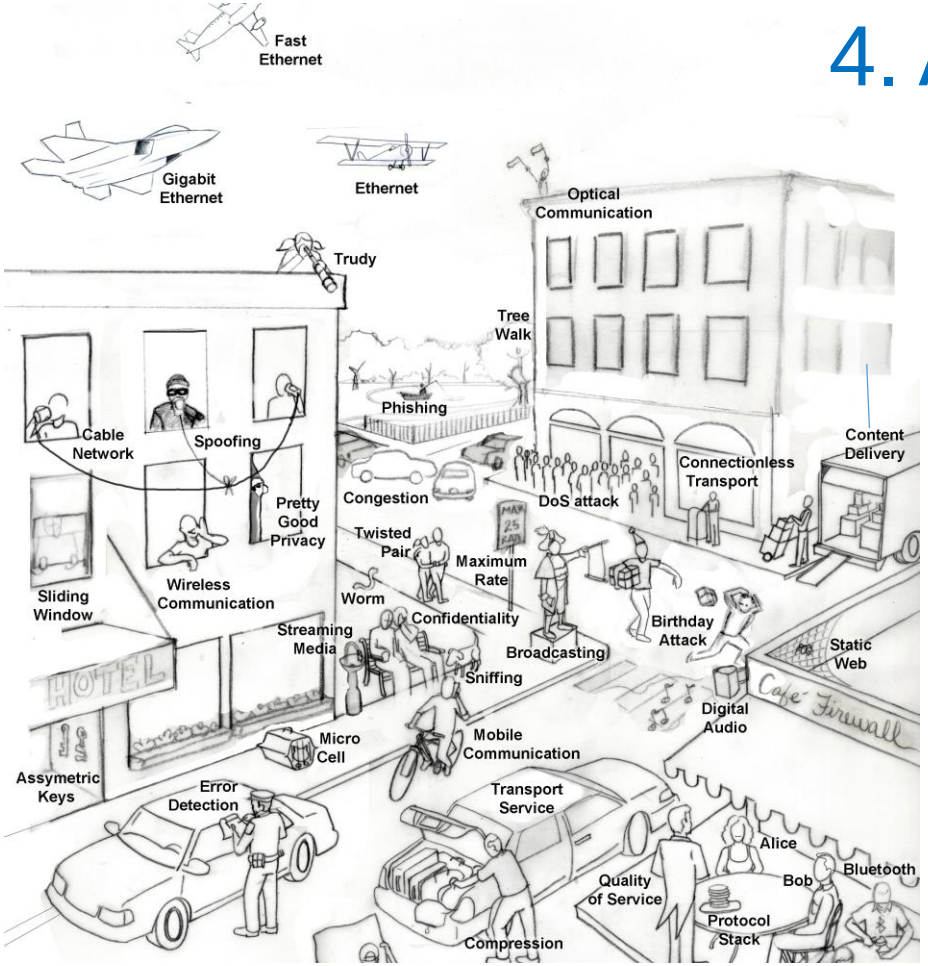
Számítógép-hálózatok

4. Adatkapcsolati réteg I.



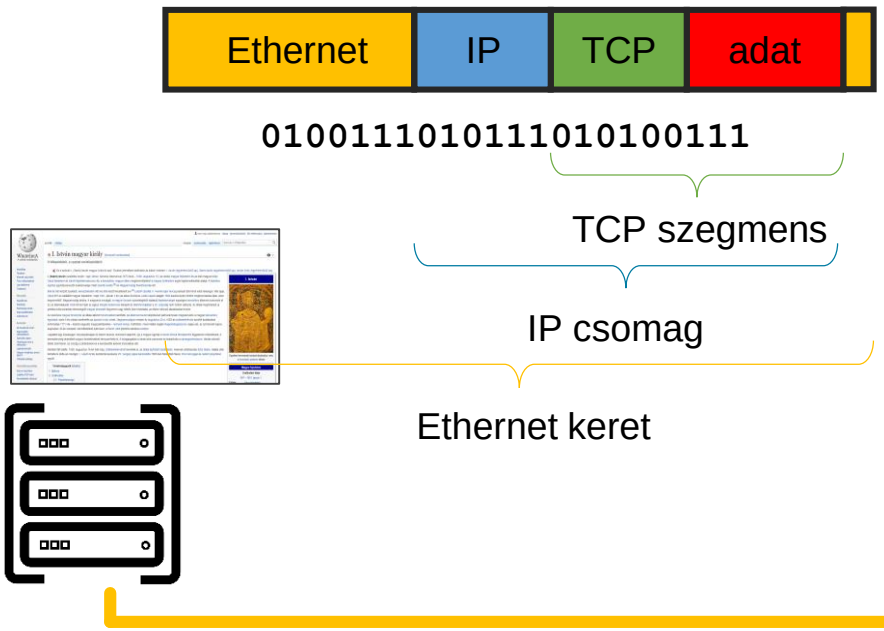
Irodalom: Tanenbaum, Wetherall: Számítógép-hálózatok

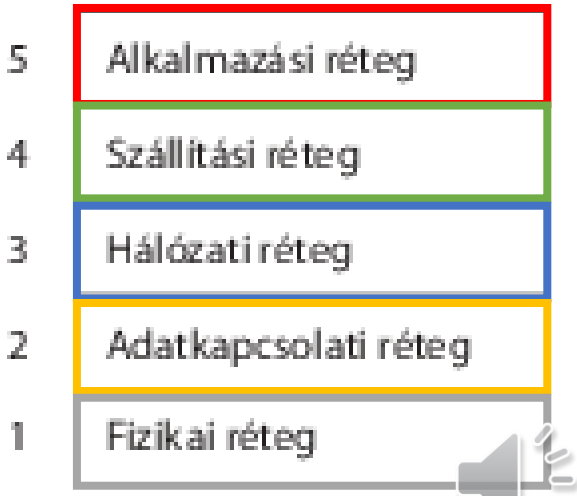
Simon Gyula, Óbudai Egyetem

Tartalom

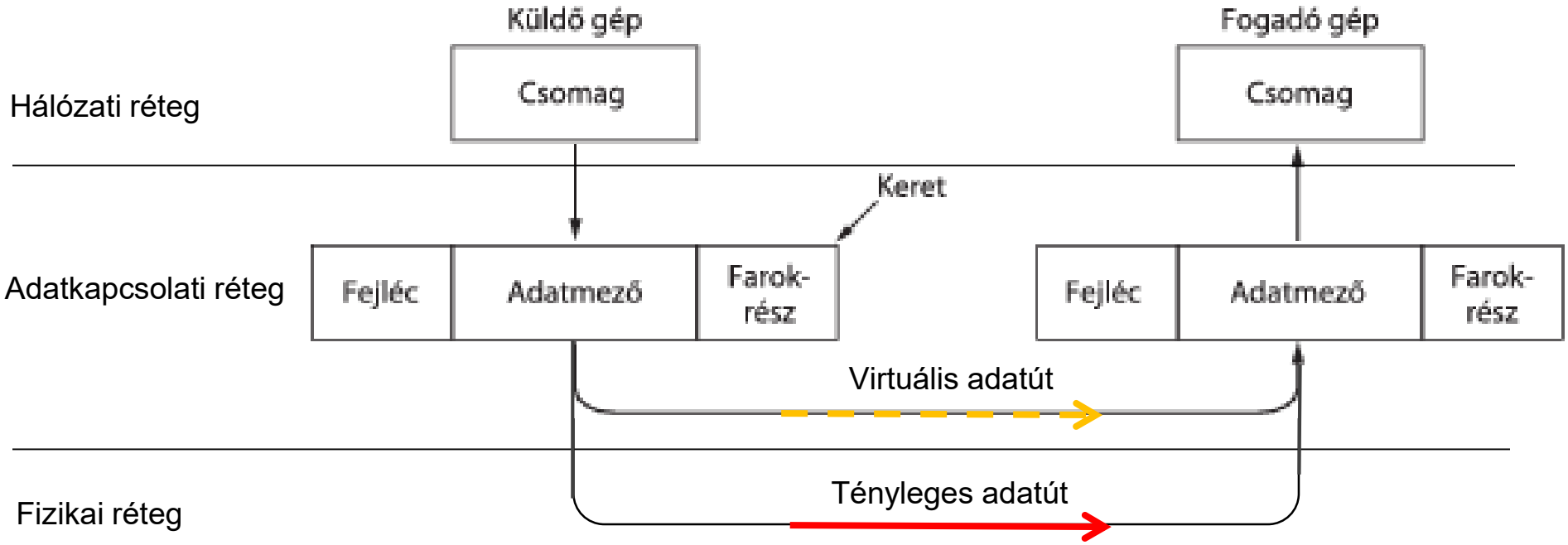
- Ismétlés
  - az adatkapcsolati réteg helye és szerepe
- Az adatkapcsolati réteg feladatai
  - Keretezés
  - Hibakezelés
    - Hibadetektáló kódok
    - Hibajavító kódok
    - Újraküldés
- Példák

Ismétlés: réteges szerkezet





Ismétlés: az adatkapcsolati réteg helye



Csomag és keret kapcsolata.
Virtuális és fizikai kapcsolat

A hálózati rétegnek nyújtott szolgáltatások

- Három lehetséges szolgáltatás
  - Nem nyugtázott kapcsolat nélküli szolgáltatás
    - A keretet kapcsolat kiépítése nélkül küldjük
    - Nincs hibajavítás ebben a rétegben (eseteges hibákat magasabb rétegekben javítjuk)
    - Példa: Ethernet
  - Nyugtázott kapcsolat nélküli szolgáltatás
    - A keretet szükség szerint újraküldjük
    - Példa: 802.11
  - Nyugtázott kapcsolat-alapú szolgáltatás
    - Először kapcsolat kiépítése
    - Ritka

Adatkapcsolati réteg feladatai

- Keretezés
  - Hol vannak az adatelemek határai?
    - Hol kezdődik a bájt?
    - Hol kezdődik egy csomag?
- Hibakezelés
  - Hibadetekció
  - Hibajavítás
  - Szükség szerint keretek újraküldése
    - Hiba detektálása esetén
- Címzés
  - A címzett és a feladó állomás azonosítása (később)
- Közeghozzáférés
  - Egy csatorna, több eszköz
  - Külön alréteg (később…)

Keretezés (1)

- A fizikai réteg bitsorozatot továbbít
  - Nincs tudomása a kerethatárokról…
  - Hol kezdődik a keret?

