automatikusan beszúr egy 0-t a kimenő bitfolyamba.
  - Ez a bitbeszúrás (bit stuffing) analóg a bájtbeszúrással, amelyben egy ESC-t szúrtunk be a kimenő bájtfolyamba az adatok között levő jelzőbájt elé.
  - Mivel a módszer a legkevesebb átmenettel jár, ezért a fizikai réteg könnyebben szinkronban marad
  - Éppen ezért alkalmazzák e módszert az USB- (Universal Serial Bus – univerzális soros sin) szabványban.0

Keretezés

- Amikor a vevő öt egymást követő 1-es bitet talál, melyet egy 0-s követ, automatikusan törli a 0-s bitet.
- Ahogyan a karakterbeszúrás teljesen átlátszó a hálózati réteg számára mindkét számítógépben, a bitbeszúrás is az.
- Ha a felhasználói adat tartalmazza a jelzőbájt bitmintáját (01111110), ez 011111010-ként továbbítódik, de a vevő memóriájában már 01111110 jelenik meg.



(a) Az eredeti adat.
(b) Az átviteli vonalon megjelenő adat.
(c) A vevő memóriájában megjelenő, a beszúrt bitek törlése utáni adat

Keretezés

- A bitbeszúrásos módszer segítségével egyértelműen felismerhetők a kerethatárok:
  - ha a vevő szem elől téveszti a határokat, semmi mást nem kell tennie, mint a bemeneti bitfolyamban a jelző mintát keresnie,
  - hiszen a minta csak kerethatárokon fordulhat elő, az adatok között sohasem.
- Mind a bájtbeszúrásnál, mind a bitbeszúrásnál megjelenik a mellékhatás:
  - a keret hossza a szállított adatok tartalmától függ.
  - Pl. ha az adatmező nem tartalmaz jelzőbájtot, 100 bájtnyi adat nagyjából egy 100 bájtos keretben átvihető,
  - míg egy kizárólag jelzőbájtokból álló adatmező a kivételbájtokkal körülbelül 200 bájtos keretben továbbítható.
  - Ebből a szempontból a bitbeszúrás kedvezőbb, mert a keretméret növekedése nagyjából 12,5%,
    - hiszen minden bájthoz egy bitet adtunk hozzá.

Keretezés

- Sok adatkapcsolati protokoll a nagyobb biztonság érdekében a módszerek valamely kombinációját alkalmazza.
- **Egy gyakori megoldásra példa:**
  - az Ethernet és a 802.11 protokollokban, ahol a keret egy jól definiált előtaggal (**preamble**) kezdődik.
  - Az előtag megfelelően hosszú is lehet (802.11 esetén 72 bit), hogy segítse a vevőt felkészülni az adatok fogadására.
  - Az előtagot egy hosszkód- (azaz bájtszám-) mező követi a fejlécben, melyet a keret végének meghatározására használnak.

Forgalomszabályozás...

Forgalomszabályozás

- **Fontos tervezési kérdés:** mit tegyünk azzal az állomással, amelyik rendszeresen gyorsabban akarja adni a kereteket, mint ahogy a vevő azokat fogadni tudná?
- Könnyen előállhat akkor, ha az adó egy gyors (vagy kevéssé terhelt) számítógép, a vevő pedig egy lassú (vagy erősen leterhelt) gép.
- **Példa:**
  - egy okostelefon weboldalt kér le egy gyors szervertől, amely teljes sebességgel kiszolgálja a kérést,
  - és a telefont teljesen elárasztja adatokkal.
- Még ha az átvitel hibamentes is, a vevő egy bizonyos ponttól kezdve nem lesz képes kezelni a folyton érkező kereteket, és néhányat el fog veszíteni.

Forgalomszabályozás

- Az adatkapcsolati réteg egyik alapvető feladata a **forgalomszabályozás** (flow control),
- Biztosítja, hogy az adatátvitel során a küldő és a fogadó fél közötti adatáramlás kiegyensúlyozott legyen
- **A gyakorlatban két megközelítés terjedt el:**
    - Visszacsatolás-alapú forgalomszabályozás (feedback-based flow control).
    - Sebesség alapú forgalomszabályozás (rate-based flow control).

