lomások és a mobil hosztok között található. Az összes közül ez utóbbi a legérdekesebb számunkra, így először ezt fogjuk tanulmányozni.

Az A-interfészen keresztül haladó adatokat tömörítik és titkosítják, és hibajavító kódolással továbbítják. A 274 bites tömörített és titkosított adatcsomagok 378 bit hosszú, hibajavító blokkokba ágyazódnak. Hibajavításra Reed–Solomon-kódolást használnak. Minden egyes RS blokkhoz hét darab 6 bites jelzőszó (flag word) csatlakozik, így végül 420 bit hosszú blokkok jönnek létre. Ezeket a blokkokat hét darab, egyenként 60 bit hosszú mikroblokkra osztják, amelyeket egymás után továbbítanak. Minden 60 bites mikroblokkot ellátnak egy 6 bites jelzőszóval, amely a csatorna állapotát mutatja. Ezek a mikroblokkok ezután vagy egy 19,2 kb/s-os leirányú (downlink) csatornára (a bázisállomás felől), vagy egy másik, szintén 19,2 kb/s-os felirányú (uplink) csatornára (a bázisállomás felé) kerülnek. Két csatorna együtt duplex üzemmódban működik. Lényegében mind a leirányú, mind a felirányú csatorna diszkrét idejű, mivel állandóan a 60 bites mikroblokkok ismétlődnek rajtuk. Mindegyik mikroblokk továbbítása 3,125 ms-ot vesz igénybe.

Minden CDPD cellának egy leirányú/felirányú csatornapárja van a hordozható állomások adatai számára. A leirányú csatorna vezérlése egyszerű, hiszen cellánként egyetlen adó van csak: a bázisállomás. Az itt megjelenő összes keret eljut minden hordozható állomásig, és azok választják ki a nekik, vagy mindegyiküknek szóló kereteket.

A felirányú csatorna az igazán trükkös, mert ezért minden adni kívánó mobil hosztnak versengenie kell. Amikor egy mobilállomás el szeretne küldeni egy keretet, elkezdi figyelni a leirányú csatornán időről időre megjelenő jelzőbitet, amely elárulja, hogy foglalt vagy szabad-e az éppen aktuális feltöltő időrés. Ha foglalt, akkor ahelyett, hogy a következő időrésre várna, inkább kihagy véletlen számú időrést, és csak ezután próbálkozik újra. Ha ekkor is foglalt a felirányú csatorna, akkor egy hosszabb véletlen ideig várakozik, mielőtt újra próbálkozna. A várakozási idő statisztikai várható értéke minden sikertelen próbálkozás után megduplázódik. Amikor végül várhatóan szabadnak találja a csatornát, megkezdi a mikroblokk elküldését.

Az algoritmusnak ezt a részét **DSMA-nak (Digital Sense Multiple Access – többszörös hozzáférés digitális érzékeléssel)** nevezik, és arra szolgál, hogy a hordozható állomások ne akarják azonnal megszerezni maguknak a csatornát, amikor az szabaddá válik. Bizonyos mértékben hasonlít a p-perzisztens réselt CSMA protokollhoz, amelyet már korábban említettünk, mivel mindkét csatornán diszkrét időréseket használ.

A probléma az, hogy a DSMA ellenére is létrejöhetnek ütközések, hiszen kettő vagy akár több állomás is választhatja ugyanazt az időrést az adás megkezdésére. Ahhoz, hogy a hordozható hosztok észrevehessék az esetleges ütközéseket, minden mikroblokkban szerepel egy jelzőbit, amelyik elárulja, hogy a felirányú csatornán előzőleg megjelent keretet sikeresen vették-e. Sajnos a bázisállomás nem tudja a döntést azonnal, egy mikroblokk vételének befejeztével meghozni, ezért az n. mikroblokk utáni hibás/hibátlan visszajelzést el kell halasztani az $n + 2$. blokkig.