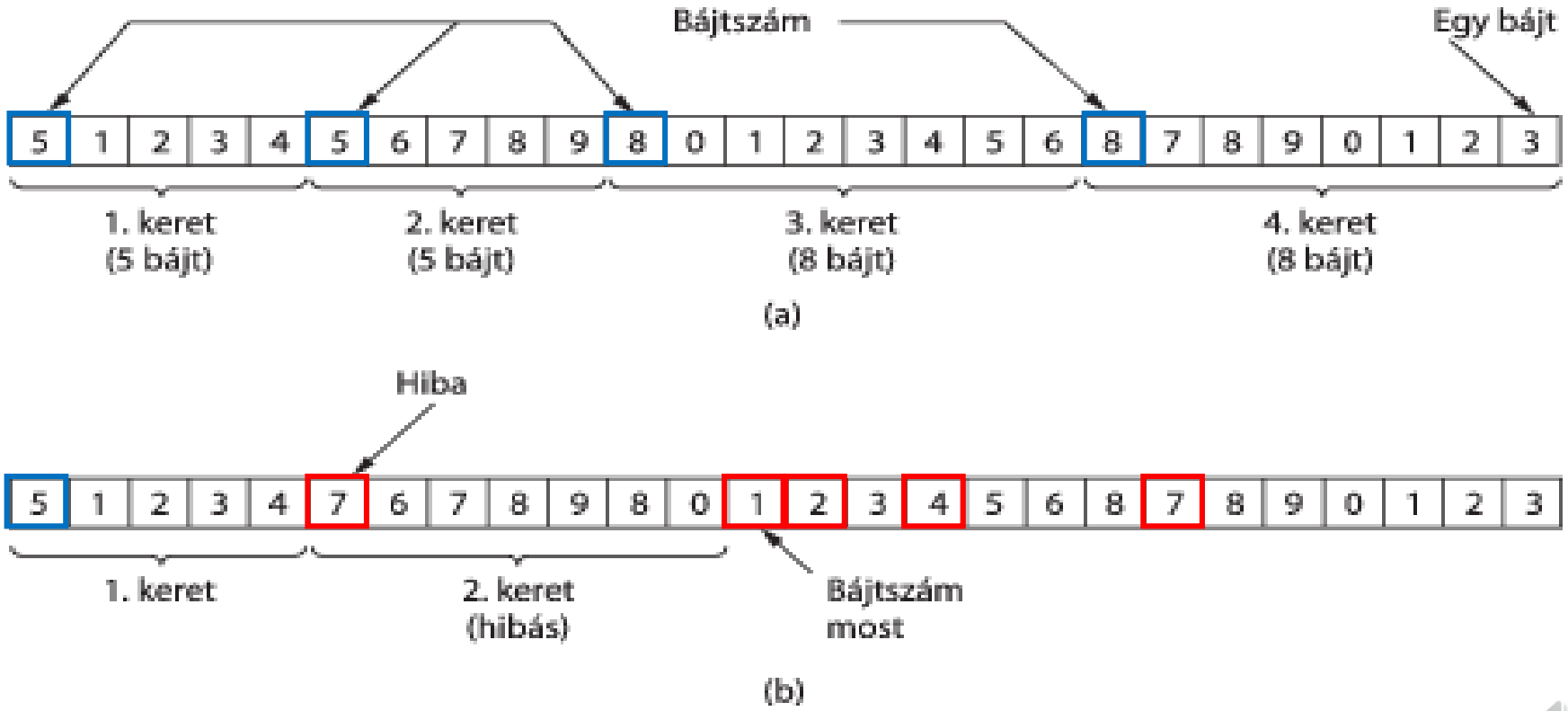
… 00 | 01100001 01101100 01101101 01100001 | 00.. → alma

… 000 | 11000010 11011000 11011010 11000010 | 0.. → ÂØÚÂ

- Keretező megoldások
  - Bájtszámlálás
  - Kezdő- és végkarakterek használata bájtbeszúrással
  - Kezdő- és végjelek használata bitbeszúrással
  - Fizikai rétegbeli kódolás megsértése

Keretezés (2)
Bájtszámlálás

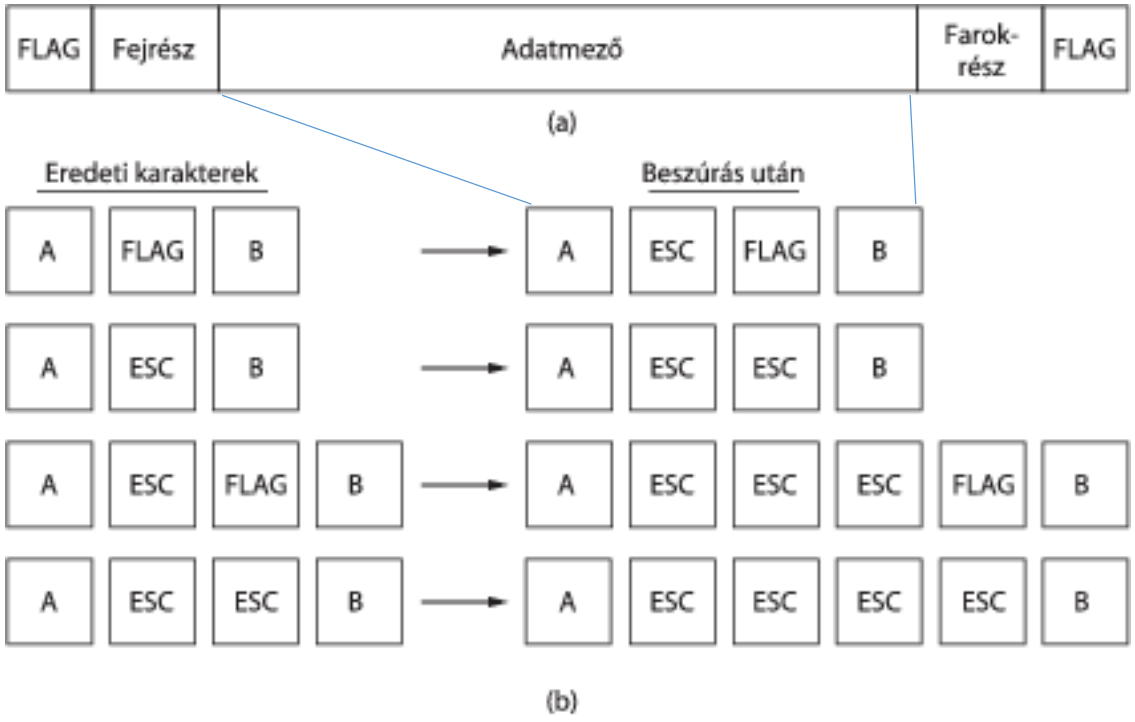
- Bájtszám: a keret mérete
- Hiba esetén az újraszinkronizáció nem lehetséges ☹



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

Keretezés (3)
Jelzőbájtok (byte stuffing)

Speciális jelzőbájt (FLAG) jelzi a keret határait
Mi történik, ha ugyanilyen bájt van az adatmezőben is? → kivételbájt (ESC) használata
Mi történik, ha ESC van az adatmezőben is? → kivételbájt (ESC) használata



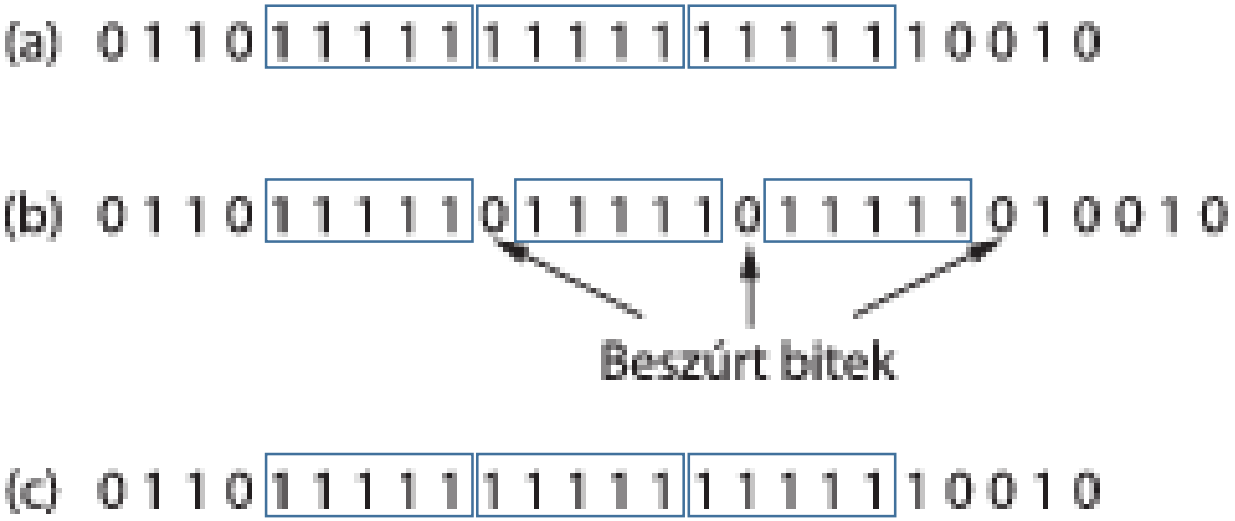
PPP
(Point-to-Point Protocol)

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

Keretezés (4)
Jelzőbitek (bit stuffing)

Jelzőbájt: 0x7E (0111 1110)
Szabályok:

- ADÓ: Ha 5db 1-es van az adatban egymás után, akkor 1 db 0 beszúrásra kerül
- VEVŐ: Ha 111110 bitmintát talál, törli a 0-t



HDLC
(Highlevel Data Link Control)
USB szabvány

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

Keretezés (5)
Fizikai rétegbeli kódolás megsértése

- Pl.: a fizikai rétegbeli moduláció lehet redundáns
  - Több lehetséges kódszó, mint amennyit felhasználunk
  - A nem használt kódszavak használhatók keretnek
- Támogatás kell a fizikai rétegből!
  - A fizikai és adatkapcsolati réteget együtt kell implementálni

Hibakezelés

- Megérkezett-e a keret? Hibamentes?