Visszacsatolás-alapú forgalomszabályozás

- A visszacsatolás-alapú forgalomszabályozási módszerek léteznek mind az adatkapcsolati, mind a felsőbb rétegekben.
- A visszacsatolás-alapú forgalomszabályozás manapság sokkal gyakoribb
    - mivel az adatkapcsolati réteget megvalósító hardverek elég gyorsak ahhoz, hogy ne veszítsenek kereteket.
- Pl.: az adatkapcsolati réteg hardver megvalósításáról, a hálózati csatoló kártyáról (Network Interface Card, NIC) néha azt mondják, hogy „vonali sebességgel fut”,
    - mivel a beérkező kereteket olyan gyorsan tudja feldolgozni, ahogy azok a vonalról a vevőbe érkeznek
    - Bármilyen túltöltés ekkor már nem adatkapcsolati probléma, így a túltöltéssel felsőbb rétegeknek kell foglalkozniuk

Visszacsatolás-alapú forgalomszabályozás

- Különféle forgalomszabályozási elgondolások ismertek, de legtöbbjük ugyanazt az alapelvet alkalmazza.
- A protokoll jól definiált szabályokat tartalmaz arra vonatkozóan, hogy a következő keretet mikor küldheti el az adóállomás.
- Ezek a szabályok általában megtiltják egy keret elküldését, amíg a vevő – akár implicit, akár explicit módon – engedélyt nem ad rá.
- Például, amikor egy kapcsolat felépül, a vevő mondhatja a következőt:
    - „Most küldhetsz nekem n keretet, de ha ezeket elküldted, ne küldj többet addig, amíg nem szólok.”

Visszacsatolás-alapú forgalomszabályozás

- **Stop-and-wait eljárás:**
  - a küldő egyetlen keretet továbbít, majd megáll, és megvárja a vevő visszajelzését.
  - Csak azután küldi el a következő keretet, ha megkapta a pozitív nyugtát.
  - Ez a módszer egyszerű és megbízható, azonban nem túl hatékony,
    - különösen nagy késleltetésű hálózatok esetén, mivel a küldő sok időt tölt várakozással.
- **Csúszó ablakos (sliding window):**
  - a hatékonyság növelésére alkalmazzák. A küldő nem egyetlen keretet küld, hanem egyszerre több keretet is útnak indíthat
    - anélkül, hogy azonnali visszaigazolásra várna.
  - A „csúszó ablak” egy olyan logikai tartományt jelöl, amely meghatározza, hogy a küldő hány keretet küldhet el nyugta nélkül.
  - Ahogy a vevő visszaigazolja a kereteket, az ablak előrecsúszik, lehetővé téve újabb keretek küldését.

Visszacsatolás-alapú forgalomszabályozás

- A csúszó ablakos módszer jelentősen javítja a hálózat kihasználtságát
- Csökkenti az üresjáratokat, és lehetővé teszi az adatátvitel folyamatosabb működését.
- Ugyanakkor bonyolultabb megvalósítást igényel, mivel a küldőnek nyilván kell tartania a már elküldött, de még nem nyugtázott kereteket.
- A forgalomszabályozás szorosan összefügg a hibakezeléssel is.
- Ha egy keret elveszik vagy megsérül, a vevő nem küld visszaigazolást, vagy negatív nyugtát (NAK) küld.
- Ilyenkor a küldő újraküldi a hiányzó keretet.
- A csúszó ablakos rendszerekben különböző stratégiák léteznek az újraküldésre,
    - Pl. **Go-Back-N** és a **Selective Repeat** módszerek

Hibakezelés és javítás...

