10. fejezet – Az adatkapcsolati réteg

Az adatkapcsolati réteg (Data Link Layer)

Előzetesen összefoglalva, az adatkapcsolati réteg feladata abban áll, hogy biztosítsa azt, hogy az adó oldali adatok a vevő oldalra is adatként jussanak el, és ne legyen belőle értelmetlen jelek sorozata. Ezt úgy valósítja meg, hogy az adatokat egyértelműen azonosítható adatkeretbe tördeli szét, ellátja a szükséges vezérlőbitekkel, majd sorrendben továbbítja azokat. A vevő oldal pedig a kapott kereteket megfelelő sorrendben összeállítja. Az adó oldal ezenkívül még a vevő által küldött nyugtázásokat is feldolgozza. Mivel a fizikai réteg a biteket értelmezés nélkül továbbítja, ezért az adatkapcsolati réteg feladata, hogy meghatározza, illetve felismerje a keretek határait. Így a felette elhelyezkedő réteg már hibáktól mentes adatokat kap (vétel esetén – adás esetén nyilván hibátlan adatokat ad...)
Másik fontos feladata az, hogy a kétirányú átvitel esetén az esetleges ütközésekből adódó problémákat megoldja, és hogy forgalomszabályozást végezzen – tájékoztassa az adót a vevő fogadási szándékáról.

Ha kiragadjuk a Hálózati–Adatkapcsolati–Fizikai rétegeket, és csak ezek működésére koncentrálunk, akkor a működés a következő féle képpen modellezhető. Az adó oldal a hálózati rétegből kapott bitfolyamot az adatkapcsolati réteg diszkrét keretekké alakítja, melyeket ellenőrző összegekkel lát el. Ezeket a kereteket alakítja bitfolyamból jelfolyammá a fizikai réteg, majd továbbítja a célállomás fizikai rétegének. A vevő oldal fizikai rétege fogadja, majd jelfolyamból bitfolyammá alakítja az adatokat. A vevő oldal adatkapcsolati rétege a keretek behatárolása és az ellenőrző összegek visszaellenőrzése után az így keletkezett bitfolyamot továbbítja a hálózati rétegnek.
A fenti elméleti leíráshoz képest, a valóságban a hálózati réteg esetében említett bitfolyam sem egy végtelen hosszúságú entitás, hanem meghatározott méretű (a keret méretének sokszorta nagyobb méretű) csomagok sorozata.
Ahhoz, hogy az adatkapcsolati réteg szolgáltatást nyújthasson a hálózati rétegnek, a fizikai réteg szolgáltatásait kell igénybe vennie.

Az adatkapcsolati réteg legjellemzőbb feladatainak felsorolása:

- hálózati rétegnek nyújtott szolgáltatás
    - nyugtázatlan összekötés nélküli szolgálat
      ilyen a megbízható csatorna alaphelyzetben az Ethernet vagy a kéretlen levél (nincs előzetes kapcsolat felépítés, kapcsolat lebontás)

    - nyugtázott összekötés nélküli szolgálat
      ilyen a megbízhatatlan csatorna, a WLAN vagy a szöveges üzenetküldés
      (nincs előzetes kapcsolatépítés, de mivel minden keret nyugtázva van, az elveszett kereteket meg lehet ismételni)
    - nyugtázott összekötés alapú szolgálat
      ez a legmegbízhatóbb átvitel, például a fájl átvitel)
      (a forrás és a cél az első fázisban felépíti az összeköttetést – inicializálódnak a megfelelő változók, számlálók, a második fázisban történik a keretek átvitele, a harmadik fázisban pedig az összeköttetés lebontása – és az erőforrások felszabadítása)
- keretezés (kezdet, vég)
    - karakterszámlálás, bájtszámlálás
    - kezdő és végkarakterek beszúrása
    - kezdő és végbitek beszúrása
    - fizikai rétegbeli kódolás megsértése
- hibakezelés, hibavédelem (Error Control)
    - pontosan egyszeri megérkezés (időzítők, számlálók kezelése)
    - ismétléssel történő javítás
- forgalom szabályozás (Flow Control)
    - gyors adó, lassú vevő helyzet kezelése

Keretezés

Az adatkapcsolati réteg legtipikusabb feladata a keretezés (nyilván adó oldalon a keretekbe tördelés, vevő oldalon a keretek eltávolítása – jellemzően az adó oldali funkciókat tárgyaljuk részletesen). A réteg alapegysége az adat keret (Data Frame).
Önmagában a keretekre tördelés nem megoldás, szükség van (például) egy ellenőrző összegre, ami megmutatja, hogy az adott keret sérülésmentesen érkezett-e meg a vevő oldalra. Amennyiben az ellenőrző összegben eltérés van, akkor az egész keretet meg kell ismételni.