- Biztosítani kell, hogy a fogadó oldal hálózati rétegéhez
  - minden keret megérkezzen, …
  - pontosan egy példányban, …
  - a megfelelő sorrendben

- Módszerek:
  - Hibajavító kódolás
  - Hibadetektáló kódolás
  - Hiba esetén újraküldés
    - Nyugta üzenetek (acknowledgment, ACK)
    - Időzítő (timer)

Hibadetektálás és hibajavítás (1)

- Hibajavító kódok
  - Előre irányuló hibajavításnak is hívják (FEC - Forward Error Correction)
  - Elég redundanciát tartalmaz ahhoz, hogy a vevő kitalálja, mi lehetett az elküldött adat
- Hibadetektáló kódok
  - Elég redundanciát tartalmaz ahhoz, hogy a vevő észrevegye, hogy hiba történt,
  - Nem tudja, hol a hiba (pl. melyik bit rossz)
  - Kérheti az újraküldést
- A hibamodell alapján választható ki az optimális megoldás
  - Nagyon ritka hiba → hibadetektálás + újraküldés
  - Sűrű hiba →hibajavítás

Hibadetektálás és hibajavítás (2)

Egyszerű példák:

- Küldjük el az üzenetet kétszer
  - Ez jó lehet hibadetektálásra (hány bit hiba esetén tévedhet?)
  - Nem lehet a hibát javítani vele

  00100111011010101110101011101011**0**0
  00100111**0**01010101110101011101011**1**0

- Küldjük el az üzenetet háromszor
  - Ez jó lehet hibajavításra is (hány bit hiba esetén tévedhet?)

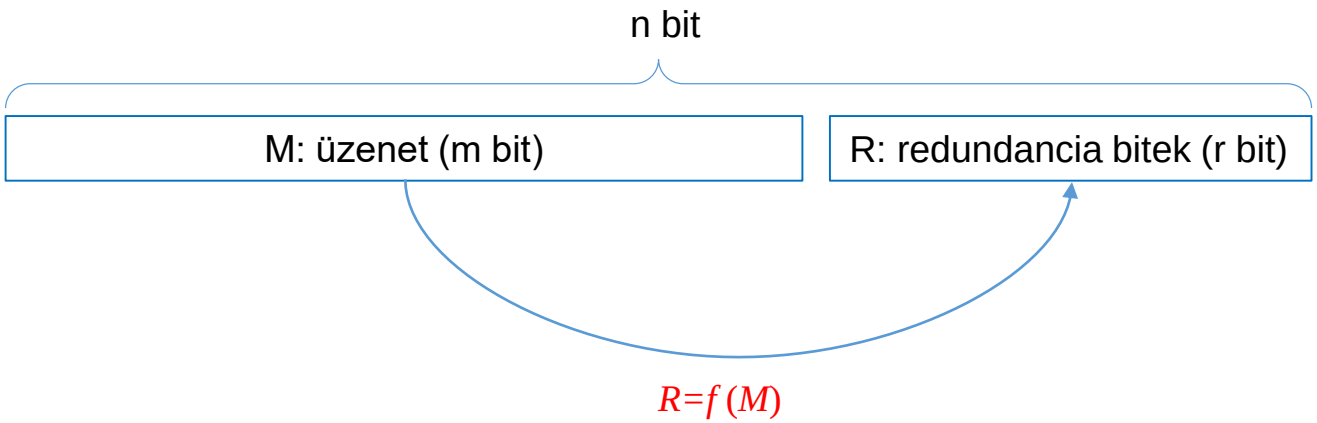
  00100111011010101110101011101011**0**0
  00100111**0**01010101110101011101011**1**0
  00100111**0**01010101110101011101011**0**0

- Okosabb megoldások is léteznek…

Hibajavító kódolás (1)

- **Hamming kódolás**
- Bináris konvolúciós kódok
- Reed-Solomon kódok
- Alacsony sűrűségű paritásellenőrző kódok

- Gyakori megoldás (szisztematikus blokk-kódok):
  - Redundancia bitek (ellenőrző bitek)



n-bites kódszó:
n=m+r
Kódsebesség: m/n

Példa
1-bit javító Hamming kód:
$m = 2^r - r - 1$
r=3, m=4
r=4, m=11
r=5, m=24
r=6, m=57
…

Hibajavító kódolás (2)
Hamming távolság

- Hamming távolság:
  - Két bitsorozat közötti távolságot méri
  - Hány bitet kell invertálni az első sorozatban, hogy a második sorozatot kapjuk?
  - Pl.: 01110010 és 11100011 távolsága: HD=3

- Ha a Hamming távolság *két érvényes kódszó között* HD=d+1
  - Tudunk *jelezni* d bites hibákat

- Ha a Hamming távolság *két érvényes kódszó között* HD=2d+1
  - Tudunk *javítani* d bites hibákat

Hibajavító kódolás (3)
Hamming kódolás

- Ellenőrző bitek:
  - Azon bitpozíciókon, amelyek kettő hatványai (1, 2, 4, 8, …)
  - A többi bit adatbit
- Írjuk fel az adatbitek sorszámát 2 hatványainak összegeként:
  - Pl. 11. bitre 11=8+2+1
- Az összegben szereplő paritás-indexek fogják az adott bitet ellenőrizni
  - Pl. a 11. adatbitet a 8., a 2. és az 1. paritásbitek ellenőrzik…
  - ezért fenti négy bit paritása páros lesz: 0+0+1+1=2

p_1	p_2	m_3	p_4	m_5	m_6	m_7	p_8	m_9	m_{10}	m_{11}
0	0	1	0	0	0	0	1	0	0	1

- Adáskor kiszámítjuk az ellenőrző biteket.
- Vételkor ismét kiszámítjuk az ellenőrző biteket.
  - Hiba **szindróma**: ahol a számított ellenőrző bit különbözik a vett ellenőrző bittől, ott 1, különben 0.

Hibajavító kódolás (4)
Hamming kódolás

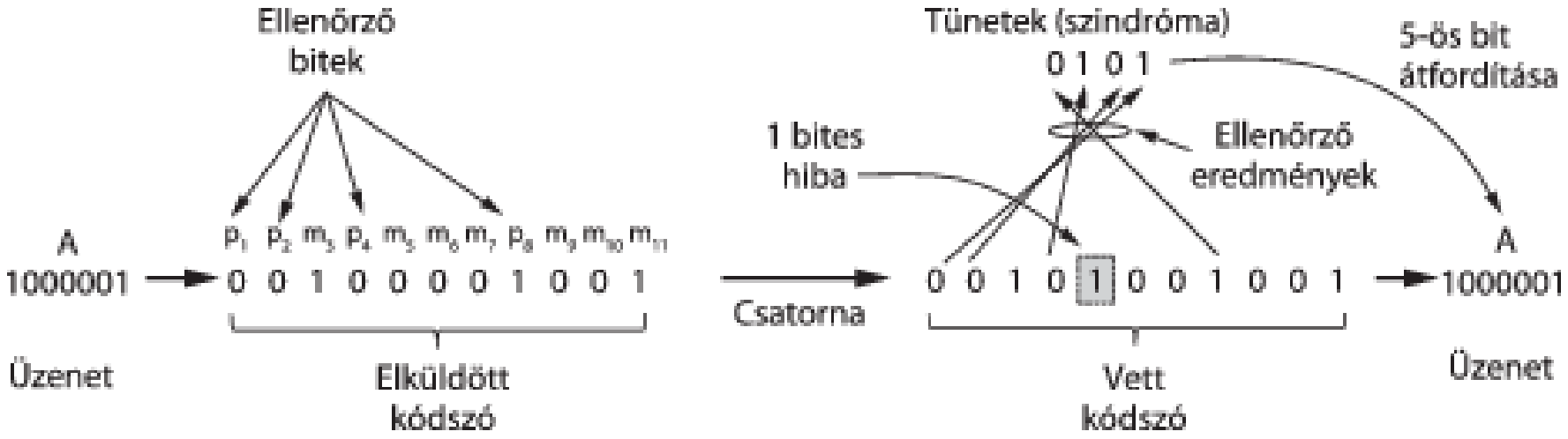
- Példa: 7 adatbit, 4 paritás bit
- Adat: 1000001
- Paritásbitek: 1, 2, 4, 8
- Kezdeti kódszó:

p_1	p_2	m_3	p_4	m_5	m_6	m_7	p_8	m_9	m_{10}	m_{11}
		1		0	0	0		0	0	1
		1+2								1+2+8