Hibakezelés

- Az adatkapcsolati réteg egyik legfontosabb feladata a **hibakezelés** (**error control**)
  - biztosítja, hogy az adatátvitel során fellépő hibák felismerhetők és – bizonyos esetekben – javíthatók legyenek.
    - A fizikai átviteli közeg nem tökéletes: az elektromos zaj, interferencia vagy egyéb zavarok következtében a továbbított bitek megváltozhatnak
    - az adatok sérüléséhez vezethet.
- **A hibakezelés célja kettős:**
  - egyrészt a hibák detektálása, másrészt azok kezelése,
  - történhet **újraküldés**sel vagy speciális esetekben **hibajavítás**sal.
  - Az adatkapcsolati réteg általában elsősorban hibadetektálást végez, míg a hibák javítása gyakran újraküldés útján történik.

Hibakezelés

- A hálózattervezők két alapvető stratégiát dolgoztak ki a hibák kezelésére.
- Mindkét módszer redundáns információkat csatol az adatokhoz.
- **1. Módszer: (error-correcting code)**
  - minden elküldött adatblokkhoz annyi redundáns információt mellékelünk
    - amennyiből a vevő ki tudja következtetni, hogy mik voltak az eredetileg elküldött adatok.
- **2. Módszer: (error-detecting code)**
  - annyi redundanciát iktatunk az adatok közé, amennyi a vevőnek lehetővé teszi, hogy a hiba tényét kikövetkeztesse.
  - A vevő ebben a megoldásban nem tudja, milyen hiba történt, ezért újraküldést kér.

- A hibajavító kódok használatát gyakran előre irányuló hibajavításnak (**Forward Error Correction, FEC**) is nevezik.

Hibakezelés

- Mindkét módszernek megvan a saját alkalmazási területe.
- A fényvezető szálakon és más, megbízható csatornákon olcsóbb, ha hibajelző kódot használunk
  - és egyszerűen újraküldjük a ritkán előforduló hibás blokkokat.
- Ezzel szemben, a vezeték nélküli összeköttetéseken és más olyan csatornákon, amelyek sokat hibáznak:
  - jobb, ha minden blokkba annyi redundanciát építünk, amennyiből a vevő már ki tudja találni, hogy mi volt az eredeti blokk.
  - Az újraküldésre ebben az esetben nem jó támaszkodni, mivel az maga is hibás lehet.

Hibakezelés

- A hibajavító és hibajelző kódok elmélete az alkalmazott matematika tárgyterülete (Galois-testek, ritka mátrixok).
- **Fontosabb hibajavító kódok:**
  - Hamming-kódok
  - Bináris konvolúciós kódok,
  - Reed–Solomon-kódok,
  - Alacsony sűrűségű paritásellenőrző kódok

Hibajelző kódok...

A paritásbit (Parity Bit)

- A paritásbit a legegyszerűbb hibadetektáló módszerek egyik az adatátvitel során fellépő hibák felismerésére.
- **Az eljárás lényege:**
  - az eredeti adathoz egyetlen extra bitet adunk hozzá, amely információt hordoz az adatsorban található 1-es bitek számáról.
- A paritásbit célja annak ellenőrzése, hogy az átvitel során megváltozott-e valamelyik bit értéke.
- Bár ez a módszer nem képes minden hibát felismerni, egyszerűsége miatt alapvető jelentőségű,
  - jól szemlélteti a hibadetektálás működésének alapelveit.
- A paritásbit alkalmazásának két fő típusa létezik:
  - **páros paritás** és **páratlan paritás**.

A paritásbit (Parity Bit)

- **Páros paritás:** az a cél, hogy az adatsorban található 1-es bitek száma páros legyen.
  - Ennek érdekében a küldő megszámolja az 1-es biteket,
  - Ha azok száma páratlan, akkor a paritásbit értékét 1-re állítja, így az összes 1-esek száma páros lesz.
  - Ha az 1-esek száma már eleve páros, akkor a paritásbit értéke 0 lesz.

- **Páratlan paritás:**
  - az adó arra törekszik, hogy az 1-es bitek száma páratlan legyen.
  - Ha az adatsorban páros számú 1-es található, akkor a paritásbitet 1-re állítja, ha pedig már eleve páratlan, akkor 0 lesz a paritásbit.