Mivel egy állomás sem tudja megállapítani, hogy a legutóbbi adása sikeres volt-e vagy sem, ha több elküldendő mikrokerete is van, egyszerűen folytatja az adást anélkül, hogy újból versenyeznie kellene a csatornáért. Ha a *következő* időrésben látja, hogy az *előző* átvitele nem sikerült, akkor leáll. Egyébként addig folytatja a forgalmazást, amíg el nem ér egy meghatározott maximális Reed–Solomon-blokkszámot, vagy

amíg a bázisállomás be nem billent egy bitet a leirányú csatornán, amely jelzi, hogy pillanatnyilag nem kíván több keretet fogadni ettől az állomástól.

A CDPD-nek egy további jellegzetessége, hogy az adattovábbítást igénylő felhasználókat csak másodosztályú polgárokként kezeli. Amikor egy újabb hangkapcsolatot rendel a bázisállomás ahhoz a csatornához, amelyet éppen CDPD-re használtak, akkor küld egy speciális jelzést a leirányú csatornán, amely lezárja a CDPD kommunikációt. Ha a bázisállomás már tudja, hogy mi lesz az új CDPD csatorna száma, akkor egyben azt is közzéteszi, egyébként pedig a mobilállomásoknak maguknak kell megtalálniuk az előre CDPD használatra kijelölt csatornák közül azt, amelyiken újból beindulhat az adatforgalom. Ilyen módon a CDPD fel tudja használni a cella szabad kapacitását anélkül, hogy a nagy pénzeket jelentő hangkapcsolatokat zavarná.

Ez alapján a leírás alapján tisztán lehet látni, hogy a CDPD szolgáltatást azután adták a hangszolgáltatáshoz, miután azt már üzembe helyezték, valamint látszik, hogy a tervezésnél alapvető feltétel volt az, hogy a már meglevő hangátviteli rendszeren semmilyen módosítást sem eszközölhettek. Emiatt az algoritmus nem törődik a CDPD forgalom létezésével, amikor hangkapcsolatok számára választ csatornát. Ez az oka annak, hogy a CDPD csatornát időnként megszakítják. Mindezek ellenére semmi sem gátolja, hogy a CDPD saját kijelölt csatornán üzemeljen, és ahogyan a népszerűsége nő, a szolgáltatók egyre szívesebben tartanak fenn exkluzív csatornát a CDPD forgalom számára.

CDMA – Code Division Multiple Access

A GSM-et akár úgy is bemutathattuk volna, mint a csatornakiosztás nyers erő (brute force) módszerét, hiszen a maga komplex módján szinte minden ismert módszert (ALOHA, TDM, FDM) magába ötvöz. A CDPD, amellyel különálló keretek továbbíthatók, alapvetően egy nemperzisztens CSMA megoldás. Most pedig vizsgáljunk meg a vezeték nélküli csatornák lefoglalására egy újabb módszert, a **CDMA-t (Code Division Multiple Access – kódosztásos többszörös hozzáférés).**

A CDMA teljes mértékben különbözik az eddig tanult összes módszertől. Voltak olyanok, amelyek a csatorna frekvenciasávokra való felosztásán alapultak, amelyeket aztán statikusan (FDM) vagy igény szerint (WDM) rendeltek az állomásokhoz, azok pedig korlátlanul használhatták a számukra fenntartott sávokat. Más algoritmusok a csatornát löketes átvitelekre használják oly módon, hogy a teljes sávszélességet egy-egy állomás rendelkezésére bocsátják statikusan (TDM fix időrésekkel) vagy dinamikusan (ALOHA). A CDMA lehetővé teszi, hogy az állomások bármikor adhassanak a teljes frekvencia spektrum felett. A többszörös egyidejű átvitelek a kódelmélet felhasználásával kerülnek szétválasztásra. A CDMA felrúgja azt a feltételezést, hogy az ütközést szenvedett keretek tartalma teljesen használhatatlanná válik. Ellenkezőleg, azt feltételezi, hogy az egyidejűleg megjelenő jelek lineárisan összeadódnak!