- Teljes kódszó:

p_1	p_2	m_3	p_4	m_5	m_6	m_7	p_8	m_9	m_{10}	m_{11}
0	0	1	0	0	0	0	1	0	0	1

Hibajavító kódolás (5)
Hibajavítás Hamming kódolással



Példa a (11, 7) Hamming-kód egybites hibajavítására

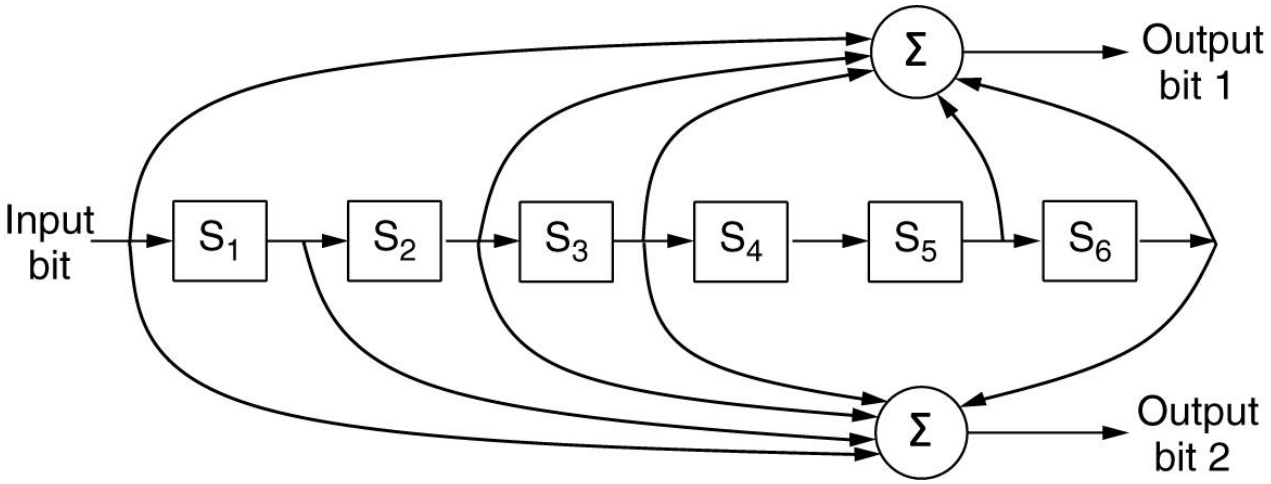
Hibajavító kódolás (6)
Konvolúciós kódok

Ez nem blokk-kód, nem is szisztematikus
A bemeneti bitsorozatot egy tárolósorban tároljuk, minden új bitnél eggyel jobbra léptetünk
A kimenetet a tárolt állapotokból számítjuk (XOR)
A dekódolás: bizonytalan döntésű dekódolás (soft decision decoding)

- Bizonytalanságokat is figyelembe lehet venni (pl. nagy valószínűséggel 1, talán 0, …)
- A logikai szinteket nem döntik el rögtön

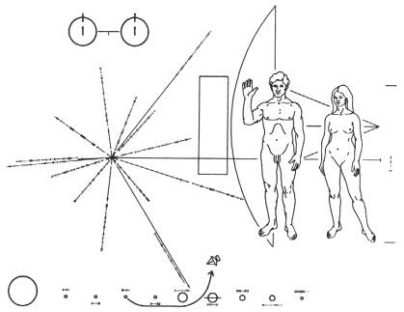
Legnépszerűbb: NASA bináris konvolúciós kódja

- 1 bitből 2 bitet képez (kódsebesség: 0.5)



A 802.11. szabványban alkalmazott NASA bináris konvolúciós kód

Voyager WiFi



Hibajavító kódolás (7)

- Reed-Solomon kódok
  - Lineáris (többnyire szisztematikus) blokk kódok, véges mezők feletti polinomokon alapulnak
  - Gyakori: m=233, r=32
  - 16 szimbólum (bájt) hibát is javítani tud
  - Akár 16x8=128 bitnyi hibacsomót is javítani tud

DSL
Műholdas kommunikáció

- Kis sűrűségű paritásellenőrző kódok
  - LDPC (Low-Density Parity Check) codes
  - Lineáris blokk kódok
  - Minden kimenő bit a bemenő bitek egy csoportjától függ
  - A kód egy ritka mátrixszal jellemezhető
  - Nagy blokkméretű adatokra különösen jó
  - Kiváló hibajavító képességekkel bír
  - Új szabványokban népszerű

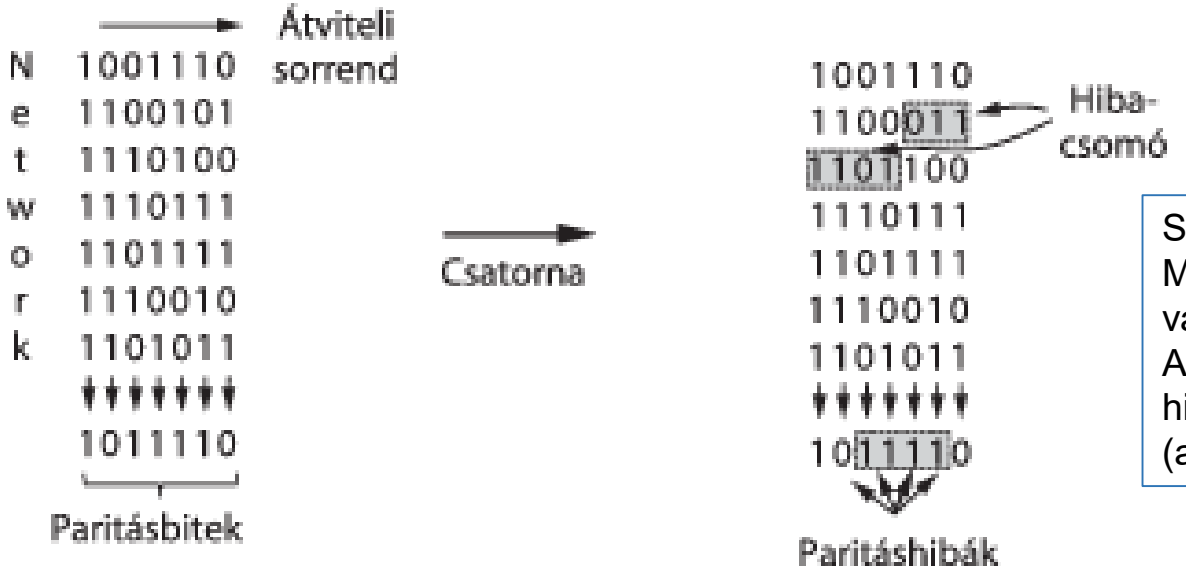
10Gbites Ethernet
PLC
Új 802.11

Hibadetektáló kódok (1)

- Lineáris, szisztematikus blokk kódok
  - paritásbit képzése
  - ellenőrző összeg képzése
  - ciklikus redundancia ellenőrzés (Cyclic Redundancy Check, CRC)

Hibadetektáló kódok (2)

- Paritásbit képzése
  - Egy paritásbit egyetlen bit hiba jelzésére alkalmas
  - A hibák azonban gyakran csoportosan fordulnak elő
  - Erre az összefésült paritásbiteket alkalmazzák
    - Paritásbit oszloponként
    - Átvitel soronként (végül a paritásbitek)



Sok hiba esetén:
Minden oszlopban ½ valószínűséggel jelezzük a hibát.
Annak a valószínűsége, hogy a hibát nem jelezzük: $(0.5)^n$
(ahol n az oszlopok száma)

Hibadetektáló kódok (3)

- Ellenőrző összeg
  - A paritásbit, vagy ezek csoportja is ilyen (egyszerű)
- IP 16 bites ellenőrző összeg:
  - 16 bites szavakat képzünk
  - Ezek összege (mod 2^{16}) az ellenőrző összeg
    - Az összegzésnél túlcsorduló biteket a legkisebb helyi értékhez adjuk
  - Jobb tulajdonságokkal bír, mint az egyszerű paritásbitek

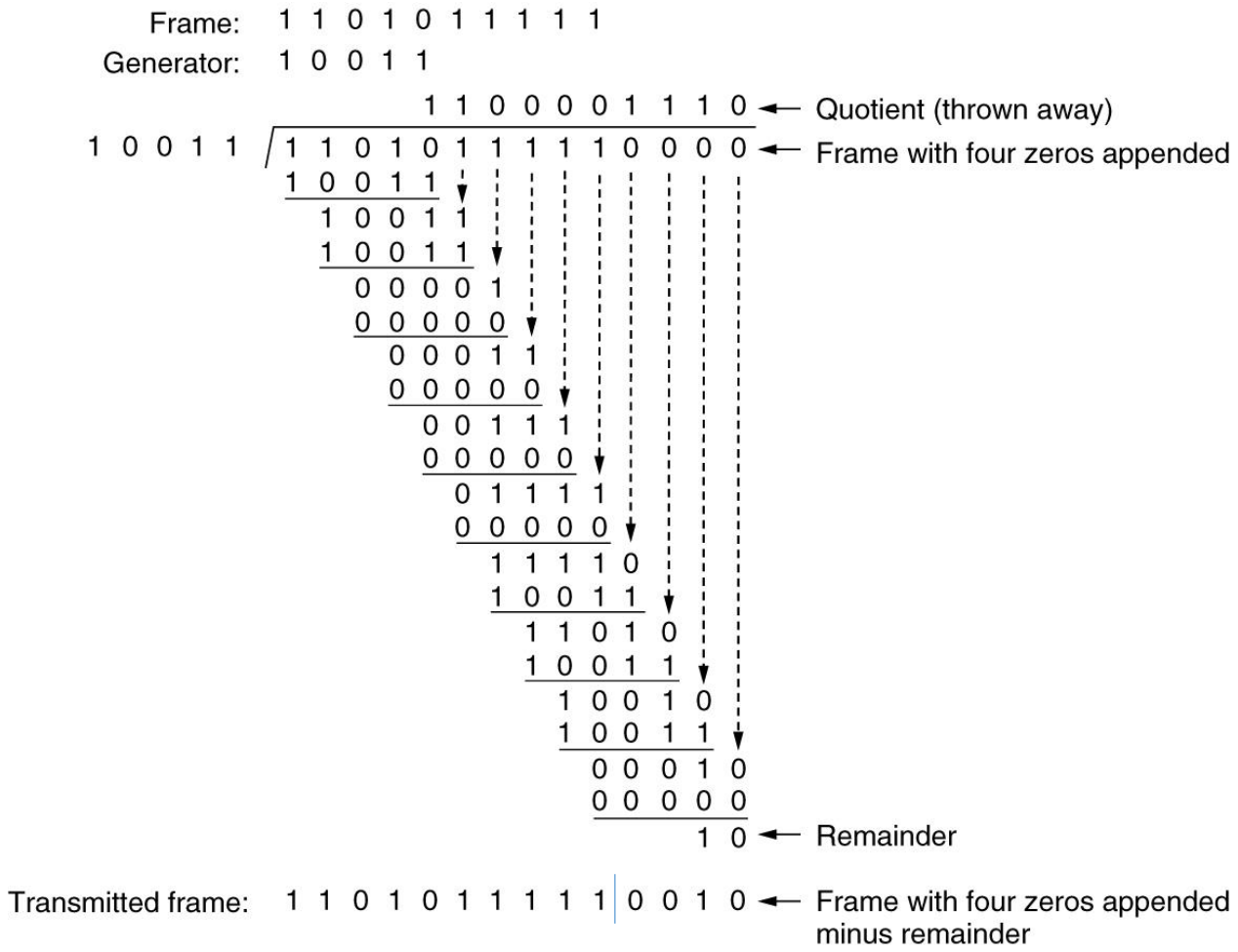
Hibadetektáló kódok (4)