A paritásbit (Parity Bit) - PÉLDA

- Tegyük fel, hogy az átvitelre kerülő adat a következő bitminta:

1011001

- Ebben az esetben az 1-esek száma négy, ami páros. Ha páros paritást alkalmazunk, akkor a paritásbit értéke 0 lesz, így az elküldött adatsor:

10110010

- Ha azonban páratlan paritást használunk, akkor a paritásbit értéke 1 lesz, hogy az 1-esek száma páratlanra változzon:

10110011

- A vevő az adatok fogadása után:
  - megszámolja az 1-es biteket, és ellenőrzi, hogy azok száma megfelel-e az előre meghatározott paritási szabálynak.
  - Ha eltérést tapasztal, akkor hibát észlel, és feltételezi, hogy az adatátvitel során legalább egy bit megváltozott.

A paritásbit (Parity Bit) - PÉLDA

- A paritásbit csak korlátozott hibadetektálási képességgel rendelkezik.
- Képes felismerni az olyan hibákat, amelyek során a bitek száma páratlan módon változik meg, például egyetlen bit hibáját.
- Ugyanakkor nem képes észlelni azokat az eseteket, amikor páros számú bit változik meg,
  - mivel ilyenkor a paritás nem változik meg, és a hiba rejtve marad.
- **További korlátozás:**
  - a paritásbit nem ad információt arról, hogy pontosan melyik bit hibásodott meg,
  - így hibajavításra önmagában nem alkalmas, csak hibadetektálásra.

A checksum (ellenőrző összeg)

- A **checksum**, vagyis ellenőrző összeg egy olyan hibadetektáló eljárás, amelyet az adatátvitel során fellépő hibák felismerésére alkalmaznak.
- A paritásbitnél fejlettebb módszerről van szó
  - nagyobb megbízhatóságot biztosít, ugyanakkor viszonylag egyszerűen megvalósítható.
- **A checksum alapelve:**
  - a küldő fél az elküldendő adatot kisebb egységekre bontja, majd ezekből egy matematikai művelet segítségével kiszámít egy összegértéket.
  - Ezt az értéket – az úgynevezett checksumot – az adat mellé csatolja, és együtt továbbítja a vevőnek.
  - A fogadó fél ugyanazt a számítást elvégzi a kapott adatokon
  - Majd összehasonlítja az eredményt a kapott checksum értékkel.
  - Ha a két érték megegyezik, az adat valószínűleg hibátlanul érkezett meg; ha eltérés van, akkor hibát észlel

A checksum (ellenőrző összeg)

- A checksum kiszámításának többféle módja létezik.
- Egyik leggyakoribb megoldás az úgynevezett **egyes komplemensű összeadás** (one’s complement addition) alkalmazása.
- **Az eljárás:**
  - az adatot azonos hosszúságú blokkokra (például 8 vagy 16 bites egységekre) bontják, majd ezeket összeadják.
  - Ha az összeadás során túlcsordulás keletkezik, a keletkező **átvitel** (**carry**) visszakerül az összeg alsó bitjeihez.
  - A végén az így kapott összeg bitenkénti negáltját veszik, és ez lesz a checksum.

Checksum - PÉLDA

- A checksum kiszámításának többféle módja létezik.
- Tegyük fel, hogy az adatot 8 bites blokkokra bontjuk, és a következő két blokkot kell elküldeni:

```
10101010
11001100
```

- Ezeket összeadva:

```
 10101010
+11001100
----------
 01110110  (carry keletkezett)
```

Checksum - PÉLDA

- A túlcsorduló bitet visszaadjuk az eredményhez:

  01110110
  +00000001

  01110111

- Ezután az eredmény bitenkénti invertálásával kapjuk meg a checksumot:

  10001000

A checksum (ellenőrző összeg)

- A küldő az eredeti adat mellé elküldi ezt az értéket is.
- A vevő az összes beérkező blokkot – beleértve a checksumot is – összeadja.
- Ha az eredmény minden bitje 1 (azaz nulla lesz az invertált formában), akkor az adatátvitel hibátlannak tekinthető.
- **A checksum előnye:**
  - képes többféle hibatípus felismerésére, például több bit megváltozását is detektálhatja,
    - ami a paritásbit esetében nem mindig lehetséges.
- Ugyanakkor nem garantálja minden hiba felismerését, különösen bizonyos speciális mintázatok esetén előfordulhat, hogy a hiba rejtve marad.

