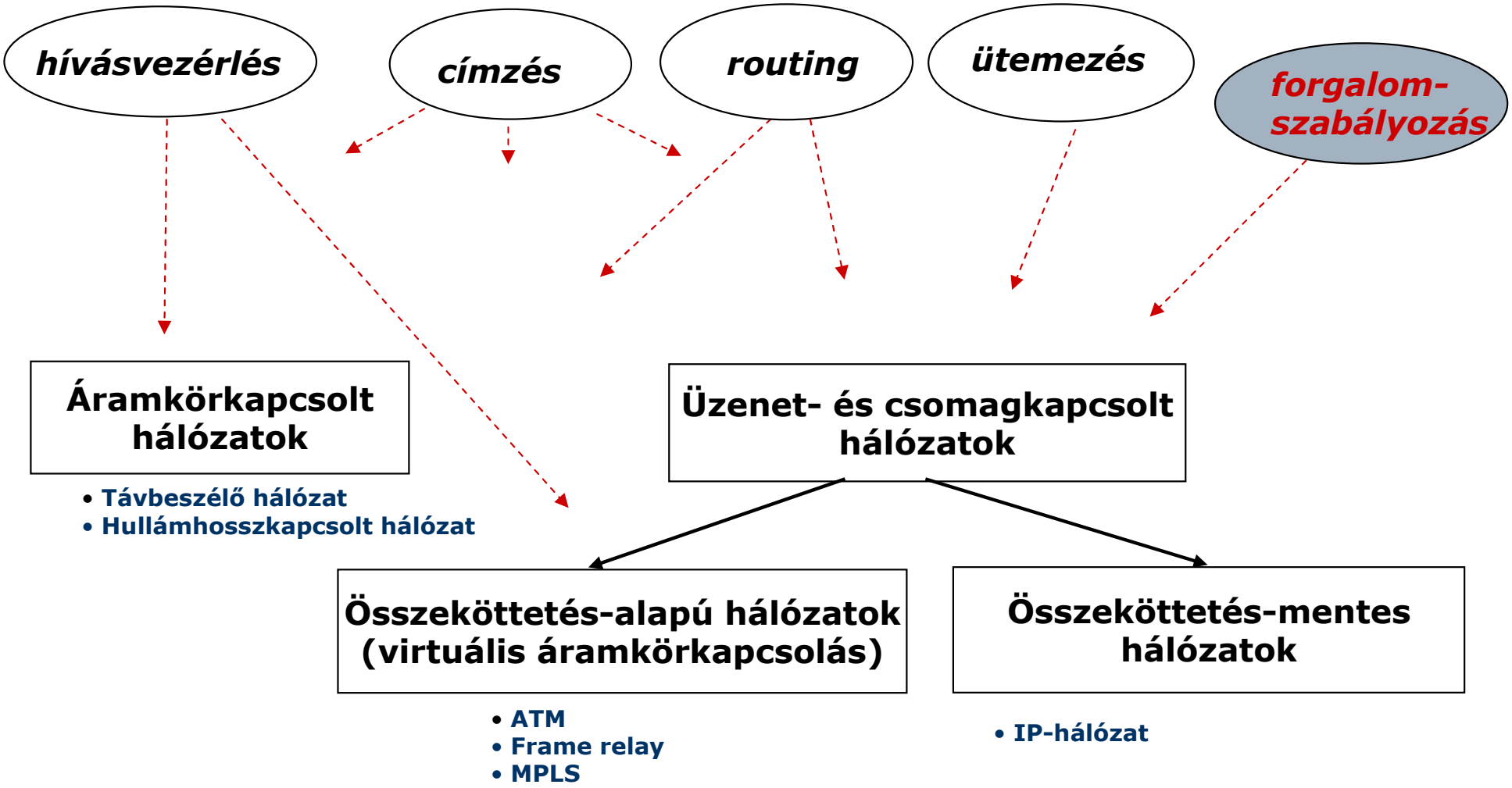
További két fontos funkció:

- Forgalomszabályozás
- Hibakezelés

Forgalomszabályozás – *flow control*

Újabb fontos funkciók és alapelvek

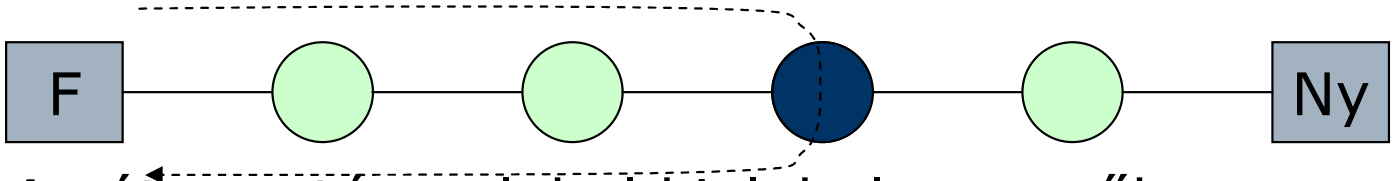


Definíció és a feladat meghatározása

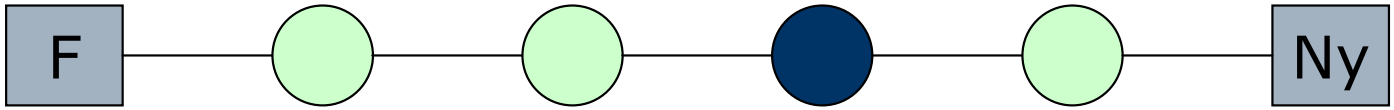
- Két rokon feladat:
- Forgalomszabályozás (*flow control*):
  - Módszerek, amelyek lehetővé teszik, hogy egy adatforrás az aktuális átviteli sebességét illessze a vevőnél és a hálózatban rendelkezésre álló kiszolgálási sebességhez
- Torlódásvezérlés (*congestion control*):
  - Azok a módszerek, amelyekkel linkek, csomópontok időszakos túlterheltségét megkíséreljük megszüntetni
    - A torlódásvezérlést értelmezhetjük általánosabban is: azok a módszerek, amelyekkel meg is lehet előzni a torlódásokat
    - Ebben az értelemben a *flow control* tekinthető a *congestion control* eszközének (előzzük meg a torlódást, mielőtt az bekövetkezne)

A működés egyszerűsített vizsgálata

- Korlátozódjunk egyetlen forrásra:
  - A csomagok több csomóponton jutnak a nyelőhöz



  - Az út mentén valahol kialakul egy szűk keresztmetszet, „bottleneck”
  - A többi csomópont elfelejthető a forgalomszabályozás modellezése szempontjából



(A késleltetéshez természetesen valamennyi csomópont hozzájárul)

A forgalomszabályozás kívánatos jellemzői

- Az alapvető cél megvalósításán túl:
  - Legyen egyszerűen megvalósítható
  - Lehető legkevesebb hálózati erőforrást vegyen igénybe
  - Hatékonysága ne függjön a szabályozott források számától
  - Biztosítsa a szabályozott források igazságos részesedését az igénybe vett erőforrásokon
  - Stabil legyen a működése

A forgalomszabályozás fajtái

- Két jelentős csoport:
  - „Nyílthurkú” – nincs visszacsatolás
  - „Zárthurkú” – van visszacsatolás
- Lehetséges a kettő kombinációja is (hibrid)
- Nyílthurkú:
  - A kommunikáció előtt a felhasználó és a hálózat forgalmi paramétereket egyeztet
  - A hálózat dönt az új összeköttetés elfogadásáról
    - *admission control, beengedésszabályozás*
  - Ennek megfelelően a hálózat erőforrásokat dedikál
  - A működés során a paraméterek ellenőrzése

Nyílthurkú szabályozás

- Forgalomleírók (traffic descriptors):
  - Egy paraméterkészlet, amely jellemzi az adatforrás viselkedését, továbbá
  - alapját képezi egy *szolgáltatási szerződés forgalmi részének*
- Egyszerű példák forgalomleírókra:
  - csúcssebesség
  - átlagsebesség
- A forgalomleírók bemenő adatai:
  - a *forgalomszabályozó*nak (regulator), valamint
  - a *felügyelő*nek (policer)

(A *regulátor* tipikusan késlelteti a túlzott forgalmat, míg a *policer* inkább eltávolítja)

Zárthurkú szabályozás

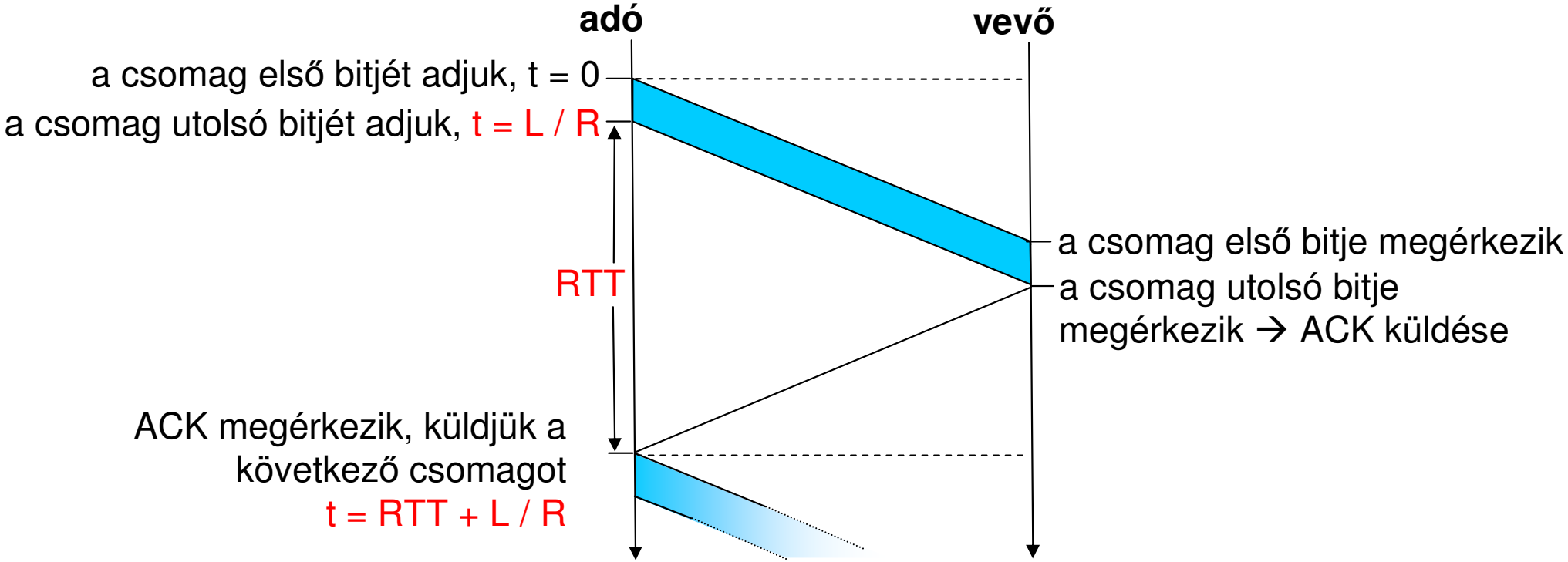
- Feltétlenül szükséges, ha:
  - nincs erőforrás-foglalás
  - túlfoglalást (*overbooking*) alkalmazunk statisztikus nyereség elérése érdekében
- Típusai:

1. generáció (csak a nyelő képessége)	**ki – bekapcsolás (on-off)**		
	stop-and-wait		
	statikus ablak (static window)		
2. generáció (a nyelő és a hálózat képessége)	**állapot vizsgálat**	**vezérlés módja**	**vezérlés helye**
	explicit	**din. ablak**	**végpont**
	implicit	**din. seb.**	**lépések**

A felsorolt módszerek

- *On-off:*
  - a nyelő engedélyezi az adást
- *Stop-and-wait:*
  - a küldő egy csomag után vár a nyugtára
- *Statikus ablak:*
  - a küldő az ablak méretével megadott számú csomag elküldése után vár csak nyugtára
- *Második generációs módszerekre a protokolloknál látunk majd példát*

A stop-and-wait működése



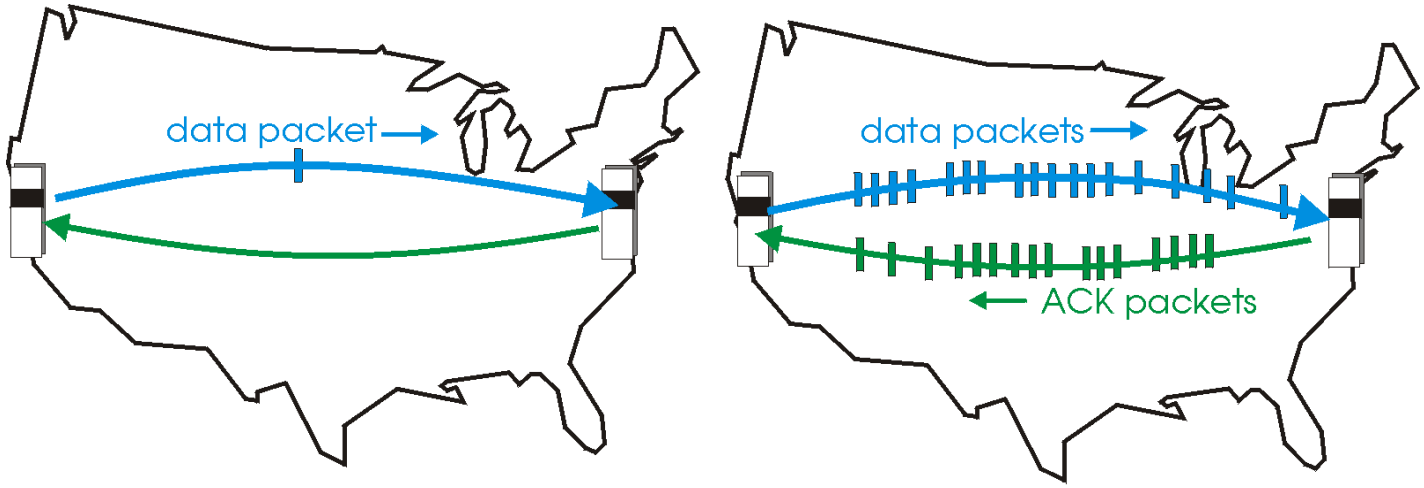
A kihasználtság: $U = \frac{L / R}{RTT + L / R}$

RTT: round-trip-time (teljes körbefordulási idő)

Statikus ablak: több egymásutáni, még nem nyugtázott csomag adása

Az adónak lehet több, adásban lévő, még nem nyugtázott csomagja

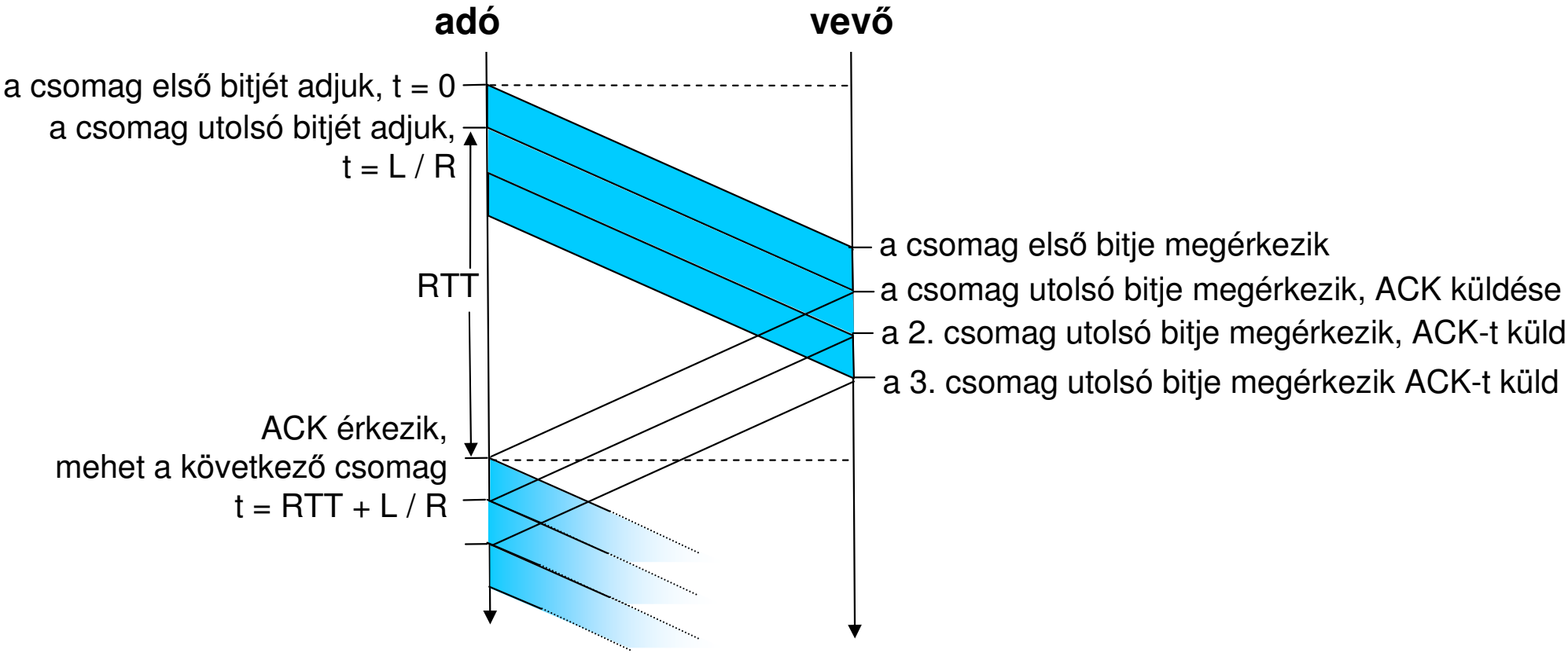
- sorszámozásra van szükség
- tárolásra van szükség az adóban



(a) a stop-and-wait protocol in operation

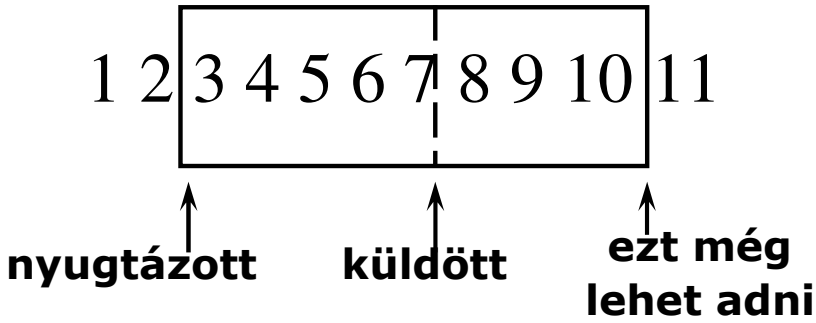
(b) a pipelined protocol in operation

Nagyobb kihasználtság a stop-and-wait-hez képest



$$U_{sender} = \frac{3 * L / R}{RTT + L / R}$$

Csúszóablakos („sliding window”) forgalomszabályozás



w = 8 (byte)

Várakozás ACK-ra: time–out

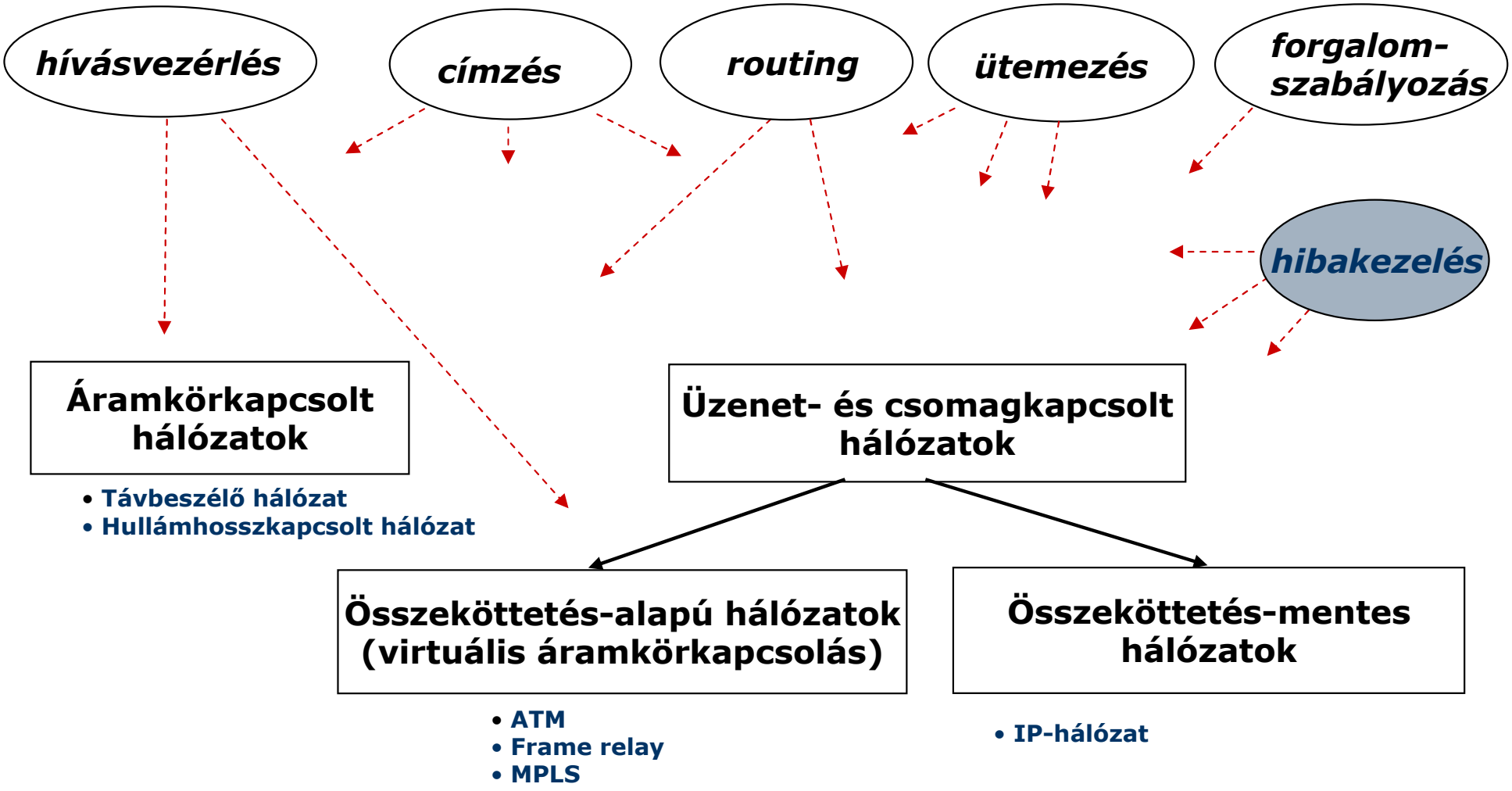
- Mekkorára válasszuk a time–out-ot?
- Probléma a túl kicsivel és a túl naggyal
- Megoldás: a teljes terjedési időhöz (round-trip time) igazítani, adaptívvá tenni
- Szabályok arra, hogy mit tegyünk, ha nem jön ACK a time-out alatt