- Ciklikus redundancia kód (CRC)
- Bitsorozat ≡ polinom
  - Pl.: 1 0 0 1 1 ≡ $1 \cdot x^4 + 0 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x + 1 \cdot 1 = x^4 + x + 1$
- Adat: M(x) polinom, fokszáma m
- Generátor: G(x) polinom, fokszáma r
  - Legmagasabb és legalacsonyabb bit: 1
  - r < m
- CRC: az $M(x)x^r/G(x)$ osztási maradéka
  - Sok hibát észlel:
  - 1bit, 2 bit, páratlan számú hiba, legfeljebb r-bites hibacsomó

Ethernet
SONET
802.11

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$$

Hibadetektáló kódok (5)



CRC számítása

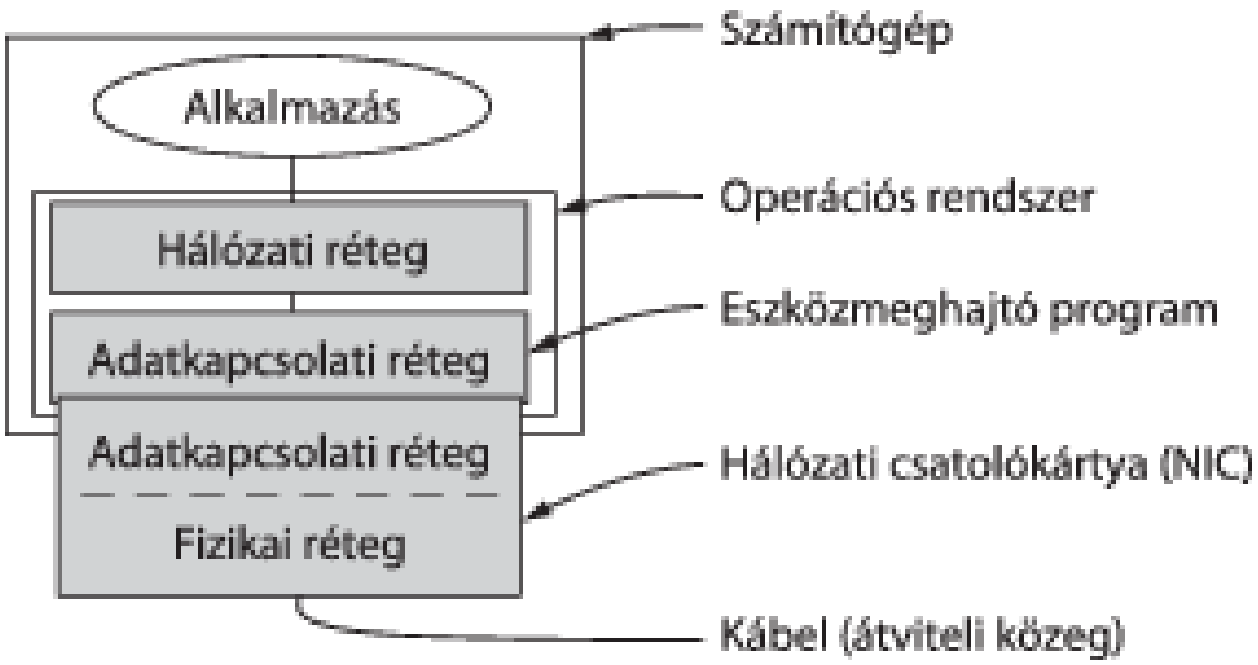
Elemi adatkapcsolati protokollok (1)

- Hibakezelési stratégiák
  - Hibajavítás
  - Hibadetektálás és **újraküldés**
- Módszer:
  - Vevő automatikusan **nyugtá**t (ACK) küld a sikeresen vett keretekről
    - Nyugta: rövid keret. Esetleg más üzenet „nyakában” is ülhet
  - Esetleg negatív nyugtát (NACK) is küldhet a sikertelen vételről
  - Adó automatikusan **újraküld**i a sikertelenül küldött kereteket
    - Honnan tudja, hogy sikertelen?
      1. NACK (de ez akár el is veszhet!)
      2. Nem kap ACK-ot egy ideig (timeout)
         - Mennyi legyen a **várakozási idő**?
           - Nem túl sok, mert akkor sokáig tart a javítás
           - Nem túl rövid, mert esetleg nem ér vissza az ACK
           - Kb.: kicsivel több, mint az oda-visszaút ideje

Elemi adatkapcsolati protokollok (2)

Hol implementáljuk az adatkapcsolati protokollokat?

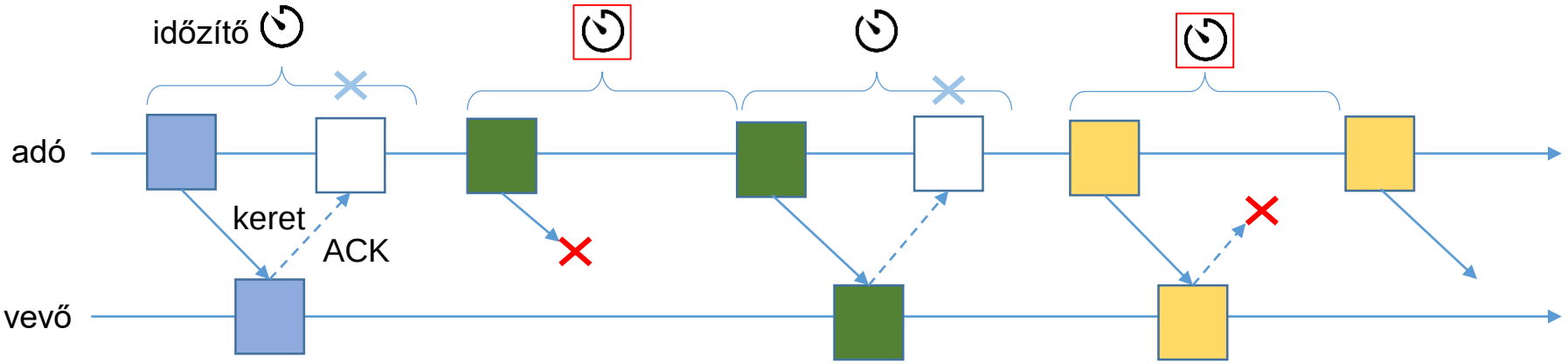
- Szoftver (operációs rendszer)
- Hardver (hálózati csatolókártya - NIC)



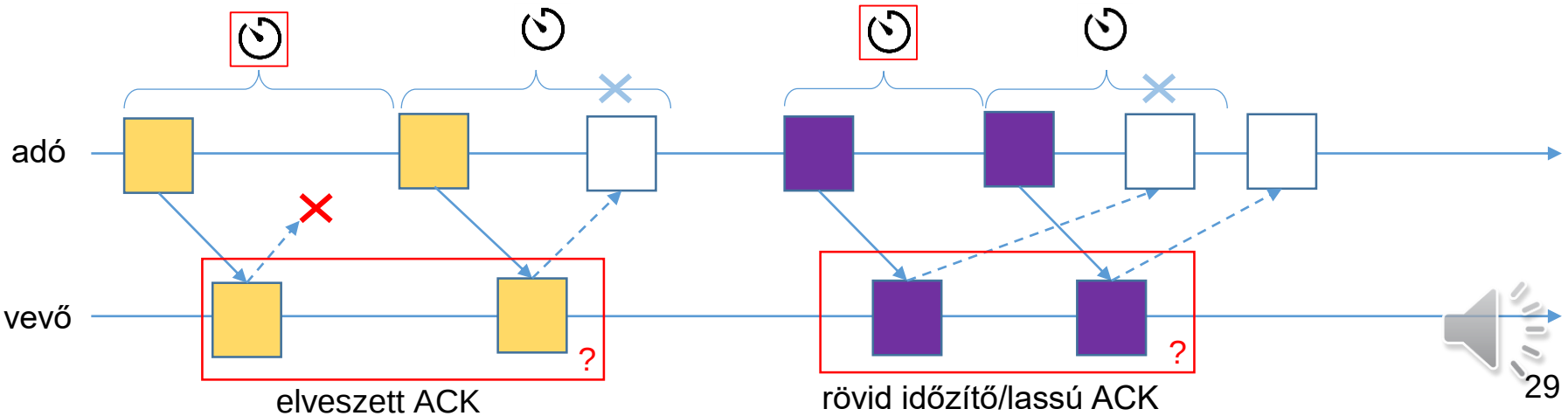
A fizikai, az adatkapcsolati és a hálózati réteg megvalósítása

Elemi adatkapcsolati protokollok (3)

- Alapgondolat:

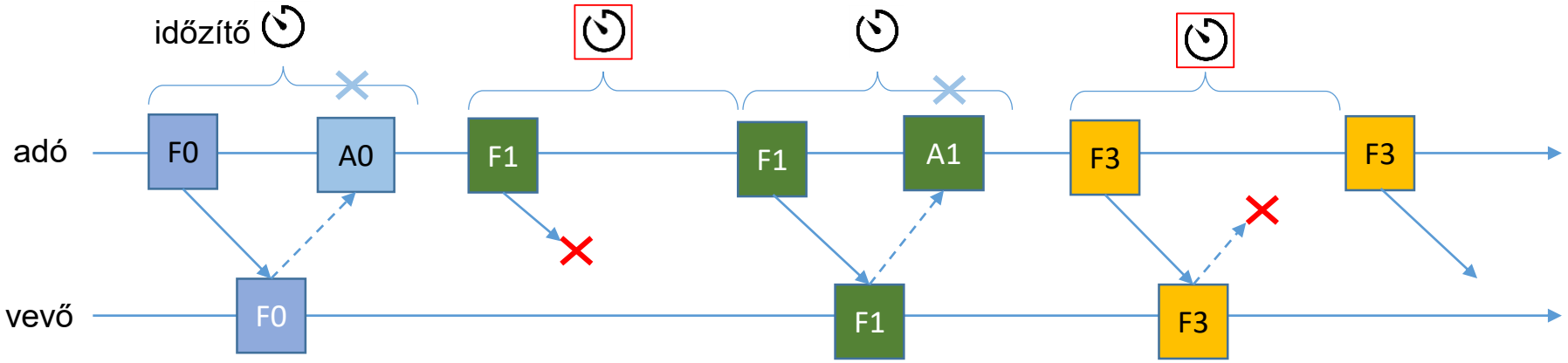


- Bonyodalmak: duplikált üzenetek a vevőnél.