Mielőtt belevágnánk az algoritmus leírásába, ismerkedjünk meg a csatornaelérés „koktélparti" elméletével. Egy nagy teremben rengeteg pár beszélget. A TDM annak felel meg, mintha az emberek a terem közepén állnának, és fordulókat szervezve, közülük mindig csak egy szólalna meg. Az FDM olyan, mintha az emberek egymástól

A: 0 0 0 1 1 0 1 1	A: (–1 –1 –1 +1 +1 –1 +1 +1)
B: 0 0 1 0 1 1 1 0	B: (–1 –1 +1 –1 +1 +1 +1 –1)
C: 0 1 0 1 1 1 0 0	C: (–1 +1 –1 +1 +1 +1 –1 –1)
D: 0 1 0 0 0 0 1 0	D: (–1 +1 –1 –1 –1 –1 +1 –1)
(a)	(b)

Hat minta:

– – 1 –	C	$S_1 = (-1\ +1\ -1\ +1\ +1\ +1\ -1\ -1)$
– 1 1 –	B + C	$S_2 = (-2\ \ 0\ \ 0\ \ 0\ +2\ +2\ \ 0\ -2)$
1 0 – –	A + B	$S_3 = (\ 0\ \ 0\ -2\ +2\ \ 0\ -2\ \ 0\ +2)$
1 0 1 –	A + B + C	$S_4 = (-1\ +1\ -3\ +3\ -1\ -1\ -1\ +1)$
1 1 1 1	A + B + C + D	$S_5 = (-4\ \ 0\ -2\ \ 0\ +2\ \ 0\ +2\ -2)$
1 1 0 1	A + B + C + D	$S_6 = (-2\ -2\ \ 0\ -2\ \ 0\ -2\ +4\ \ 0)$

(c)

$S_1 \bullet C = (1 +1 +1 +1 +1 +1 +1 +1)/8 = 1$
$S_2 \bullet C = (2 +0 +0 +0 +2 +2 +0 +2)/8 = 1$
$S_3 \bullet C = (0 +0 +2 +2 +0 -2 +0 -2)/8 = 0$
$S_4 \bullet C = (1 +1 +3 +3 +1 -1 +1 -1)/8 = 1$
$S_5 \bullet C = (4 +0 +2 +0 +2 +0 -2 +2)/8 = 1$
$S_6 \bullet C = (2 -2 +0 -2 +0 -2 -4 +0)/8 = 1$

(d)

4.16. ábra. *(a) Négy állomás bináris töredéksorozata. (b) Bipoláris töredéksorozatok. (c) Hat átviteli példa. (d) A C állomás jelének visszaállítása*

távoli csoportokba gyűltek volna, és mindegyik csoport egymáshoz hasonló módon, de egymástól teljesen függetlenül folytatná a párbeszédet. A CDMA ahhoz hasonlít, mintha a vendégek ismét a terem közepén gyűltek volna össze, de minden pár egyszerre beszélne, igaz más-más nyelven. A franciául beszélő páros csakis a francia nyelvre figyel, és minden egyebet zajként kezel. Ennek megfelelően a CDMA kulcsa az, hogy képesek legyünk kiszűrni a hasznos jelet, miközben minden egyebet eldobunk, mintha véletlenszerű zaj lenne. A következőkben a CDMA-ról egy részben leegyszerűsített leírást adunk.

CDMA esetén minden bit-időt m rövid intervallumra, úgynevezett **töredékre (chip)** osztanak. Tipikusan 64 vagy 128 töredék van bitenként, de a példánkban az egyszerűség kedvéért csak 8 töredék/bitet használunk.

Minden állomáshoz egy m bites kód, más néven **töredéksorozat (chip sequence)** tartozik. Egy 1-es bit elküldéséhez az állomás elküldi a saját töredéksorozatát. Ha 0-t akar továbbítani, akkor a töredéksorozatának egyes komplemensét küldi el. Semmilyen egyéb minta használata sincs megengedve. Ilyen módon $m = 8$ esetén, ha az A állomás töredéksorozata 00011011, akkor 1-es bit küldéséhez a 00011011 sorozatot használja, míg 0-s bit továbbításához az 11100100 sorozatot.