Összefoglalás

- A küldő által kibocsátott adatátviteli sebességet kordában kell tartani, mert
  - a vevő nem győzi annak feldolgozását
  - a hálózatban képződő szűk átviteli keresztmetszetek torlódáshoz vezethetnek
- Két alapvetően eltérő megközelítés:
  - az információtovábbítást megelőzi egy szerződéskötés
  - folyamatosan befolyásoljuk a küldőt

Hibakezelés – *error control*

Újabb fontos funkció: hibakezelés



Tartalom

- A hibák keletkezésének okai
  - *bit- és csomaghibák*
- Alapvető hibajavítási stratégiák
  - *visszirányú hibajavítás és hibajavító kódolás*
- A hibajavító kódolás módszerei
  - *nem foglakozunk ezekkel, a „Kódolástechnika” tárgyban szerepelt*
  - *paritáskód, Hamming-kód, CRC, Reed-Solomon-kód, konvolúciós kódolás*
- Csomaghibák kezelése
  - *go-back-n és szelektív ismétlés*

Hibaszintek: bithiba és csomaghiba

- A végpontok közötti *(end-to-end)* átvitel során két szinten történhet hiba:
  - bithiba: 0 → 1, 1 → 0
    - megj.: törléses hiba
  - csomaghiba, amely lehet:
    - csomagvesztés
    - csomagtöbbszöröződés
    - helytelen csomagsorrend
- A hibakezelés az a folyamat, amellyel a bármelyik szinten jelentkező hiba *észlelhető* és/vagy *javítható*
  - *hibák **észlelése***
  - *hibák **javítása***

Bithibák okai

- Az információt a fizikai közegen valamilyen jel állapotának (pl. feszültségének, frekvenciájának, fázisának, intenzitásának) megváltoztatásával továbbítjuk
- Digitális átvitelnél egy v. több bináris értéknek (0-nak illetve 1-nek) egy-egy jelállapotot feleltetünk meg
- A vételi oldalon a jel értékét előre meghatározott referenciákkal összehasonlítva nyerjük vissza az információt
- A vett jel ugyanakkor csak ideális átvitel esetén egyezik meg pontosan valamelyik referenciával
- Negatív hatások: zajok, zavarok, csillapítás, jelalaktorzulás

Zajok okozta bithibák

- A zajok típusai és modelljei:
  - „véletlen zaj”:
    - a zajminták Gauss- (normális) eloszlású vsz.-v.-val modellezhetők
    - tipikusan a termikus zaj ilyen
    - szórványos, egymástól független bithibákat okoz
  - impulzuszaj:
    - nem normáleloszlású
    - tipikus forrásai erősáramú berendezések, amelyek pl. szikrákat bocsátanak ki
    - hibacsomókat okoz

Bithibák mobil rádióhálózatokban

- Cellás mobil rádióhálózatokban a cellaváltás és elhalkulás hibalöketeket eredményez
  - a cellaváltás tipikusan mintegy 150 ms időtartamú, ezalatt a vett jel szinte teljes egészében hibás
  - ez megelőzhető, ha előbb létrehozzuk az új kapcsolatot és csak azután bontjuk a régit, de ez rontja a frekvenciák kihasználtságát
- A vezeték nélküli átvitel során
  - a terjedés útjában lévő közeg
    - jelerősség-csökkenést okoz,
  - a különböző tereptárgyak visszaverődést okoznak
  - ezek összefoglaló neve „fading”, hatása különböző struktúrájú bithibák

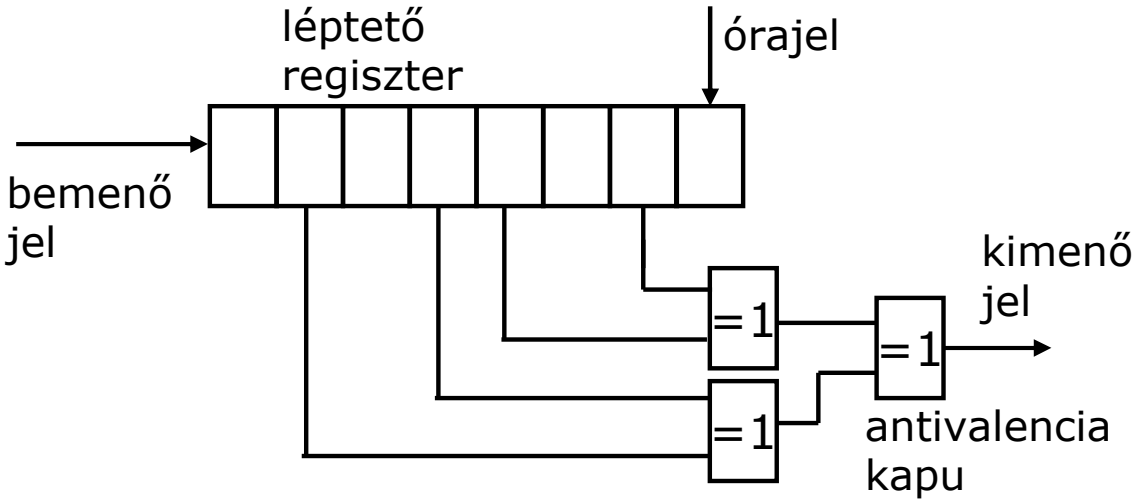
Hibák a szinkron elvesztése miatt

- Bithibát okoz az adó és a vevő közötti bitszinkron elvesztése
  - a vevő periódikusan mintavételezi a vett analóg jelet, tipikusan jelátmeneteket keres
  - a jelátmeneteket felhasználva előállítja a vételi órajelet
  - ha a vett jelben túl kevés az átmenet, az előállított órajel elcsúszik az adó órajeléhez képest, így a vevő nem a megfelelő időpontban mintavételez
  - ez hibacsomósodást okoz

Bitkeverés a szinkronvesztés megelőzésére

- A szinkron elvesztésének megelőzésére az adónak kellő számú átmenetet kell produkálnia
- Erre láttunk kódolási technikákat (pl. Manchester-kód)
- Más módszer: az adó járulékos átmeneteket hoz létre az adatok módosításával
- Ezt a módosítást a vevőnek vissza kell állítania
- A módosítást leginkább bitkeveréssel (*scrambling*) végezzük:
  - a bemeneti adatot egy léptetőregiszterbe léptetjük be
  - a regiszter különböző mélységű bitjeiről adatokat csatolunk ki, ezek *kizáró vagy* kapcsolata lesz az adó kimeneti jele
  - a vételi oldalon egy azonos felépítésű áramkörrel állítjuk vissza az eredeti adatokat

A bitkeverés technikája



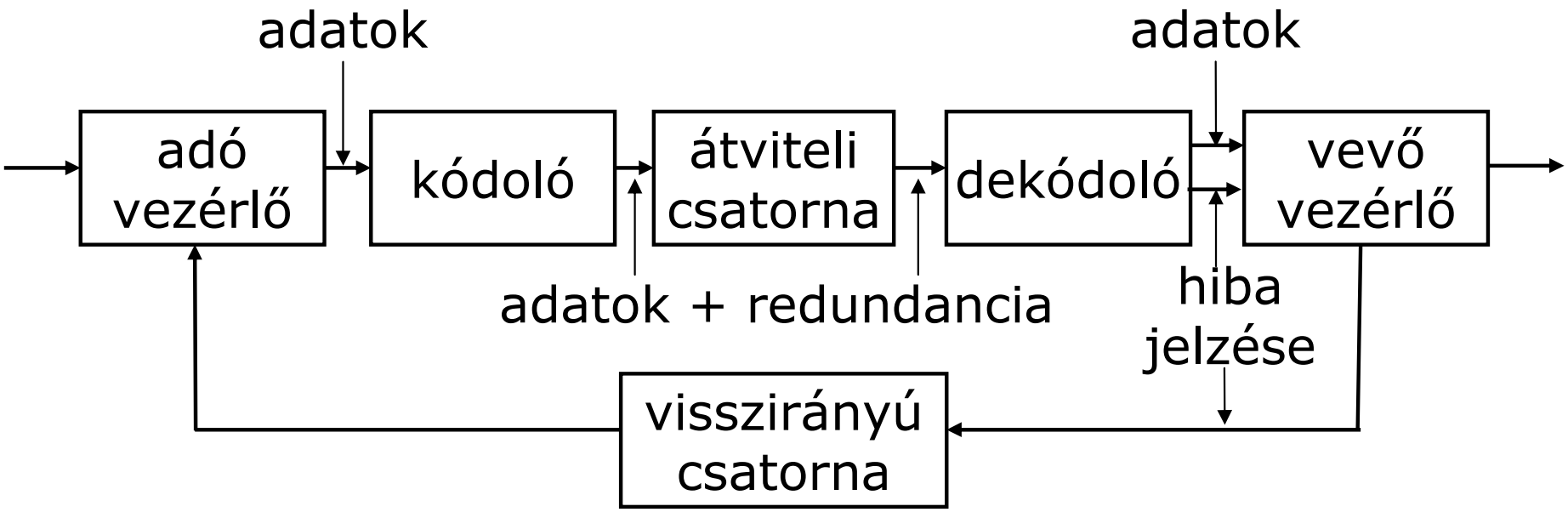
- A bitkeverők nem adnak tökéletes megoldást:
  - minden konfigurációhoz létezik olyan bemeneti jelsorozat, amely csupa 0 vagy csupa 1 kimenetet eredményez
  - egy hibásan vett bitet a bitkeverő annyiszor ismétel meg, amennyi a kicsatolások száma

Hibajavítási stratégiák

- Bithibák javítására két lehetőség van:
  - visszirányú hibajavítás (*Backward Error Correction*), leggyakoribb megoldás: automatikus ismétléskérés (*Automatic Repeat Request, ARQ*)
  - megelőző hibajavítás (*Forward Error Correction, FEC*)