Checksum (ellenőrző összeg)

- A módszer hatékonysága függ az alkalmazott blokk méretétől és az alkalmazott algoritmustól.
- Nagyobb blokkméret esetén általában jobb hibadetektálási képesség érhető el, azonban ez növeli a számítási igényt is.
- A checksum széles körben alkalmazott eljárás különböző hálózati protokollokban.
- Pl. az IP és a TCP protokollok is használnak ellenőrző összeget az adatok integritásának ellenőrzésére.
- Bár a modern rendszerekben gyakran erősebb hibadetektáló módszereket, például CRC-t alkalmaznak, a checksum továbbra is fontos szerepet játszik a hálózati kommunikációban.

Ciklikus redundanciaellenőrzés (CRC)

- A ciklikus redundanciaellenőrzés, röviden **CRC** (**Cyclic Redundancy Check**), az egyik legelterjedtebb és leghatékonyabb hibadetektáló eljárás az adatátvitelben
- Az adatkapcsolati rétegben különösen fontos szerepet tölt be
  - mivel képes a legtöbb gyakori hibatípus megbízható felismerésére,
  - beleértve az egybites, több bites és úgynevezett csomós (burst) hibákat is
- A CRC működésének alapja a **matematikai polinomok** alkalmazása
- Az adatbit-sorozatot egy bináris polinomként értelmezzük, majd ezt a polinomot elosztjuk egy előre meghatározott generátorpolinommal
- Az osztás során keletkező maradékot használjuk hibadetektáló kódként, amelyet az eredeti adathoz csatolva továbbítunk

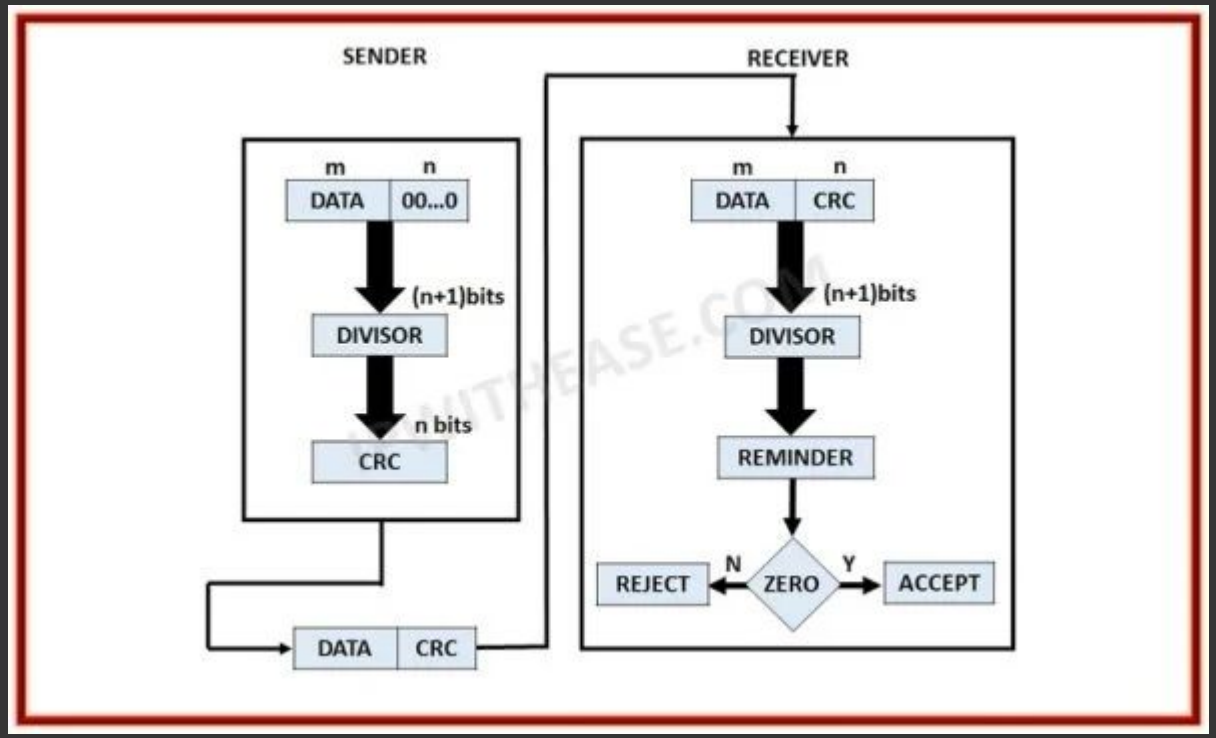
Ciklikus redundanciaellenőrzés (CRC)