Ha a továbbítandó információ mennyiségét b b/s-ról mb töredék/s-ra növeljük, kénytelenek vagyunk a szükséges sávszélességet is m-szeresére növelni (amennyiben nem változtatunk sem a moduláción, sem a kódolási eljáráson), ezért a CDMA a szórt

spektrumú kommunikáció (spread spectrum communication) egy formája. Ha 100 állomás számára egy 1 MHz-es sáv áll rendelkezésre, akkor FDM technikával mindegyiknek 10-10 kHz áll rendelkezésére, így 1 bit per Hz-cel számolva 10 kb/s-mal adhatnak. CDMA használata esetén minden állomás számára rendelkezésre áll az egész 1 MHz-es sáv, így a töredéksebesség 1 millió töredék másodpercenként. Kevesebb, mint 100 töredék per bit esetén, az állomásokra jutó effektív sávszélesség magasabb CDMA eseetén, mint FDM mellett, ráadásul a csatornafoglalás problémája is megoldott, ahogyan azt rövidesen be is látjuk.

Oktatási célokra bipoláris kódolást érdemes használni, amely esetén a bináris 0-nak –1, az 1-nek +1 felel meg. A töredéksorozatokat zárójelek között fogjuk jelölni, így az A állomás által küldött 1-eshez tartozó töredéksorozat (–1 –1 –1 +1 +1 –1 +1 +1) lesz. A 4.16.(a) ábrán négy példaállomás bináris töredéksorozatát láthatjuk. A 4.16.(b) ábrán ugyanezeket láthatjuk a mi bipoláris jelölésünkkel.

Minden állomás saját egyedi töredéksorozattal rendelkezik. Használjuk az $\mathbf{S}$ jelölést az S állomás töredéksorozatára, valamint $\overline{\mathbf{S}}$ jelölést a sorozat negáltjára. A töredéksorozatoknak páronként **ortogonálisaknak** kell lenniük, vagyis minden $\mathbf{S}$ és $\mathbf{T}$ párra a normalizált skaláris szorzatnak (amit az $\mathbf{S} \bullet \mathbf{T}$ jelöl) 0-nak kell lennie. Képlettel kifejezve:

$$\mathbf{S} \bullet \mathbf{T} = \frac{1}{m} \sum_{i=1}^{m} S_i T_i = 0 \tag{4.5}$$

Magyarul, ahány töredékpár egyezik, annyinak kell különböznie is a sorozatokban. A későbbiekben döntő jelentőségű lesz az ortogonális tulajdonság. Vegyük észre, hogy amennyiben $\mathbf{S} \bullet \mathbf{T} = 0$, akkor $\mathbf{S} \bullet \overline{\mathbf{T}}$ szintén 0! Minden töredéksorozatra igaz, hogy az önmagával számított normalizált skaláris szorzata 1:

$$\mathbf{S} \bullet \mathbf{S} = \frac{1}{m} \sum_{i=1}^{m} S_i S_i = \frac{1}{m} \sum_{i=1}^{m} S_i^2 = \frac{1}{m} \sum_{i=1}^{m} (\pm 1)^2 = 1$$

Ez azért van így, mert az m összetevő mindegyike 1, így az összegük m. Vegyük észre azt is, hogy $\mathbf{S} \bullet \overline{\mathbf{S}} = -1$.

Egy bit-idő alatt minden állomásra igaz, hogy vagy 1-es bitet küld töredéksorozatának továbbításával, vagy 0-s bitet küld töredéksorozata negáltjának továbbításával, vagy egyszerűen csendben marad, vagyis semmit sem továbbít. Pillanatnyilag tekintsük úgy, hogy minden állomás szinkronban van, így mindegyik töredéksorozat ugyanabban a pillanatban kezdődik.

Amikor kettő vagy több állomás egyszerre ad, akkor bipoláris jeleik lineárisan összeadódnak. Például, ha egy töredékperiódus alatt három állomás +1-et, egy pedig –1-et ad, akkor az eredmény +2 lesz. Úgy is gondolhatunk erre, mint feszültségek összegére: három állomás +1 voltot, egy pedig –1 voltot ad kimenetén, ami 2 voltot eredményez.