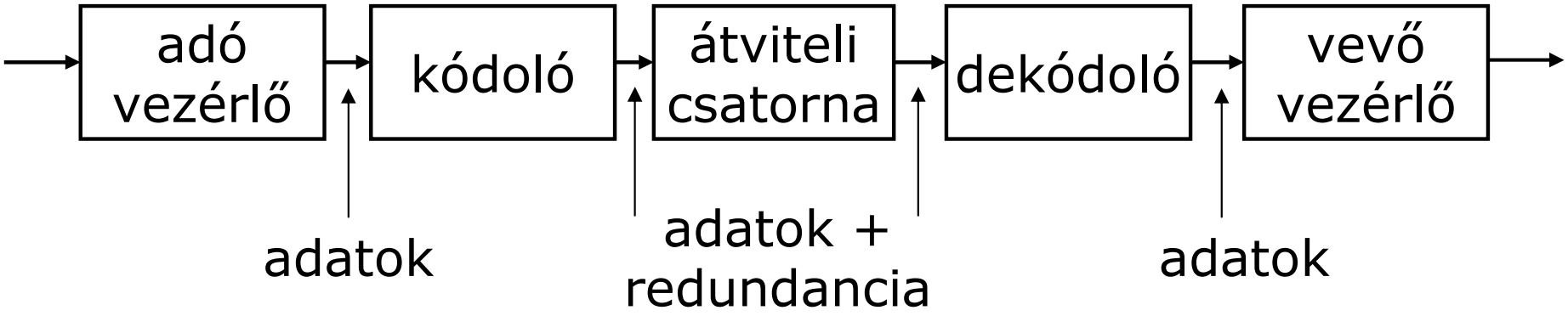
    ~ hibajavító kódolás
- Automatikus ismétléskérés esetén a vevő észleli a hibát és ismétlést kér az adótól
- Megelőző hibajavítás esetén a hiba a vett információ alapján a vétel helyén kijavítható
- Mindkét módszer alapja redundáns információ továbbítása a tényleges adatokkal együtt, a hiba észlelése/javítása céljából

Visszirányú hibajavítás (Backward Error Correction)



- Az adatcsomagokat a küldő *hibajelző kódolással* látja el
- A vevő dekódolja a vett csomagokat
- Minden hibásnak talált csomagot eldob és a visszirányú csatornán kéri újbóli elküldését

Megelőző hibajavítás (Forward Error Correction)



- A kódoló az adatokat redundáns információval egészíti ki, ez a *hibafelfedést és hibajavítást* szolgálja
- A dekódoló dekódolja a vett információt, megállapítja hogy történt-e hiba, és javítja, ha igen
- Ára, hogy a redundáns kiegészítés mérete jelentős, az adatok méretével összehasonlítható lehet

Bithibák felfedése és javítása

- Azokat az eljárásokat, amelyekkel redundáns információt adunk az adatokhoz, hibakezelő kódolásnak (*Error Control Coding, ECC*) nevezzük
- Két alapvető hibakezelő kódolási módszer:
  - blokk-kódolás
  - konvolúciós kódolás
- *A hibajavító kódolás alapvető módszereivel a Kódolástechnika c. tárgyban foglalkoztak*

Bit- és csomaghibák kezelése

- A bitszintű és a csomagszintű hibakezelés módszerei egymásra épülnek:
  - bithibák érzékelésére/javítására redundanciát használnak
  - csomaghibák kezelésére kihasználják a bit- valamint a csomagszintű hibadetektálás módszereit, hogy a vevőnél azonosítsák és figyelmen kívül hagyják a hibás csomagokat, vagy
  - megismételtessék azokat az adóval
- A vevő vagy *pozitív nyugtát (ACK - acknowledgement)* küld minden hibátlanul vett csomagról az adónak, vagy *negatív nyugtát (NACK – negative ACK)* küld a hibásan vett csomagokról

Csomaghibák fajtái

- A csomagszintű hibák:
  - csomagvesztés
  - csomagok többszöröződése
  - csomagbeszúrás
  - csomagok átsorrendeződése

A csomagvesztés okai

- A javíthatatlan (mert sokszoros, burst-ös) bithibák miatti csomagvesztés a vezetéknélküli hálózatokban gyakori (akár 40%-os is lehet)
- Vezetékes hálózatokban a csomagvesztés oka inkább a hálózati csomópontok időleges túlterhelése
- Ez jelentősen függ a terhelés jellegétől:
  - egyenletes terhelés esetén a túlterhelés ritka, a csomagvesztés aránya alacsony
  - ingadozó, lökésszerű terhelés esetén a csomagvesztés aránya nő

Csomagok többszöröződése

- A csomagok ismételt elküldése a csomagvesztés általános javítási módja
- Az adó a csomag elküldése után bizonyos ideig várakozik a vétel nyugtázására, és ha az nem érkezik meg, ismét elküldi a csomagot
- Lehet, hogy a csomag már először is hibátlanul megérkezett, csak erről az adó nem értesült, mert
  - a nyugta elveszett, vagy
  - csak a várakozási idő letelte után érkezik meg
- Mivel az adó már megismételte a csomagját, csomagtöbbszöröződés lép fel

Átsorrendeződés és beszúrás

- Ugyanazon kapcsolathoz tartozó csomagok sorrendje felborulhat, ha azok különböző útvonalakon érkeznek a vevőhöz
- Egyes útvonal-irányító eljárások a vett csomagokat több irányba küldik tovább, hogy a hálózat átbocsátóképességét jobban kihasználják
- Ilyenkor a vevőhöz a különféle útvonalak miatt az eredetitől eltérő sorrendben érkezhetnek a csomagok
- Csomagbeszúrás történhet akkor, amikor a csomag fejrésze észlelhetetlenül hibásodik meg:
  - egy vevő az 1-es és 2-es VCI-n vár csomagokat
  - egy csomag fejrészében a 2-es VCI észlelhetetlenül 1-re változik
  - ez az 1-es VCI-nél egy beszúrt csomagot eredményez
  - *(VCI: virtuális csatorna-azonosító, helyi érvényű cím összeköttetés-alapú csomagkommunikációnál)*

Csomaghibák észlelése: sorszámozás

- A küldő minden csomag fejrészébe beilleszt egy csomagonként növekvő sorszámot, ennek segítségével
  - felismerjük az átsorrendeződést és a csomagtöbbszöröződést
  - a csomagvesztés hézagként jelentkezik a sorszámokban
  - a csomagbeszúrás is kiderülhet (feltéve, hogy a beszúrt csomag sorszáma jelentősen eltér a hibátlan csomagok aktuális sorszámaitól)
- A csomagok sorszámai korlátos méretűek, ezért kellően sok csomag továbbítása esetén körülfordulhatnak

Alsó korlát a sorszám méretére *

- A sorszám a csomag fejének része, mérete – különösen rövid csomagok esetén - kihat az átviteli kapacitás kihasználtságára
- A sorszám *n*-nel jelölt hosszát (*bitekben*) befolyásolja:
  - a csomag maximális élettartama (*Maximum Packet Lifetime, MPL*), az az idő, ameddig egy csomag létezhet a hálózaton, [s]
  - az a *Tmax* idő, amelyen belül egy nyugtára várakozó küldő állomás ismétli a csomagokat, [s]
  - a leghosszabb *A* idő egy csomag vétele és a rá vonatkozó nyugta elküldése között, [s]
  - a küldő állomás *R* csomagtovábbítási sebessége, [csomag/s]
- Ezekkel az adatokkal $2^n \geq (2MPL+Tmax+A)R$

Alsó korlát a sorszám méretére (folyt.) *

- **Hogy jön ki az, hogy $2^n \geq (2MPL+Tmax+A)R$?**
- **Az adó *Tmax* ideig folytatja a csomagja ismétlését az első kísérlettől kezdve**
- **Legrosszabb esetben a vevő közvetlenül a csomag élettartamának lejárta előtt, tehát *Tmax+MPL* idő elteltével kapja meg azt**
- ***A* idő elteltével küldi a nyugtát, az legfeljebb *MPL* idő múlva érkezik meg az adóhoz**
- **Ezalatt az adó legfeljebb *(2MPL+Tmax+A)R* csomagot, így sorszámot generálhatott**
- **Ez persze jó nagy sorszámkorlátot ad, példa:**

PÉLDA
Legyen MPL=120 s, Tmax=60 s, A=0,5 s,
a csomagméret 40 byte, így 2 Mbit/s-es adatsebesség esetén
$R=2*10^6/40*8$ csomag/s
Ekkor n≥21

A csomagvesztés adó általi észlelése

- A csomagvesztés észlelése történhet:
  - időzítéssel (timeout)
  - negatív nyugtázással (negative acknowledgement)
- Időzítés alkalmazása esetén a küldő állomás minden csomag elküldésekor beállít egy időzítőt
- Az időzítő értéke az idő előrehaladtával automatikusan csökken
- Ha az időzítő lejártáig (nulla értékéig) nem érkezik nyugta a csomagra (pozitív nyugtázás), akkor a küldő feltételezi, hogy a csomag vagy a nyugta elveszett és ismét elküldi a csomagot

Csomagvesztés észlelése: időzítés

- Kritikus pont az időzítés mértéke:
  - túl kis érték esetén sok felesleges ismétlés jelentkezik
  - túl nagy érték esetén lassú, vontatott lesz a hibajavítás folyamata
- Az időzítés mértéke lehet:
  - rögzített, általában a teljes körülfordulási idő (Round Trip Time, RTT) 1-2-szerese
  - változó, a mért RTT értéktől illetve annak valamilyen átlagától függő

Csomagvesztés észlelése: negatív nyugtázás

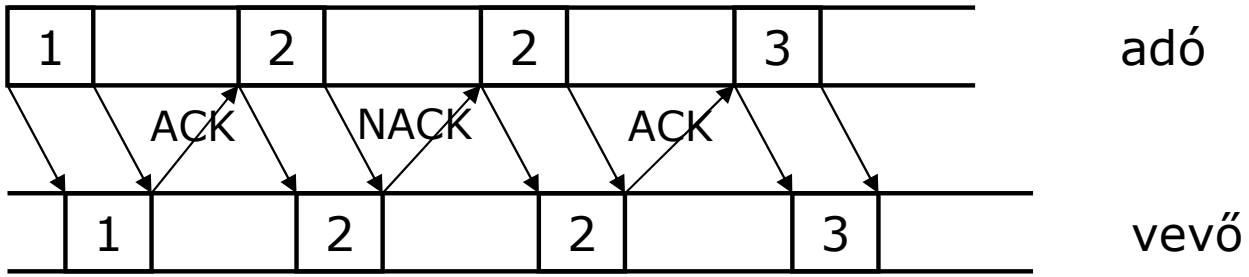
- Az időzítés helyett alkalmazott negatív nyugtázás esetén a vevő értesíti a küldőt a csomagvesztésről, negatív nyugta, NACK küldésével
- A vevő NACK-ot küld, ha hibás csomagot vagy hézagot észlel a vett csomagok sorszámában
- A NACK több hibás/hiányzó csomag sorszámát is tartalmazhatja
- NACK vételekor a küldő ismételten elküldi a kért csomagokat
- A negatív nyugtázás problémái:
  - **csomagvesztés általában torlódáskor lép fel, az ilyenkor küldött NACK csomagok csak növelik a hálózat terhelését**
  - **a NACK is elveszhet, ilyenkor a vevőnek kell ismételnie, de ehhez a vevőnél kell időzítést alkalmazni**

Csomagismétlési módszerek

- Az automatikus ismétléskérés (ARQ) körébe tartozó módszer
- Csomagvesztésnél a csomagismétlési stratégia dönt arról, hogy melyik csomagot kell ismét elküldeni
- Stratégiái:
  - megállás és várakozás (*stop-and-wait*)
  - visszalépés *n*-nel (*go-back-n*)
  - szelektív csomagismétlés (*selective retransmission*)