A négy legáltalánosabban használt keretezési módszer:

- karakterszámlálás, bájtszámlálás
- kezdő és végkarakterek beszúrása
- kezdő és végbitek beszúrása
- fizikai rétegbeli kódolás megsértése

Karakterszámlálás, bájtszámlálás

Ez a keretezési módszer a keret hosszának, azaz a benne foglalt bájtok számának megfelelő adatot írja bele egy fejlécbe, az úgynevezett bájtszámmezőbe. Amikor a vevő oldal adatkapcsolati rétege megkapja az így képzett keretet, a bájtszámmezőből kiolvassa a keret hosszát (azaz bájtok számát).
Ennek az algoritmusnak az a (potenciális) hibája, hogy az átviteli hiba szerencsétlen esetben pont a bájtszám mezőt érintheti, azaz ronthatja el. Így az átvitel kiesik a szinkronból, képtelenség lesz megtalálni a következő keret elejét (illetve végét).
Ez a módszer ma már jellemzően nincs használatban.

Kezdő és végkarakterek beszúrása

Az előző megoldás szinkronizációs hibáját például elkerülhetjük olyan módon, hogy a keret elejét és a végét is egy-egy különleges jelzőbájttal (Flag) látjuk el. Létezett olyan megoldás, ahol a keret elejét illetve végét különböző jelzőbájttal látták el, de a gyakorlatban az egyforma jelzőbájtok használata gyakoribb. Amennyiben az átvitel bármely okból is kiesne a szinkronból, akkor csak meg kell keresnünk a jelzőbájtot, és máris megtaláljuk az éppen átvitel alatt álló keret végét.
Problémát az jelentheti, hogy hiába választunk speciális karaktert, bizonyos tartalmak esetében (pl. bináris átvitel, MP3, MKV, ZIP) a jelzésre választott speciális karakter bitmintája szerepelhet az átvitt adatban is. Egyik megoldás az, hogy egy extra ESC (Escape – kivétel bájt) bájtot használunk, szúrunk be pluszba a jelzőbájtunk elé. Ez a bájtbeszúrás (Byte Stuffing) módszere.
Mi a megoldás arra, hogy ha az ESC az átvitt adatfolyamban is szerepel? Be kell elé szúrni még egy ESC-t, így biztosak lehetünk abban, hogy a két ESC valójában egy ESC tartalommal bír. Ilyen módon abban is biztosak lehetünk, hogy önmagában csak a jelzőbájtunk előtt fog az ESC szerepelni. Az adatfolyamban így csak páros számú egymás utáni ESC bájtok fordulhatnak elő. Így egyértelműen – bár többszörös bájtbeszúrás árán, mert a vevő oldalon majd vissza kell állítani eredeti tartalmat – képesek vagyunk jelezni a keret határait, a keret hosszától függetlenül.

F	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	...	a_{n-7}	a_{n-6}	a_{n-5}	a_{n-4}	a_{n-3}	a_{n-2}	a_{n-1}	a_n	F
ESC	F	a_1	a_2	a_3	a_4	F	a_6	a_7	...	a_{n-6}	a_{n-5}	a_{n-4}	a_{n-3}	F	a_{n-1}	a_n	ESC	F
ESC	F	a_1	a_2	ESC	ESC	a_4	F	a_6	...	a_{n-5}	ESC	ESC	a_{n-3}	F	a_{n-1}	a_n	ESC	F
ESC	F	a_1	a_2	ESC	ESC	ESC	ESC	a_5	...	a_{n-5}	ESC	ESC	a_{n-3}	F	a_{n-1}	a_n	ESC	F