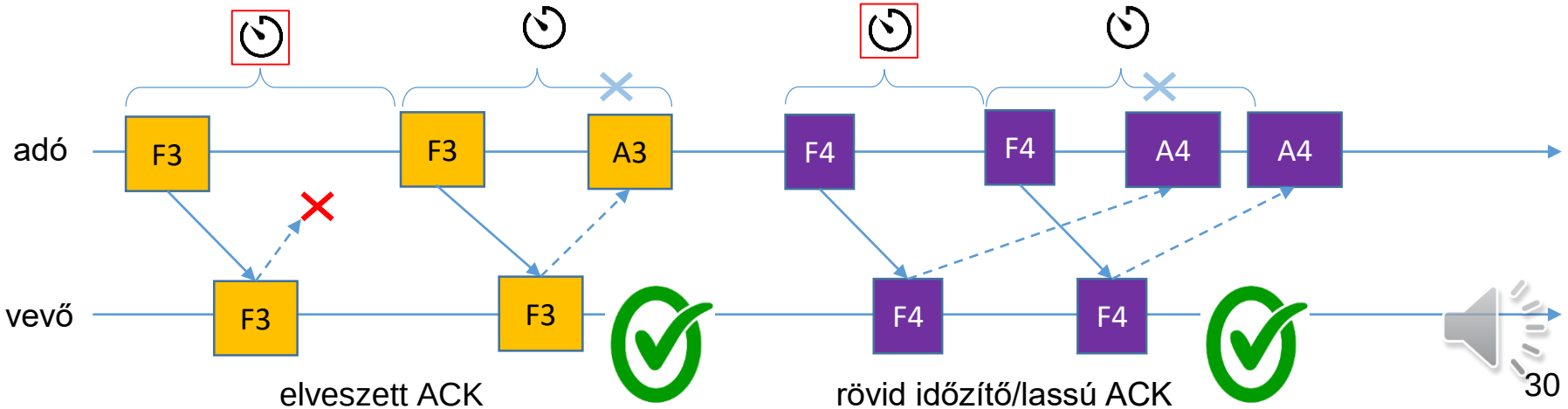


Elemi adatkapcsolati protokollok (4)
Megáll és vár protokoll (Stop-and-Wait)

- Minden üzenetet és nyugtát sorszámoznunk kell, így megkülönböztethetők



- Mi a helyzet a gonosz eseteknél?



Elemi adatkapcsolati protokollok (5)
Megáll és vár protokoll (Stop-and-Wait)

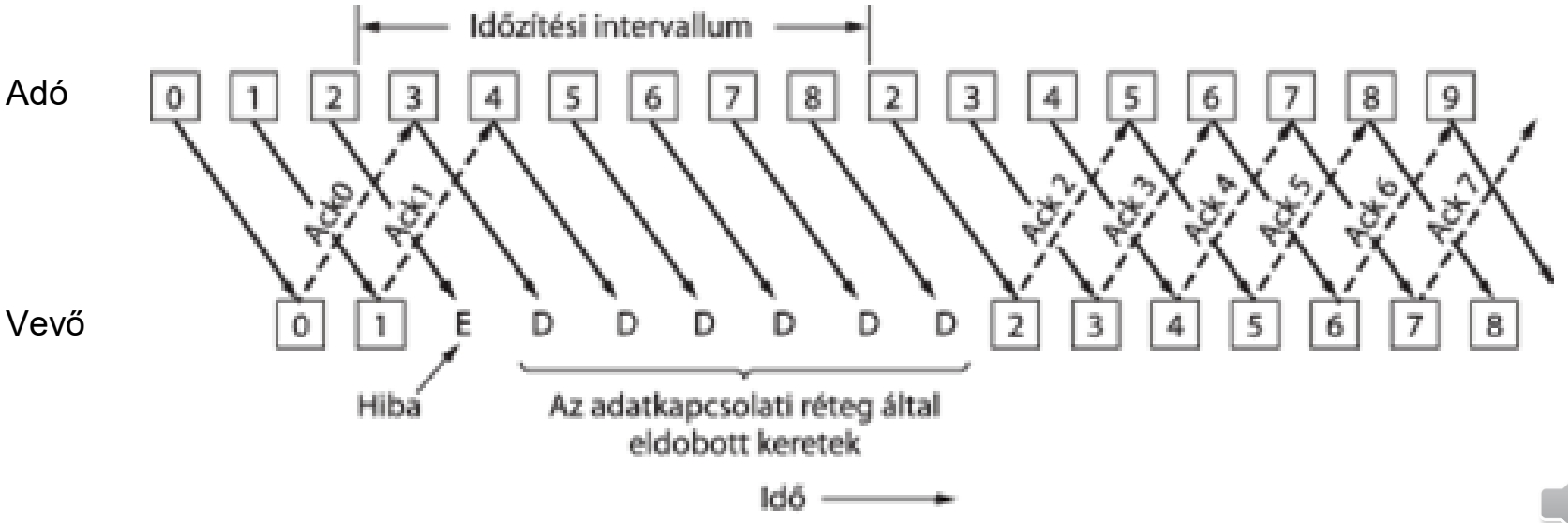
- A megáll és vár protokoll
  - Amíg nincs nyugta, nem küldi a következő keretet
  - Egyszerre csak egy keret van úton
  - Helyi hálózaton (kis késleltetés) használható, de rossz nagy sávszélesség-késleltetés szorzat (BD) esetén
- Példa:
  - Csatorna sávszélessége: B=1Mb/s
  - Késleltetés (egy irányban): D=50ms
  - Keret mérete: M=10kbit
  - Üzenet megfordulási idő: 100ms, azaz 10 üzenet/sec küldhető
  - 10 üzenet = 100kbit → adatsebesség = 100kb/s (<<1Mb/s)
- Mi a baj?
  - Egyszerre csak egy üzenet van úton
  - Pl.: BD=50kb, vagyis egyszerre akár 5 üzenet is lehetne úton…

Elemi adatkapcsolati protokollok (6)
Csővezetékezés (pipelining)

- Ablakok használata
  - Adó oldali ablak mérete:
    - Ennyi keret lehet egyszerre elküldve, de még nem nyugtázva
    - Ablak optimális mérete: kb. 2BD bit
  - Adó oldalon tároljuk az elküldött, de nem nyugtázott kereteket
    - Mindegyikhez egy időzítőt indítunk
    - A nem sikeres üzeneteket újraküldjük
    - Sikeresen elküldött üzeneteket kivesszük az ablakból
    - Hálózati réteg beteszi az ablakba az elküldendő csomagot (ha van üres hely)
  - Vevő oldalon tároljuk a befogadható üzeneteket
    - több módszer is lehet:
      - vevő ablak mérete = 1 (n-visszalépéses)
      - vevő ablak mérete nagyobb (szelektív ismétlés)
    - A sikeresen vett keretekből a csomagokat a helyes sorrendben a hálózati rétegbe feladja

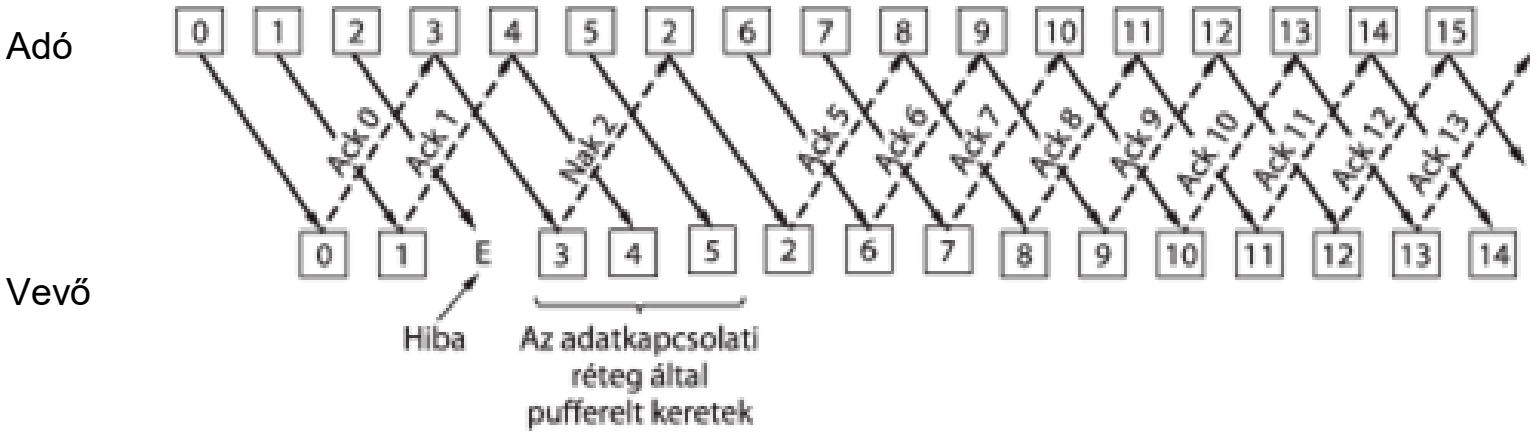
Elemi adatkapcsolati protokollok (7)
Az n-visszalépéses csővezetékezés