- A CRC számításának első lépése az, hogy az eredeti adat végéhez nullákat fűzünk.
- A hozzáfűzött nullák száma megegyezik a generátorpolinom fokszámával.
- Ezután az így kapott bitmintát elosztjuk a generátorpolinommal bináris osztás segítségével.
- Fontos, hogy ez az osztás nem hagyományos aritmetikai osztás, hanem XOR műveleteken alapuló műveletsor.
- Az osztás során kapott maradék lesz a CRC érték, amelyet az eredeti adat végéhez illesztünk.
- Az így létrejövő bitfolyam kerül átvitelre.

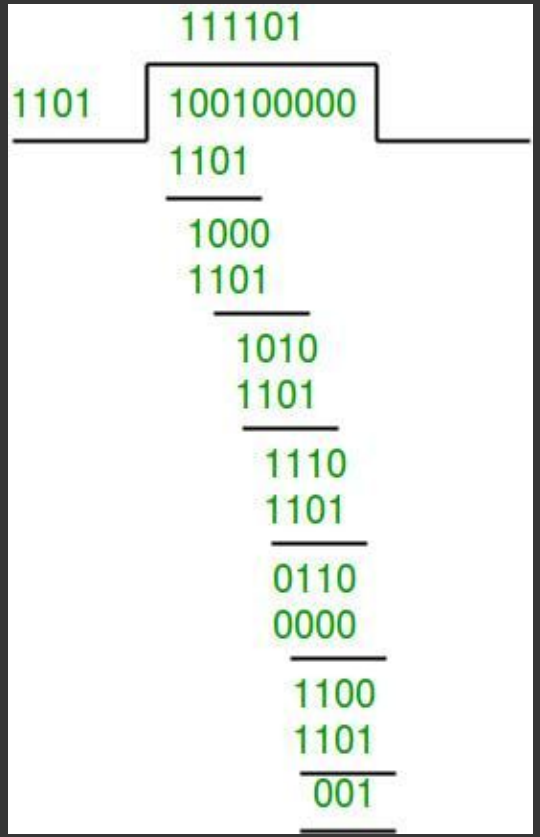
Ciklikus redundanciaellenőrzés (CRC)

- A vevő ugyanazzal a generátorpolinommal végzi el az osztást a teljes beérkezett adaton (az eredeti adaton és a CRC-n együtt).
- Ha az osztás maradéka nulla, akkor az adatot hibátlannak tekinti; ha nem nulla, akkor hibát észlel.
- A CRC egyik legnagyobb előnye, hogy rendkívül hatékony a különböző típusú hibák felismerésében.
- Különösen jól kezeli az úgynevezett burst hibákat, amikor egymást követő bitek sorozata sérül.
- A megfelelő generátorpolinom kiválasztásával biztosítható, hogy a legtöbb gyakori hibatípus nagy valószínűséggel felismerhető legyen.