Megállás és várakozás

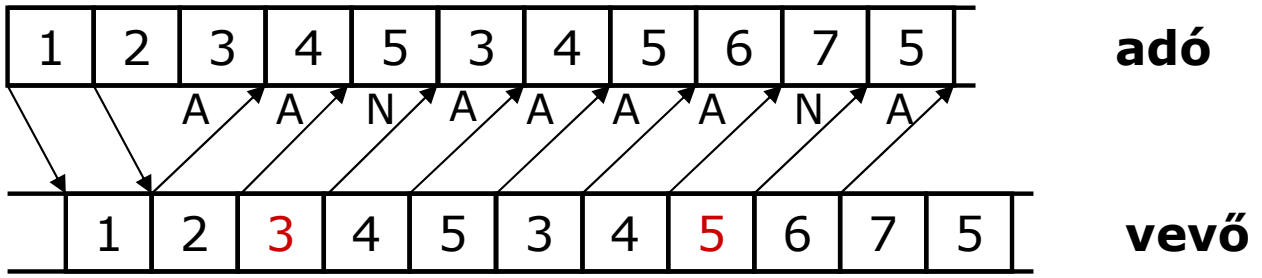
- Az adó minden csomag elküldése után leáll a továbbítással, és a vevőtől érkező nyugtára vár
- A vevő minden csomagot nyugtáz
  - a hibátlanokat pozitív (ACK)
  - a hibásakat/hiányzókat negatív (NACK) nyugtával
- Negatív nyugta vételekor, illetve nyugta hiányában (időzítés) az adó megismétli a csomagot
- Egyszerű, de nem hatékony eljárás
  - adó sokat vár
  - sok a visszairányú csomag



Visszalépés *n*-nel – *go-back-n*

- Ablaktípusú, amire már láttunk példát a forgalomszabályozásnál
- A hibakezelő ablak azon csomagok sorszámainak halmaza, amelyeket az adó már elküldött, de még nem nyugtáztak
  - A hibakezelő ablak legnagyobb mérete rögzített
  - Minden elküldött csomag eggyel növeli az ablak méretét
  - Ha az ablak mérete eléri a rögzített maximumot, az adó leáll
  - Minden pozitív nyugta csökkenti a hibakezelő ablak méretét
- A csomagküldés és az ACK vétele egyaránt eltolja az ablakot a nagyobb csomagsorszámok irányába, ezért ezt csúszó ablakos hibakezelésnek is nevezik
- A visszalépés *n*-nel azt jelenti, hogy amikor egy csomag ismétlését kéri a vevő, az adó a hibakezelő ablakban található összes csomagot ismételten elküldi

Visszalépés n-nel: példa



- **Az adó**
  - ***két ACK-t kap és 5 csomagot küldött el a NACK vétele előtt***
  - ***hibakezelő ablakában tehát a 3, 4, 5 sorszámú csomagok vannak***
  - ***a NACK észlelésekor megismétli az ablakban található összes csomagot***
  - ***a második hibára vonatkozó NACK vételekor a hibakezelő* ablakban az 5, 6, 7 sorszámú csomagok találhatók**
- **A folyamat addig tart, amíg a kívánt (3-as illetve 5-ös) csomag hibátlanul meg nem érkezik, és ez idő alatt a vevő a megismételni kért csomagot követő csomagokat figyelmen kívül hagyja**

Visszalépés n-nel: értékelés

- Biztonságra törekszik, mivel egy csomag hibája esetén is több csomagot ismétel, ezért jobban kezeli a hibacsomókat
- Egyszerűbb, mert a vevőnek nem kell tárolni a már beérkezett csomagokat
- Kihasználtságcsökkenést okoz, mivel többet ismétel a szigorúan szükségesnél
- Ha az eredeti hibát túlterhelés okozta, a fölös ismétlések csak rontanak a helyzeten
- Külső ellenőrzés nélkül ez odáig fokozódhat, hogy a hálózat csak csomagismétlést végez, valódi forgalom nincs, ez a torlódás miatti összeomlás (*congestion collapse*)

A go-back-n hatékonysága

- *p* csomaghiba-valószínűség és *W* maximális hibakezelő ablakméret mellett az adó legfeljebb

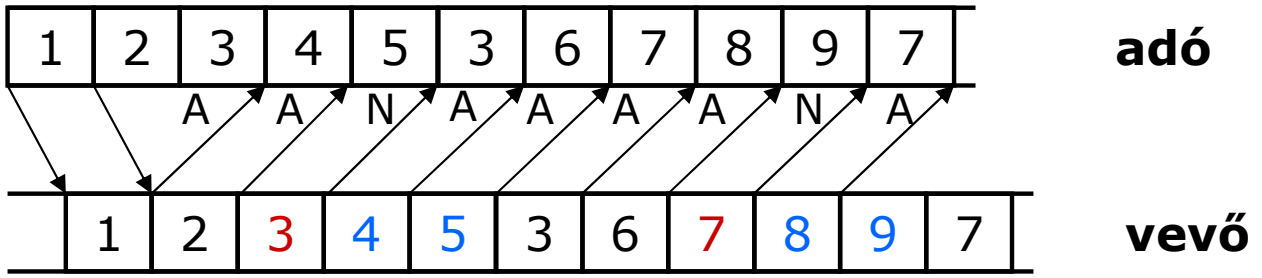
  (1-p)/(1-p+Wp)

  arányban használja a csatornát nem-ismételt csomagok (hasznos adat) továbbítására
- Példa: p=0,01, W=250,

  (1-0,01)/(1-0,01+0,01*250)=0,283.

  Ha $p=10^{-5}$, akkor 0,997-re javul

Szelektív csomagismétlés



- A küldő állomás csak a kért csomagokat ismétli
- A vevő csak a hibásan vett csomagokat dobja el
- A hibásan vett csomag utáni jó csomagokat egy külön pufferben tárolja
- A helyesen vett ismételt csomag után a külön puffer tartalmát bemásolja a vételi pufferbe
- Nagyon hatékony módszer, bonyolultabb algoritmussal és nagy vevőoldali puffermemória-igénnyel

Összefoglalás

- A hibák keletkezésének okai
  - *bit- és csomaghibák*
- Alapvető hibajavítási stratégiák
  - *visszirányú hibajavítás és hibajavító kódolás*
- A hibajavító kódolás módszerei
  - *nem foglalkoztunk ezekkel, a „Kódolástechniká”-ban tárgyalták*
  - *paritáskód, Hamming-kód, CRC, Reed-Solomon-kód, konvolúciós kódolás*
- Csomaghibák kezelése
  - *go-back-n és szelektív ismétlés*

Kommunikáció az adatkapcsolati rétegben

Hálózati modellek: ISO-OSI és a TCP/IP-szerű 5-rétegű

Alkalmazási	(7. Application)
Megjelenítési	(6. Presentation)
Viszonylati	(5. Session)
Szállítási	(4. Transport)
Hálózati	(3. Network)
Adatkapcsolati	(2. Data Link)
Fizikai	(1. Physical)

ISO OSI

Alkalmazási	(Application)
Szállítási	(Transport)
Hálózati	(Internet)
Adatkapcsolati	(Data Link)
Fizikai	(Physical)

TCP/IP-szerű

interface-réteg a TCP/IP-ben

- Az adatkapcsolati réteg:
  - szolgáltatásokat nyújt a hálózati rétegnek
  - használja a fizikai réteg szolgáltatásait
  - nagykiterjedésű hálózatok két szomszédos csomópontja, illetve helyi hálózatok csomópontjai közötti adatátvitelt biztosítja
  - **tartalmazhat hibadetektálási/hibajavítási funkciókat, a fizikai rétegben jelentkező hibák kezelésére**

Adatkapcsolati rétegbeli kommunikáció

- Adatkapcsolati réteg (Data Link Layer):
  - a hétrétegű ISO-OSI és az ötrétegű „kombinált” architektúra második rétege
  - adatokat továbbít egy hálózat két szomszédos csomópontja illetve egy helyi hálózat csomópontjai között
  - biztosítja:
    - az adatátvitel funkcionális és eljárási eszközeit
    - a hibadetektálást, esetleg a hibajavítást a fizikai rétegben keletkező hibákra vonatkozóan
  - egyes architektúrákban két alrétegre oszlik:
    - logikaikapcsolat-vezérlés (Logical Link Control, LLC)
    - közeghozzáférés-vezérlés (Media Access Control, MAC)

Adatkapcsolati rétegbeli protokollok (többségükről nem lesz szó)

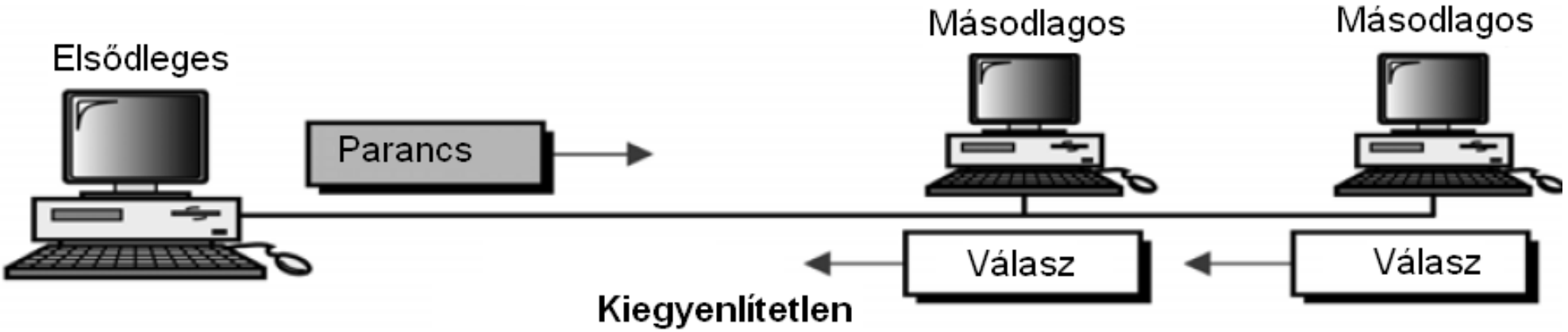
- High-Level Data Link Control, HDLC (Magasszintű adatkapcsolat-vezérlés)
- *Synchronous Data Link Control, SDLC (Szinkron adatkapcsolat-vezérlés)*
- *Link Access Protocol-Balanced, LAPB (Kapcsolathozzáférési protokoll, kiegyenlített)*
- *Link Access Protocol- Frame Mode Services, LAPF (Kapcsolathozzáférési protokoll, kerettovábbítás számára)*
- Message Transfer Part-2, MTP-2 - Üzenetátvivő rész 2: az ITU 7-es jelzésrendszer adatkapcsolati szintű része, nyilvános telefonhálózatokban használják
- Point-to-Point Protocol - PPP

HDLC *

- High Level Data Link Control
  - bitorientált, szinkron adatkapcsolati protokoll
  - szabványos (ISO 13239)
  - az ISO az IBM által definiált SDLC protokoll továbbfejlesztésével definiálta
  - egyaránt támogatja az összeköttetés/alapú és az összeköttetésmentes szolgáltatásokat
- Számos további protokoll alapja a HDLC
- Adategysége a keret (frame)