Kezdő és végbitek beszúrása

A bájtbeszúrás hasznos és működő megoldás, de nyilvánvaló, hogy sok „felesleges” adattal terheli az átviteli csatornát. Ezt a problémát orvosolja az eredetileg a HDLC (Highlevel Data Link Control / Magas Szintű Adatkapcsolati Vezérlés) protokollhoz kifejlesztett bitbeszúrás. Ez a módszer lehetővé teszi, hogy tetszőleges számú bit legyen a keretben, sőt, hogy a karakterkódok is tetszőleges számú bitből (ne csak 8 bitből) álljanak.
A megoldás a következő módon épül fel. Minden keret egy speciális bitmintával indul, amit szintén jelzőbájtnak (Flag) nevezünk. Tartalmilag így néz ki: 01111110_2 azaz 6db egymást követő 1-es. Amikor az adó oldalon az adatkapcsolati réteg 5db egymást követő 1-est talál az adatok között, akkor automatikusan beszúr ezek után egy 0-át. Azaz 6db 1-es csakis a Flag-ben fordulhat elő az átalakítás után. Átvitelre már az így átalakított adat kerül. A vevő oldalon ugyanez történik csak fordítva, azaz minden egymást követő 5db 1-es után az adatkapcsolati réteg töröl 1db 0-át.
A bitbeszúrásos módszerrel egyértelműen felismerhetők a kerethatárok. A szinkron elvesztése esetén meg kell keresni a 6db egymást követő 1-est, és így meg is találjuk a kerethatárokat, hiszen a 6db 1-es csak ott fordulhat elő.
Ezt az eljárást használja az USB technológia is. Ezzel a módszerrel a „feleslegesen” átvitt adatok mennyisége jelentősen csökken, de nyilván számottevő mértékben megmarad.

Fizikai rétegbeli kódolás megsértése

Ez a megoldás a fizikai réteg kódolásainak tulajdonságait használja ki. Történetesen arról van szó, hogy bizonyos kódolások esetén létezhetnek olyan jelsorozatok, jelváltások, amelyek az adatátvitel során objektíve nem fordulhatnak elő. Így ezek felhasználhatóak jelzésekre, mivel az átvitt adattal semmiképpen sem téveszthetőek össze.
Ilyen lehetőséget nyújt például a 4B/5B kódolás. Ez a megoldás 4 bites adatcsoportokat kódol, pontosabban cserél 5 bites adatcsoportokba, NRZI kódolás mellett. A 32 (2^5) lehetséges jelsorozatból viszont csak 16 (2^4) jelsorozat van használatban az adatok kódolására, azaz a maradék 16 jelsorozatból néhány (ez esetben a fele) felhasználható például a keret elejének és végének a jelzésére, illetve vezérlő utasításként. Az eljárás alapja tehát – kódolás és dekódolás esetén is – egy helyettesítő táblázat használata.
Az eljárást azért hívjuk kódsértésnek, mert olyan kód is átvitelre kerül, ami normál körülmények között nem kerülhetne átvitelre, nem használatos kódolásra. Ezek a kódok, azaz jelsorozatok könnyen azonosíthatóak, és így a keretek beazonosításához nincs szükség az átvitt adatok módosítására (majd visszaalakítására).

Adat (4B)	Kódszó (5B) a csatornába		Kódszó (5B) nem használt		Kódszó (5B) jelzésre használt	A jelzés jelentése
0000	11110		00001		00000	Quiet (signal lost)
0001	01001		00010		11111	Idle
0010	10100		00011		11000	Start #1
0011	10101		00101		10001	Start #2
0100	01010		00110		01101	End
0101	01011		01000		00111	Reset
0110	01110		10000		11001	Set
0111	01111		11000		00100	Halt
1000	10010					
1001	10011					
1010	10110					
1011	10111					
1100	11010					
1101	11011					
1110	11100					
1111	11101					

Hasonló elv mentén létezik még például a 8B/10B, és a 64B/66B kódolás is. Összességében ezek az eljárások terhelik meg a legkevesebb „feleslegesen” átvitt adattal az adatátviteli csatornát.
[Érdekes észrevétel, hogy a kódolások elnevezésében a bit van „nagy B-vel” jelölve…]

Az legtöbb adatkapcsolati protokoll a gyakorlatban, a nagyobb biztonság érdekében (még ha ezzel jelentősen meg is növeli az átvitt adatok mennyiségét) a fenti módszerek valamely kombinációját alkalmazza. Például a keret az IEEE802.11 protokollban egy 72 bites, az IEEE802.3 protokollban egy 56 bites előtaggal (Preamble) kezdődik. Ez kellően hosszú ahhoz, hogy a vevő oldal fel tudjon készülni az adatok fogadására. Ezt követi még a fejlécben egy hosszúságot jelző kód (azaz a bájtszám). Így a keret tulajdonságai (kezdete és hossza) is többszörösen biztosítva van az átvitel során.