Ciklikus redundanciaellenőrzés (CRC)



Ciklikus redundanciaellenőrzés (CRC)

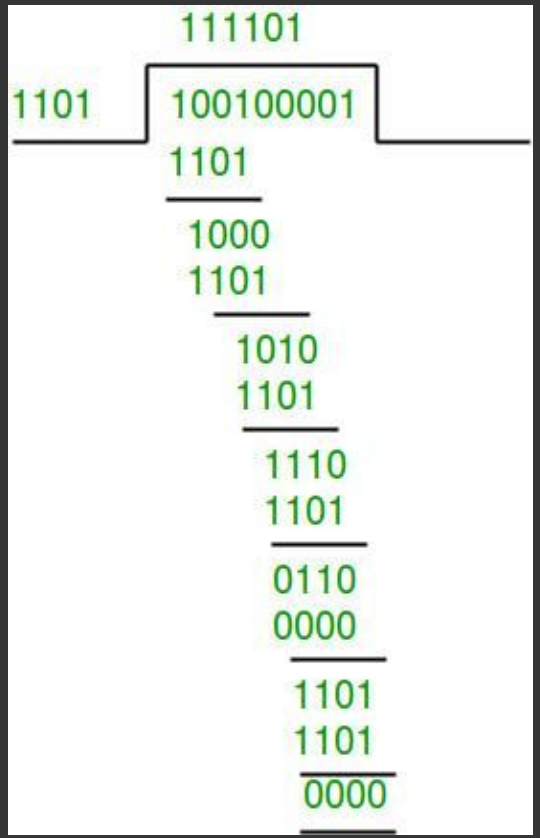


Sender Side

- Data = 100100
- Generator Polynomial (Key) = x3 + x2 + 1 (1101)

The remainder is 001. Thus the data sent is 100100001.

Ciklikus redundanciaellenőrzés (CRC)



Receiver Side

- Code word received at the receiver side 100100001
- The remainder is 0, hence the data received has no errors.
- CRC Implementation - O(n) Time and O(n) Space

Ciklikus redundanciaellenőrzés (CRC)

- A CRC számítás elsőre bonyolultnak tűnhet, a gyakorlatban hatékonyan implementálható hardveresen és szoftveresen is
  - Pl. shift regiszterek segítségével.
  - Ez lehetővé teszi, hogy nagy sebességű hálózatokban is alkalmazható legyen.
- **Fontos megjegyezni:**
  - a CRC nem képes a hibák javítására, csupán azok felismerésére
  - Hibadetektálás esetén az adatkapcsolati réteg vagy eldobja a hibás keretet, vagy felsőbb mechanizmusok (például ARQ eljárások) segítségével újraküldést kezdeményez.
- A CRC-t széles körben alkalmazzák különböző hálózati technológiákban.
- Például az Ethernet keretek végén található ellenőrző mező (**FCS – Frame Check Sequence**) is CRC-n alapul.
- Ez biztosítja, hogy a hálózaton átvitt keretek nagy megbízhatósággal ellenőrizhetők legyenek.

MAC cím...

MAC-cím

- Az adatkapcsolati réteg egyik alapvető eleme a fizikai címzés, amelyet a **MAC**-címek (**Media Access Control**) valósítanak meg.
- A MAC-cím egy egyedi azonosító, amely minden hálózati eszköz hálózati interfészéhez (például hálózati kártyához) tartozik,
  - lehetővé teszi az eszközök azonosítását a helyi hálózaton belül.
- A MAC-címek biztosítják, hogy az adatkapcsolati rétegben az adatok a megfelelő fizikai eszközhöz jussanak el.
- Míg a magasabb rétegek logikai címeket, például IP-címeket használnak,
  - addig az adatkapcsolati réteg a tényleges hardverhez kötött címekkel dolgozik.

MAC-cím

- A MAC-cím általában **48 bites** hosszúságú, és hexadecimális formában szokás megjeleníteni. Egy tipikus MAC-cím a következő formában néz ki:

  00:1A:2B:3C:4D:5E

- A cím hat darab, egyenként 8 bites (1 bájtos) részből áll, amelyeket általában kettőspontokkal vagy kötőjelekkel választanak el egymástól.
- A MAC-cím két fő részből épül fel:
  - Az első 24 bit az úgynevezett gyártói azonosító (OUI – **Organizationally Unique Identifier**),
    - megmutatja, hogy melyik gyártó készítette az adott hálózati eszközt
  - A második 24 bit az eszköz egyedi azonosítója
    - a gyártó rendel hozzá az adott interfészhez.
  - Ez a felépítés biztosítja, hogy minden MAC-cím globálisan egyedi legyen.