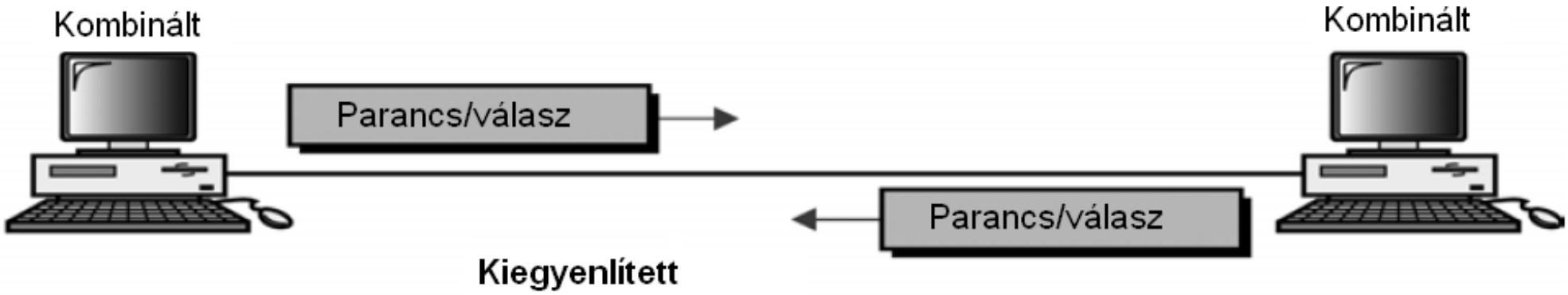
HDLC állomások és konfigurációk (1) *

- A HDLC protokoll a három állomástípusra alapozva háromféle konfigurációt, kiépítést ismer:
  - ***kiegyenlítetlen:***
    - egy elsődleges és egy v. több másodlagos állomásból áll
    - az elsődleges állomás vezérli a többit
    - működhet fél-duplex vagy duplex módban
    - pont-pont közötti vagy pont-többpont hálózatokban



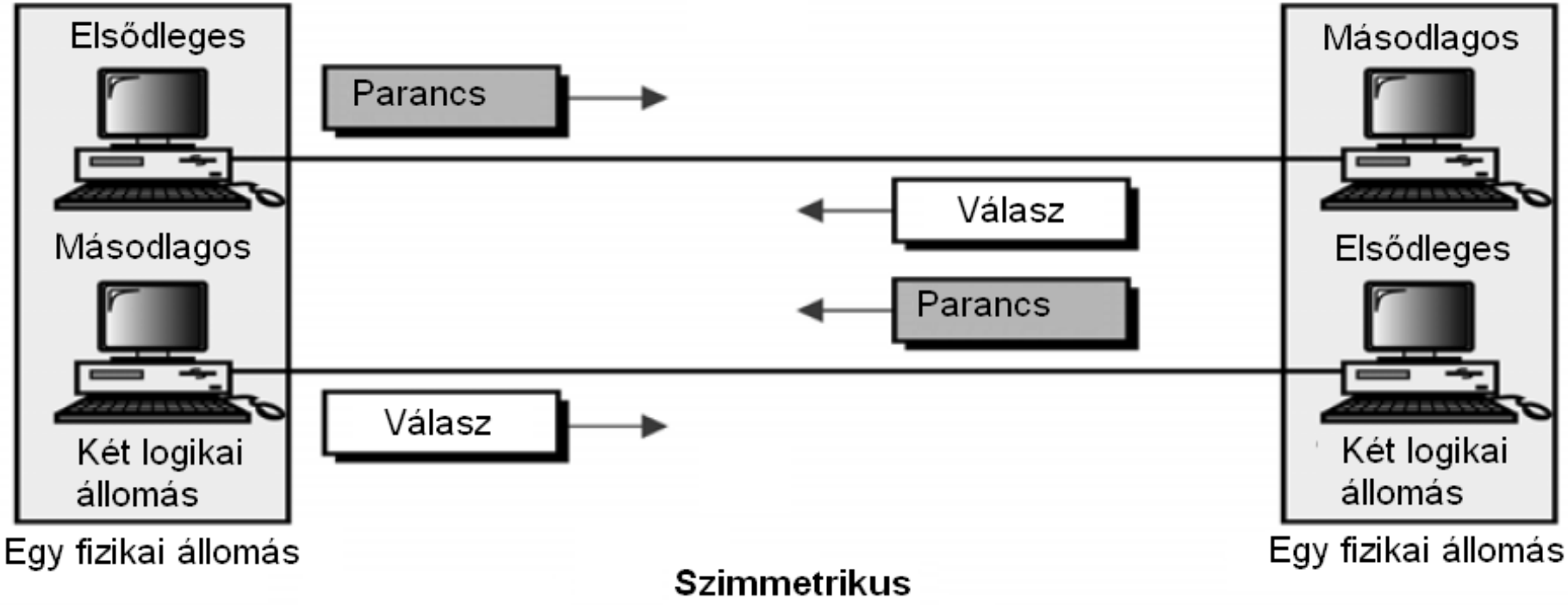
HDLC állomások és konfigurációk (2) *

- *kiegyenlített:*
  - kettő kombinált állomásból áll
  - az állomások egyenrangúak
  - fél-duplex v. duplex módban működhet
  - pont-pont közötti kapcsolaton



HDLC állomások és konfigurációk (3) *

- ***szimmetrikus:***
  - minden fizikai állomás két logikai állomásból áll, az egyik elsődleges a másik másodlagos
  - külön vonal köti össze a logikai állomáspárokat
  - ritkán alkalmazott konfiguráció



A HDLC „működési módjai” (1) *

- A működési mód:
  - két, adatcserét folytató állomás közötti viszony
  - meghatározza, hogy ki vezérli az adatkapcsolatot
- Három működési mód:
  - normálválasz-mód
    (Normal Response Mode, NRM)
  - aszinkronválasz-mód
    (Asynchronous Response Mode, ARM)
  - aszinkron kiegyenlített mód
    (Asynchronous Balanced Mode, ABM)

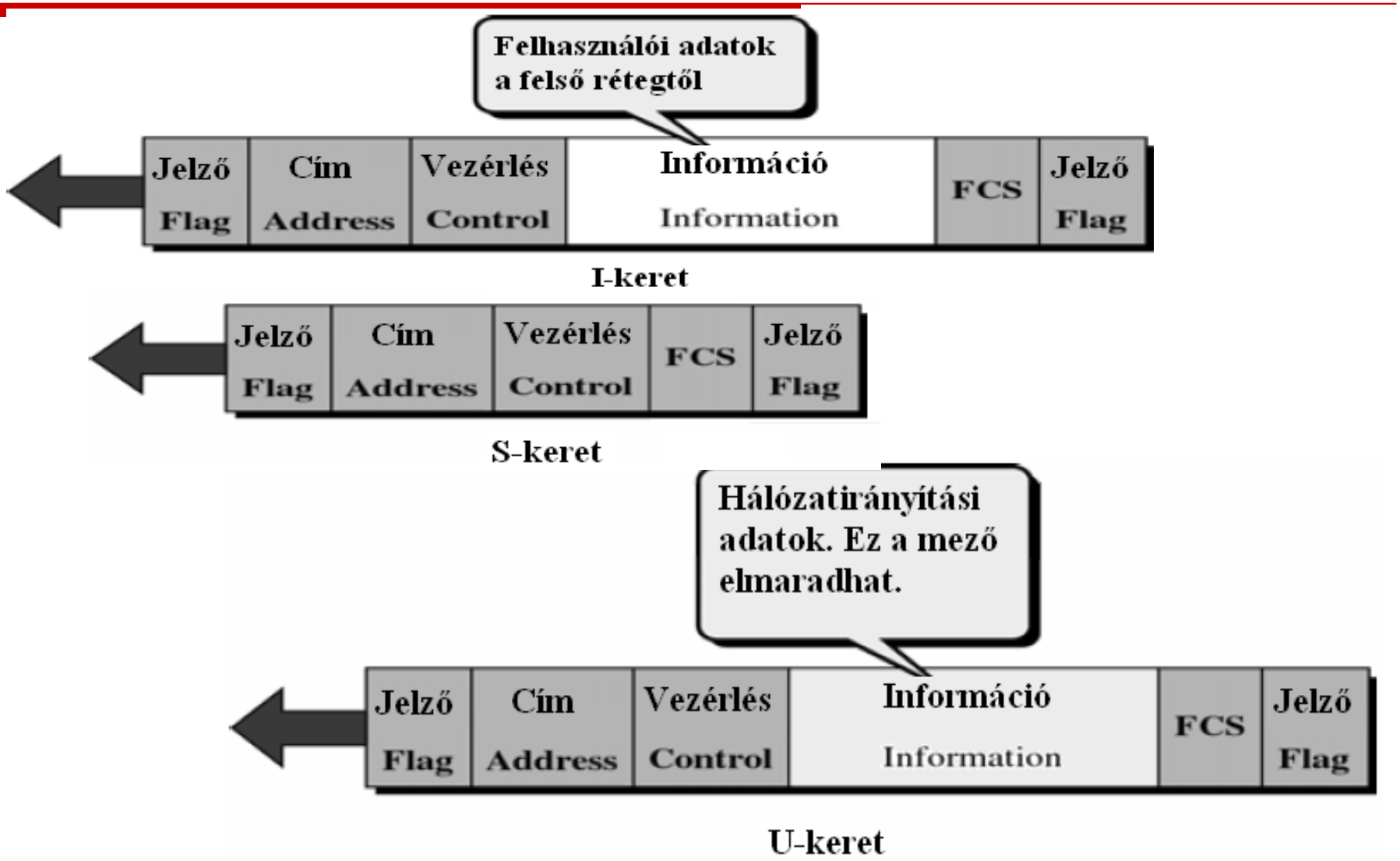
A HDLC „működési módjai” (2) *

- NRM: a szabályos elsődleges-másodlagos viszony megvalósulása
  - a másodlagos állomás csak az elsődleges engedélyével adhat
  - Az engedély birtokában a másodlagos állomás egy v. több, adatot tartalmazó keretből álló válasz továbbítását kezdeményezheti
- ARM: a másodlagos állomás az elsődleges engedélye nélkül maga is kezdeményezhet átvitelt, ha a vonal tétlen
  - Az elsődleges állomástól minden üzenet még mindig a másodlagoshoz jut, és csak azután kerülhet további eszközökhöz
- ABM: minden állomás egyenrangú $\Rightarrow$ csak kombinált állomások között, pont-pont kapcsolatban
  - Bármelyik kombinált állomás kezdeményezhet átvitelt a másik engedélye nélkül

HDLC keretek *

- Háromféle keret:
  - információs (Information, I), felhasználói adatok, illetve ezekre vonatkozó vezérlési információk továbbítására
  - felügyeleti (Supervisory, S) az átvitel vezérlését szolgáló információkhoz
  - számozatlan (Unnumbered, U), a rendszermenedzsment-információk továbbítása számára fenntartva