- (go-back-n protocol)
- A vevő a **soron következő** üzenetre vár, csak azt fogadja (és nyugtázza)
- Minden mást eldob (vevő oldali ablak mérete: 1)
- Időzítő: elveszett üzenetre nem jön vissza ACK, innen **újrakezdi** az adó
- Hiba estén lassú a javítás…



Elemi adatkapcsolati protokollok (8)
Az szelektív ismétléses csővezetékezés

- (selective repeat)
- A vevő a hibás üzeneteket eldobja, de tárolja a helyesen vett üzeneteket (az ablakméreten belül)
- Nyugta: az utolsó olyan helyesen vett üzenetre, amely előtt minden más üzenet is helyesen megjött (pl.: F1, F2, F3, F4, x, F6, F7, F8, x, F10 → A1, A2, A3, A4)
  - Kumulatív ACK
- Adó időzítő lejár: az *első* nem nyugtázott üzenet küldi
- Gyakran használják negatív ACK-val együtt

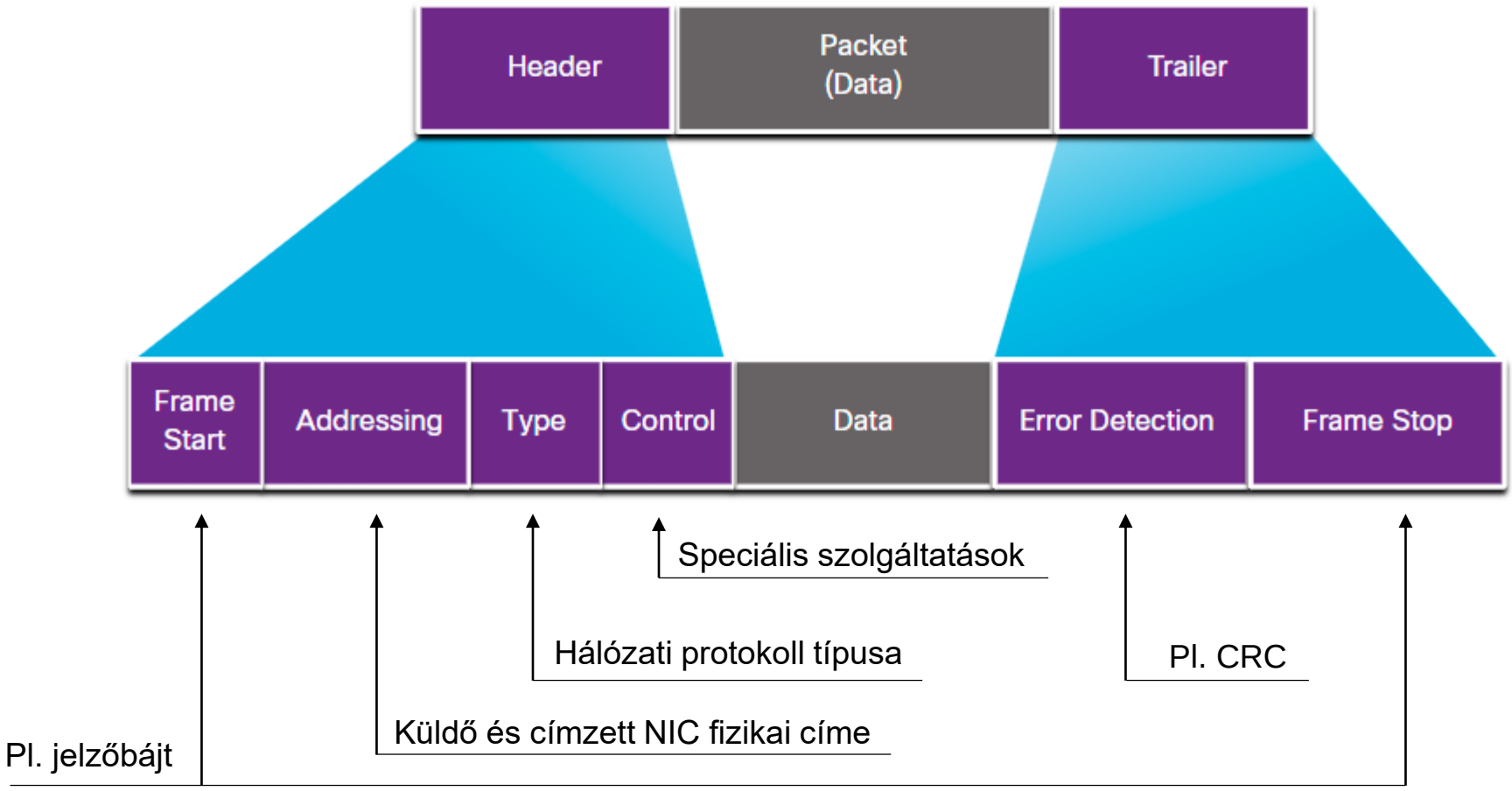


Elemi adatkapcsolati protokollok (9)
Kétirányú kapcsolatok

- Eddig csak egyirányú kapcsolatokat vizsgáltunk
- A valóságban kétirányú kapcsolatokat használunk
- Mindkét fél adhat és vehet (egyben nyugtázhat)
- A nyugták utazhatnak külön keretben, de…
- … ha van ellenirányú forgalom, akkor egy normál adatkeretre is ráültethető a nyugta (hiszen ez csak néhány bájtnyi info)

  → piggybacking

Keretek felépítése (példa)

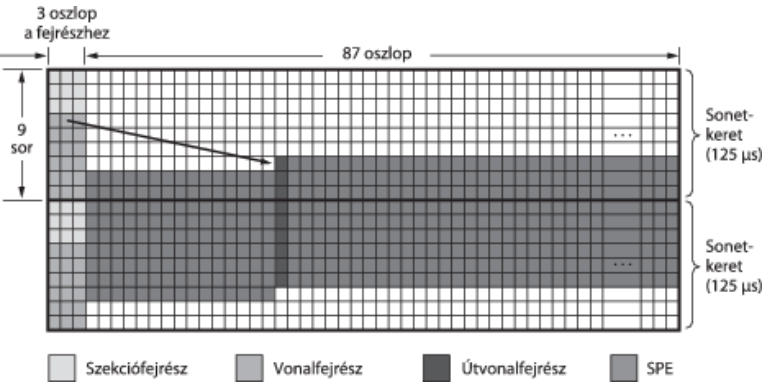


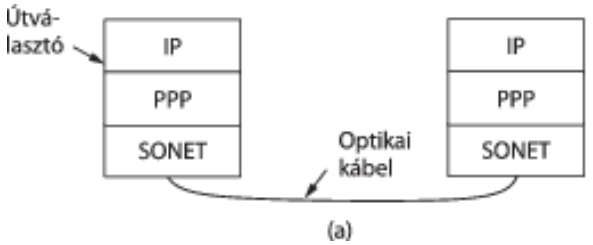
Példák adatkapcsolati protokollokra

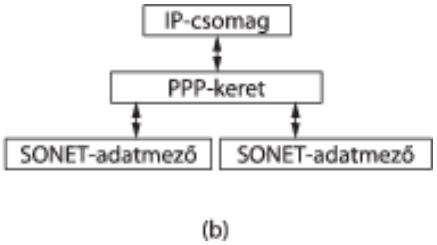
- Kerettovábbítás SONET felett
- PPP
- ADSL
- Ethernet (később)
- WiFi (később)

Kerettovábbítás SONET felett

- Ismétlés: SONET
  - SONET keretek 125μs-onként, keretenként 810B
  - Folyamatos SONET-keret áramlás (akkor is, ha nincs adat)
  - Szinkronizálást biztosít



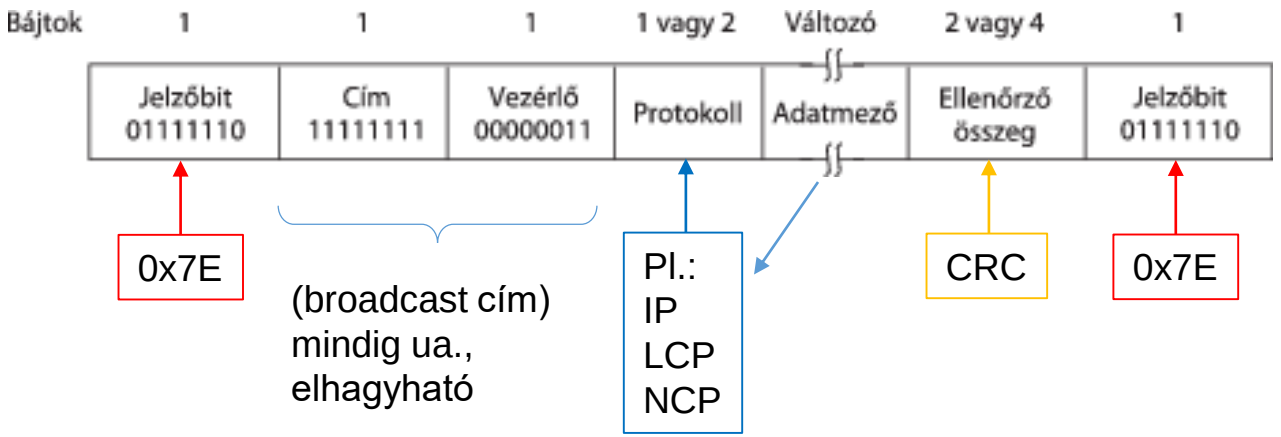




- A SONET-keretbe egy PPP keret kerül
  - PPP: kétpontos protokoll (Point-to-Point Protocol)