MAC-cím

- A MAC-címeket általában a gyártás során égetik be a hálózati eszközbe, ezért gyakran „fizikai címnek” is nevezik őket
- Bár bizonyos esetekben szoftveresen módosíthatók (MAC spoofing), alapvetően állandó azonosítóként működnek.
- Az adatkapcsolati rétegben az adatok keretek (frame-ek) formájában kerülnek továbbításra, és ezek a keretek tartalmazzák a forrás és a cél MAC-címet.
- Amikor egy eszköz adatot küld a hálózaton, a keret fejlécébe beírja a saját MAC-címét forrásként, valamint a célállomás MAC-címét célként.
- A hálózat többi eszköze megvizsgálja a célcímet, és csak az a készülék dolgozza fel a keretet, amelynek a MAC-címe megegyezik a célcímmel.

MAC-cím

- Fontos szerepet játszanak a MAC-címek a helyi hálózatok működésében
  - különösen az olyan eszközök esetében, mint a **switch**.
- Bizonyos esetekben speciális MAC-címek is használatosak.
  - Ilyen például a broadcast cím, amelynek értéke:

FF:FF:FF:FF:FF:FF

- Ha egy keret célcíme ez az érték, akkor azt a hálózat minden eszköze megkapja.
- Emellett léteznek multicast címek is, amelyek egy eszközcsoport számára teszik lehetővé az adatküldést.

Switch-ek...

Switch működése

- A switch egy olyan hálózati eszköz, amely az adatkapcsolati rétegben működik
- **Feladata:** a helyi hálózaton belüli eszközök közötti adatforgalom hatékony továbbítása.
- A switch működése szorosan kapcsolódik a MAC-címekhez
  - ezek alapján dönti el, hogy egy beérkező keretet melyik irányba kell továbbítani.
- A switch több porttal rendelkező eszköz, amelyhez különböző hálózati berendezések csatlakoznak
  - például számítógépek, nyomtatók vagy más hálózati eszközök
- Amikor egy keret érkezik a switch egyik portjára:
  - az eszköz megvizsgálja a keret fejlécében található forrás MAC-címet,
  - eltárolja azt egy belső táblában, amelyet **MAC-címtábl**ának nevezünk
  - Ez a tábla azt tartalmazza, hogy melyik MAC-cím melyik porton érhető el.

Switch működése

- Amikor a switch egy újabb keretet kap, megnézi annak cél MAC-címét
- Ha a célcím szerepel a MAC-címtáblában:
  - akkor a switch pontosan arra a portra továbbítja a keretet, amelyhez a cél eszköz tartozik.
- Ha a célcím nem ismert:
  - akkor a switch úgynevezett **„flooding”** módszert alkalmaz: a keretet minden porton kiküldi, kivéve azt, amelyen beérkezett.
    - Így a megfelelő eszköz biztosan megkapja az adatot
- Ez a működés jelentős előrelépést jelent a régebbi hub eszközökhöz képest:
  - A hub minden beérkező adatot minden porton továbbított, függetlenül a célállomástól.
  - A switch ezzel szemben képes célzott adatátvitelre, ami csökkenti a hálózati forgalmat és növeli a hatékonyságot.

Switch működése

- A hub működéséből adódóan minden csatlakoztatott eszköz egyetlen ütközési tartományt (collision domain) alkot,
- Switch esetében minden port külön ütközési tartományt képez.
- Ez azt jelenti, hogy a switch használata esetén az eszközök párhuzamosan, ütközések nélkül kommunikálhatnak,
    - nagyobb teljesítményt és hatékonyabb hálózati működést eredményez.
- A switch tehát kulcsszerepet játszik a helyi hálózatok működésében,
    - lehetővé teszi több eszköz egyidejű, ütközésmentes kommunikációját.
- A MAC-címek alapján történő intelligens továbbítás révén optimalizálja az adatforgalmat, és hozzájárul a hálózat stabil és gyors működéséhez

Switch

Köszönöm a figyelmet!