HDLC keretek *



HDLC keretek mezői: a *jelző* *

- Jelző (Flag):
  - minden keret elején és végén
  - rögzített mintázat (01111110)
  - a vevő (keret)szinkronizálására szolgál
  - ez a minta az adatok között **nem** fordulhat elő:
    - 5 db egymásutáni 1-es adatbit után az adó automatikusan beszúr egy 0-t
    - a vevő az 5 egyes utáni 0-t automatikusan törli
      - adat: 0111111110 ⇒ kódolt: 011111**0**1110 ⇒ dekódolt: 0111111110
    - ez biztosítja a HDLC kód átláthatóságát (transzparencia), vagyis azt, hogy tetszőleges adatbit-sorozat továbbítható
    - a keretek közötti szünetben jelzők folytonos továbbítása történik

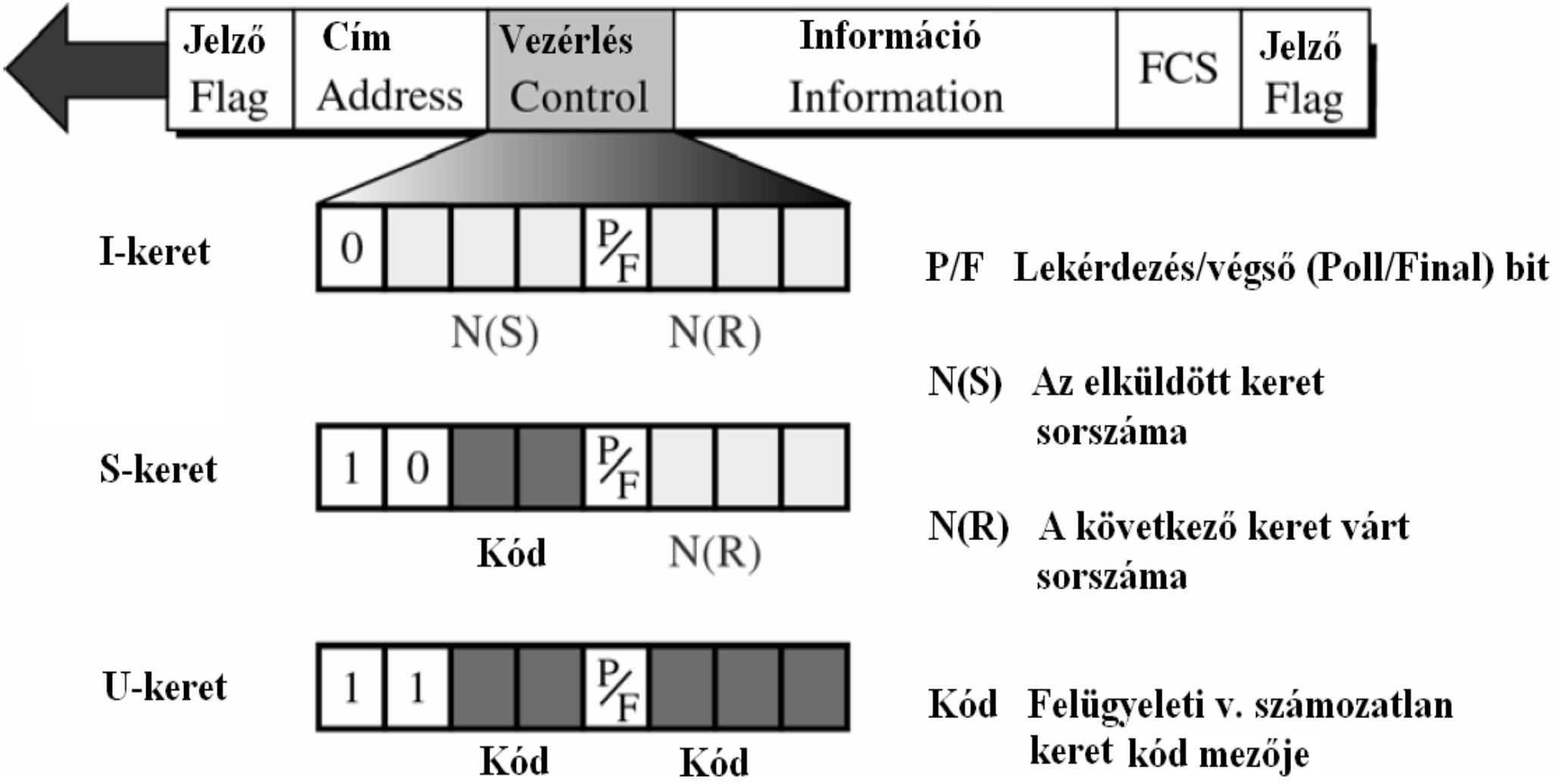
HDLC keretek mezői: a *jelző* *

- Két további bitsorozatnak van még rögzített jelentősége:
  - 7 v. több, de 15-nél kevesebb egymás utáni 1-es bit jelzi a forgalom megszakítását (abort)
    - ez akkor lehet szükséges, ha egy magasabb prioritású keretet kell sürgősen továbbítani
  - 15 vagy több egymásutáni 1-es jelzi a tétlen csatornát
    - a HDLC vevő tétlen állapotában egy keret kezdetére vár
    - folytonosan figyeli a vonalat, jelzőt keres
    - szokás még „hunt mode”-nak is nevezni

HDLC keretek mezői: a *cím* *

- Címmező:
  - 8 bit v. annak többszöröse, a hálózattól függően
  - 8 bites cím utolsó bitje mindig 1-es
  - több byte-os címnél az utolsó byte utolsó bitje 1, a többié 0
  - minden állomásnak egyedi címe van
  - kiegyenlítetlen konfigurációban a cím a parancs és a válaszkeretben is a másodlagos állomásra vonatkozik
  - kiegyenlített konfigurációban a parancskeretben a célállomás címét, a válaszkeretben a küldő állomás címét tartalmazza
  - a cím első bitje jelzi, hogy egy (=0) vagy több (=1) állomásra vonatkozik

HDLC keretek mezői: a *vezérlés* *



HDLC keretek mezői: a *vezérlés* *

- Vezérlés-mező:
  - meghatározza a keret típusát
  - „I”-keretben végfelhasználói információt szállít a két állomás között
  - felügyeleti („S”) keretben vezérlő funkciókat lát el:
    - keret nyugtázása
    - keretismétlés-kérés
    - kerettovábbítás időleges felfüggesztésének kérése
  - számozatlan („U”) keretben kapcsolatvezérlési funkciók:
    - kapcsolat inicializálása
    - kapcsolat bontása
    - egyéb kapcsolatvezérlő funkciók

HDLC keretek mezői: a *vezérlés* *

- P/F (Poll/Final) bit:
  - csak akkor van jelentése, ha értéke 1
  - ha az elsődleges állomás küldi a másodlagosnak (parancs), akkor állapotlekérdezést (Poll) jelent
  - ha a másodlagos küldi az elsődlegesnek (válasz), akkor az állapotot tartalmazó keretet jelzi
  - normálválasz-módban a másodlagos állomás ezzel jelezheti adásának végét is (Final)

HDLC keretek mezői: a *kód* *

- Kódmező értelmezése felügyeleti („S”) keretben:
  - hibakezelés és áramlásvezérlés
    - 00: vevő készenlétben (Receive Ready, RR), az N(R)-1 számú keret pozitív nyugtázása és az N(R)-edik keret vételére a készség jelzése
    - 01: visszautasítás (Reject, REJ), n-nel történő visszalépés kérése, a vevő kéri N(R)-től a keretek ismétlését
    - 10:vevő nincs készenlétben, (Receive Not Ready, RNR), az R(N)-1 számú keret pozitív nyugtázása és a további keretek küldésének időleges felfüggesztésének kérése
    - 11: szelektív visszautasítás (Selective Reject, SREJ), szelektív ismétlés kérése, a vevő kéri az N(R) számú keret ismétlését

HDLC keretek mezői: a *kód* *

- Kódmező értelmezése számozatlan („U”) keretben:
  - az adatkapcsolati vezérlő funkciókat tartalmazzák
  - néhány példa
    - SNRM: normál válasz mód beállítása
    - SARM: aszinkron válasz mód beállítása
    - SABM: kiegyenlített válasz mód beállítása
    - DISC: kapcsolat bontása
    - UA: számozatlan nyugtázás
    - FRMR: keret visszautasítása

HDLC keretek mezői: az *információ* *

- Információ mező:
  - nincs minden keretben, csak az információs és a számozatlan keretben található
  - a küldő által a fogadónak küldött tényleges adatbiteket tartalmazza
  - mérete hálózatfüggő, de egy adott hálózaton belül állandó
  - lehetőség van arra, hogy egy „I” keretben az adatbitek mellett vezérlési információt is továbbítsunk, ez a „hátonvitt” (piggyback) módszer

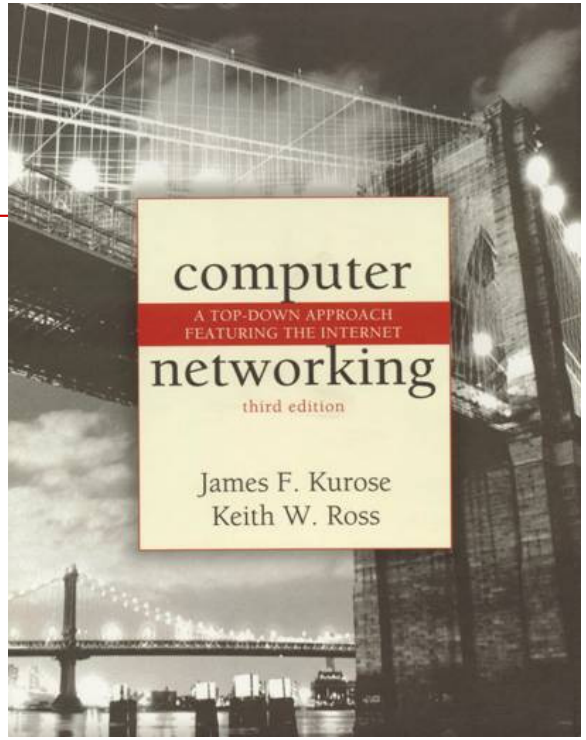
HDLC keretek mezői: az *ellenőrző összeg* (FCS) *

- Keretellenőrző összeg (Frame Check Sequence, FCS) mező:
  - a HDLC hibadetektálást szolgáló mezője
  - a cím-, vezérlési és információs mezőkre vonatkozik
  - 16 vagy 32 bites CRC lehetséges
  - FCS [16 bit] = $X^{16} + X^{12} + X^5 + 1$
  - FCS [32 bit] = $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$

Point-to-Point Protocol

PPP

Slide-ok a Kurose-Ross könyvből



Computer Networking: A Top Down Approach Featuring the Internet, 3rd edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2004.