A 4.16.(c) ábrán hat olyan példát láthatunk, amelyek egy vagy több állomás egyidejű forgalmazása mellett jöttek létre. Az első példában a C állomás ad 1-es bitet, így egyszerűen C saját töredéksorozatát kapjuk meg. A második példában B és C

egyaránt 1-es bitet adnak, így a bipoláris töredéksorozatuk összegét kapjuk, részletezve:

$$(-1\ -1\ +1\ -1\ +1\ +1\ +1\ -1) + (-1\ +1\ -1\ +1\ +1\ +1\ -1\ -1) = (-2\ 0\ 0\ 0\ +2\ +2\ 0\ -2)$$

A harmadik példa esetén az *A* állomás 1-et, míg a *B* állomás 0-t küld, míg a többiek csendben maradnak. A negyedik példában *A* és *C* 1-et, míg *B* 0-t ad, az ötödikben mind a négy állomás 1-et sugároz, míg végül az utolsó példa során *A*, *B* és *D* 1-et, *C* pedig 0-t küld. Vegyük észre, hogy a hat sorozat, S_1-től S_6-ig mind egy-egy bitidőt reprezentál.

Ahhoz, hogy egy adott állomás által generált bitsorozatot visszaállíthassunk, ismernünk kell még annak töredéksorozatát. A visszaállítás elvégezhető, ha a vett sorozat (az összes forgalmazó állomás jeleinek lineáris összege) és a figyelni kívánt állomás töredéksorozatának normalizált skaláris szorzatát képezzük. Ha tehát a vett sorozat **S**, a vevő pedig a **C** töredéksorozattal rendelkező állomásra figyel, akkor egyszerűen kiszámítja az $\mathbf{S} \bullet \mathbf{C}$ skaláris szorzatot.

Ahhoz, hogy megértsük, hogyan működik az eljárás, tegyük fel, hogy egyszerre *A* és *C* 1-et, míg *B* 0-t küld! A vevő az $\mathbf{S} = \mathbf{A} + \overline{\mathbf{B}} + \mathbf{C}$ sorozatot látja, és a következő számítást végzi:

$$\mathbf{S} \bullet \mathbf{C} = (\mathbf{A} + \overline{\mathbf{B}} + \mathbf{C}) \bullet \mathbf{C} = \mathbf{A} \bullet \mathbf{C} + \overline{\mathbf{B}} \bullet \mathbf{C} + \mathbf{C} \bullet \mathbf{C} = 0 + 0 + 1 = 1$$

Az első két tag eltűnik, mivel a töredéksorozatokat előrelátóan, a (4.5.) képletnek megfelelően, páronként ortogonálisnak választottuk meg. Most már érthető, miért is volt olyan fontos ez a tulajdonság a töredéksorozatok megválasztásánál.

A helyzetnek egy másik megközelítése lehet, ha úgy képzeljük el, mintha a három töredéksorozat külön-külön érkezne meg ahelyett, hogy összegződtek volna. Ezután a vevő kiszámítaná mindegyikre a skaláris szorzatot külön-külön, majd összeadná az eredményeket. Az ortogonális tulajdonság miatt az összes skaláris szorzat 0-t adna a $\mathbf{C} \bullet \mathbf{C}$ kivételével. Ugyanazt az eredményt kapjuk akkor is, ha először az összeadást, majd a skaláris szorzást végezzük el, mint amikor először szorzunk, és csak aztán összegzünk.

Azért, hogy még egyértelműbb legyen a dekódolás menete, vegyük ismét elő a 4.16.(d) ábra hat példáját! Tegyük fel, hogy a vevő mind a hat összegzett kódból (S_1-től S_6-ig) a *C* állomás által küldött biteket szeretné kinyerni! Kiszámítja a vett **S** sorozatok és a 4.16.(b) ábra **C** töredéksorozatának skaláris szorzatait, majd veszi ezek 1/8-át, mivel jelen esetben $m = 8$. Mint láthatjuk, minden alkalommal a helyes bitet sikerült dekódolni. Tisztára olyan, mintha franciául beszélnénk!