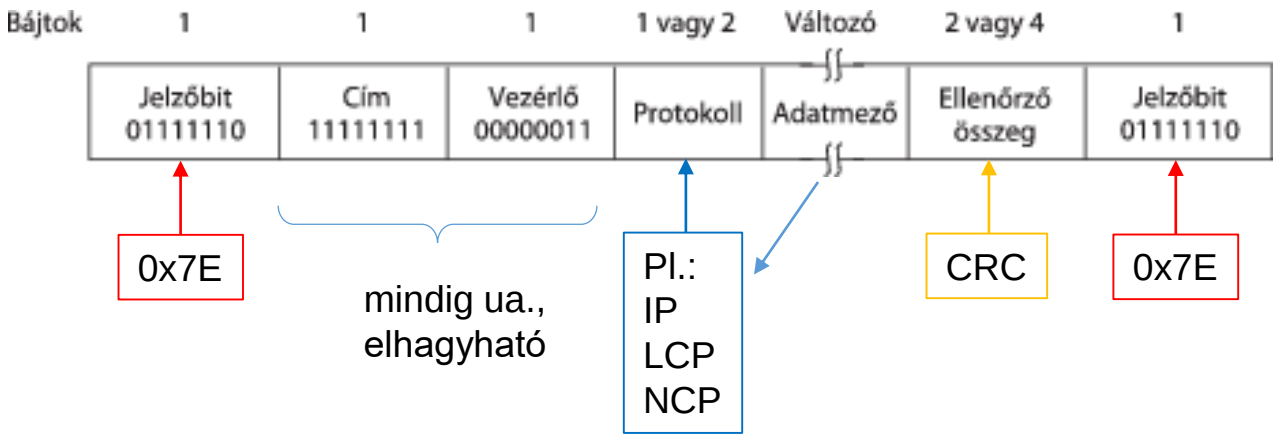
PPP (1)

- PPP: kétpontos protokoll
  - SONET és ADSL is ezt használja
- Keretezés:
  - Bájt-beszúrás
    - Jelzőbájt (FLAG): 0x7E, kivételbájt (ESC): 0x7D
    - Járulékos trükk: ESC után 0x20-val XOR-ozva lesz a következő bájt
      - Így nem lesz FLAG sehol az adatban, könnyebb a fejrész keresése



PPP (2)

- Adatkapcsolat-rétegbeli protokollok:
  - LCP: Link Control Protocol
    - Vonal felélesztése
    - Opciók megbeszélése (pl. cím és vezérlő elhagyása, protokollmező hossza)
    - Vonal elengedése
  - NCP: Network Control Protocol
    - Hálózati réteg opcióinak megbeszélése
    - Pl.: IP címek hozzárendelése

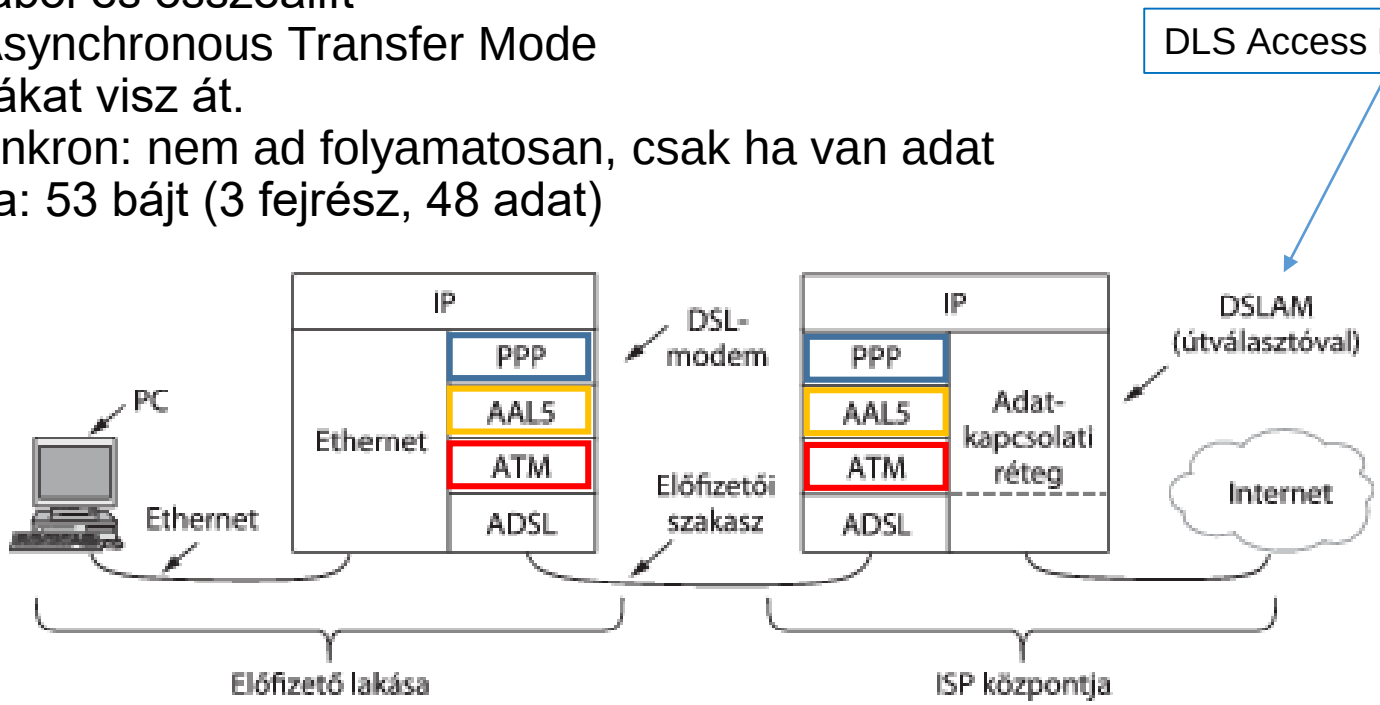


ADSL (1)

PPP:
- Csak Adat és Protokoll mezők vannak
- Többi mező szerepét átveszi az AAL5

AAL5: ATM Adaptation Layer 5
- Köztes réteg PPP és ATM között
- Darabol és összeállít

ATM: Asynchronous Transfer Mode
- Cellákat visz át.
- Aszinkron: nem ad folyamatosan, csak ha van adat
- Cella: 53 bájt (3 fejrész, 48 adat)



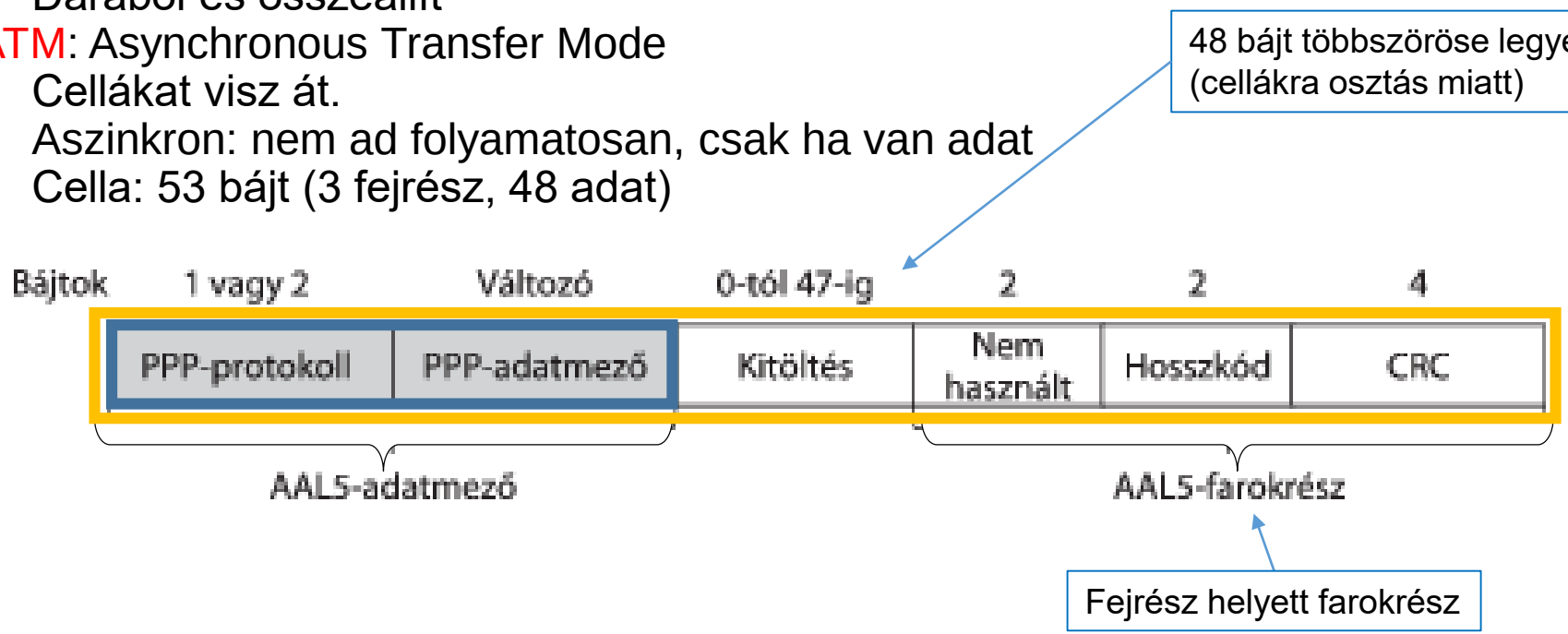
ADSL protokoll verem

ADSL (2)

PPP:
- Csak Adat és Protokoll mezők vannak
- Többi mező szerepét átveszi az AAL5

AAL5: ATM Adaptation Layer 5
- Köztes réteg PPP és ATM között
- Darabol és összeállít

ATM: Asynchronous Transfer Mode
- Cellákat visz át.
- Aszinkron: nem ad folyamatosan, csak ha van adat
- Cella: 53 bájt (3 fejrész, 48 adat)



PPP adatokat hordozó AAL5 keret