PPP Design Requirements [RFC 1557]

- packet framing: encapsulation of network-layer datagram in data link frame
  - carry network layer data of any network layer protocol (not just IP) *at same time*
  - ability to demultiplex upwards
- bit transparency: must carry any bit pattern in the data field
- error detection (no correction)
- connection liveness: detect, signal link failure to network layer
- network layer address negotiation: endpoint can learn/configure each other's network address

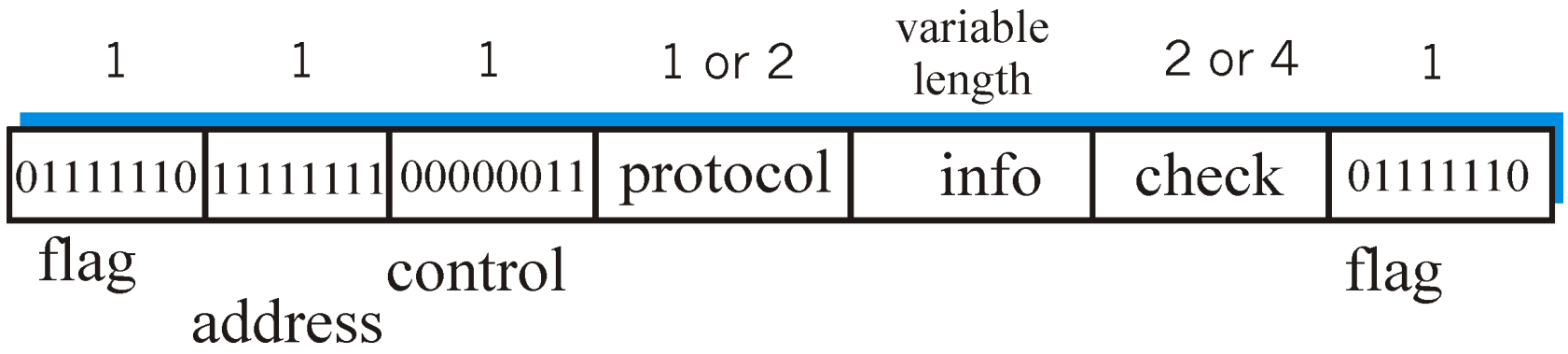
PPP non-requirements

- no error correction/recovery
- no flow control
- out of order delivery OK
- no need to support multipoint links (e.g., polling)

Error recovery, flow control, data re-ordering
all relegated to higher layers!

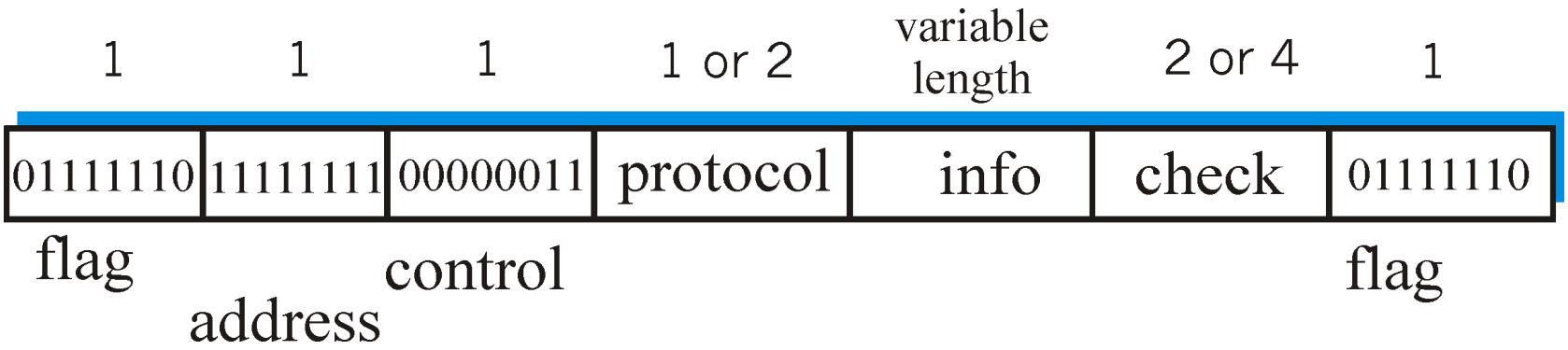
PPP Data Frame

- Flag: delimiter (framing)
- Address: does nothing (only one option)
- Control: does nothing; in the future possible multiple control fields
- Protocol: upper layer protocol to which frame delivered (e.g., PPP-LCP, IP, IPCP, etc.)



PPP Data Frame

- info: upper layer data being carried
- check: cyclic redundancy check for error detection

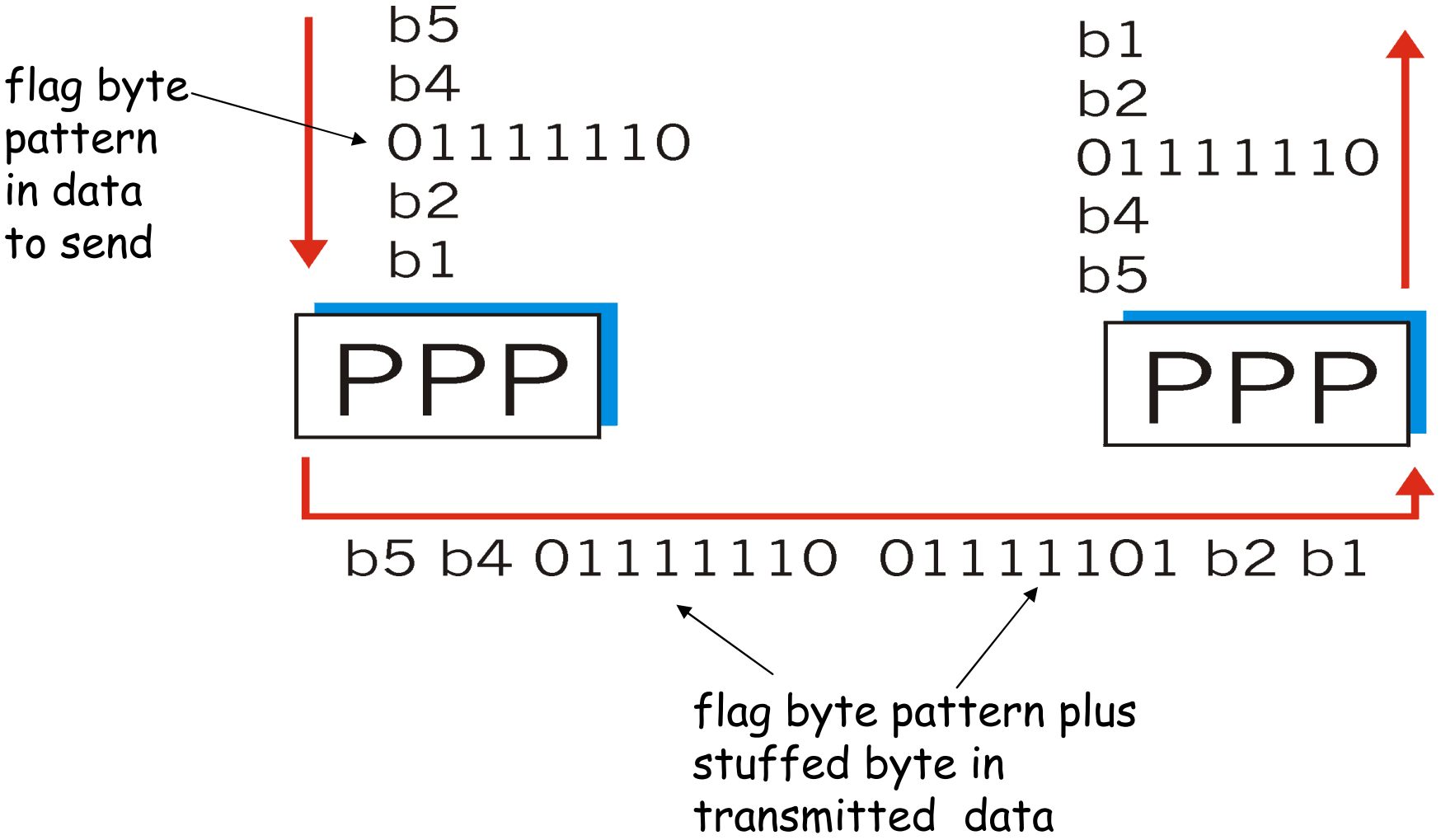


Byte Stuffing

- "data transparency" requirement: data field must be allowed to include flag pattern <01111110>
  - Q: is received <01111110> data or flag?

- Sender: adds ("stuffs") extra < 01111110> byte after each < 01111110> *data* byte
- Receiver:
  - two 01111110 bytes in a row: discard first byte, continue data reception
  - single 01111110: flag byte

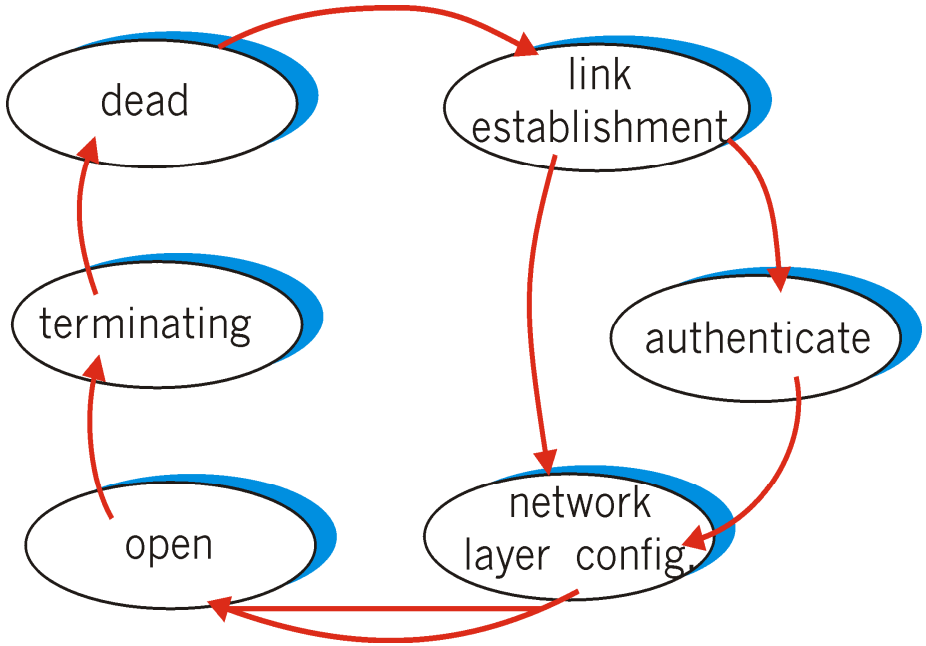
Byte Stuffing



PPP Link Control Protocol

Before exchanging network-layer data, data link peers must

- configure PPP link (max. frame length, authentication)
- learn/configure network layer information
  - for IP: carry IP Control Protocol (IPCP) msgs (protocol field: 8021) to configure/learn IP address



PPP a gyakorlatban

- Főként kapcsolatfelépítésre használják (hitelesítés is ebbe ágyazva)
- Áramkörkapcsolt (pont-pont) hálózatokban
  - nem kell címzés (vezérlési sík kezeli)
  - nem kell sorrendhelyes átvitel (eleve garantált)
  - Pl.: modemes (dial-up) kapcsolat
- PPPoE – PPP over Ethernet
  - többnyire ADSL vagy WiFi kapcsolat fölött a „központtal” egy pont-pont kapcsolat felépítése hitelesítéssel

Összefoglalás

- Két fontos további elvi kérdésről ill. funkcióról volt szó:
  - forgalomszabályozás
  - hibakezelés
- és egy ponton kissé bővítettük a hálózatokkal kapcsolatos ismereteinket:
  - adatkapcsolati rétegbeli protokollok
- Mivel a hálózati réteget már tárgyaltuk, specifikusan az IP-t, ezért legközelebb:
  - szállítási (transzport-) rétegbeli protokollok