Egy ideális, vagyis zajmentes CDMA rendszer kapacitása (a kiszolgálható állomások száma) korlátlanul nagy lehet éppúgy, ahogy egy zajmentes Nyquist-csatorna kapacitása is korlátlanul nagy lehet, ha mintánként több és több bitet használunk. A gyakorlatban, a fizikai korlátok miatt a kapacitás jóval kisebb. Először is feltételeztük, hogy a töredékek időbeli szinkronban vannak. A valóságban ez megvalósíthatatlan. A legtöbb, amit meg lehet tenni az adó és a vevő szinkronizálása érdekében az, hogy az adó egy elég hosszú, ismert töredéksorozatot sugároz, amire a vevő rá tud állni. Az

összes többi (nem szinkronizált) átvitel ezután már csak véletlenszerű zajnak fog látszani. Ha ezekből nincsen sok, akkor az alapvető algoritmus elfogadhatóan fog működni. Óriási elméleti anyag létezik, amely a töredéksorozatok és a zaj szuperpozíciójával foglalkozik (Pickholtz és mások, 1982). Várhatóan zaj jelenlétében annál nagyobb a helyes dekódolás valószínűsége, minél hosszabbak a töredéksorozatok. A további védelem érdekében a bitsorozatok hibajavító kóddal is elláthatók, de a töredéksorozatok sohasem használnak ilyen kódolást.

A fent tárgyaltak mögött a háttérben megjelent egy feltételezés, amely szerint a vevő az összes forgalmazó állomás jelét egyforma jelszinttel fogja. A CDMA alapvetően olyan vezeték nélküli rendszerekben használatos, amelyekben egy fix bázisállomás és több, a bázisállomástól különböző távolságban levő hordozható állomás van. A bázisállomásnál fogott jelek erőssége attól függ, hogy a különböző adók milyen távolságban vannak. Jó heurisztikus megoldás, ha minden hordozható állomás olyan erősséggel sugároz a bázisállomás felé, mint az attól fogott jelek erősségének inverze, így tehát egy olyan állomás, amelyik gyenge jeleket fog a bázisállomástól, sokkal erősebbeket sugároz, mint egy olyan, amelyik erős jeleket fog. A bázisállomás pedig akár határozott utasításokat is adhat a hordozható állomásoknak, hogy növeljék vagy csökkentsék a sugárzott jelek erősségét.

Azt is feltételeztük, hogy a vevő ismeri az adó kilétét. Törvényszerű, hogy megfelelő számítási kapacitással, a vevő odafigyelhet egyszerre az összes forgalmazóra is, ha párhuzamosan mindegyikre futtatja a dekódoló algoritmust. A valóságban sajnos ezt egyszerűbb mondani, mint végrehajtani. A CDMA-val kapcsolatban sok más komplikáció is felmerül, amelyek fölött el kell siklanunk ebben a rövid ismertetőben. Mindezek ellenére a CDMA egy nagyon okos séma, amelyet egyre elterjedtebben használnak vezeték nélküli kommunikáció során.

Azon olvasók, akik erősebb villamosmérnöki háttérrel rendelkeznek, és szeretnék mélyebben megismerni a CDMA-t, olvassák el Viterbi (1995) munkáit. Crespo és munkatársai írásaiban (1995) egy olyan alternatív szórási sémáról olvashatnak, amelyben a szórást az idő helyett a frekvencia tartományában oldják meg.

4.3. Szabványos LAN-ok és MAN-ok

Most, hogy befejeztük az absztrakt csatornakiosztási protokollok általános tárgyalását, itt az ideje, hogy megnézzük, hogyan alkalmazhatók szabályaink valódi rendszerek, történetesen LAN-ok esetében. Ahogy azt az 1.7.2. szakaszban már említettük, az IEEE több szabványt is előállított helyi hálózatokhoz. Ezeket összefoglaló néven **IEEE 802**-ként ismerjük, amely magában foglalja a CSMA/CD, a vezérjeles sín (token bus) és a vezérjeles gyűrű (token ring) hálózatokat. Ezek a szabványok különböznek a fizikai réteget, valamint a MAC alréteget illetően, de az adatkapcsolati réteg szintjén már kompatibilisek. Az IEEE 802 szabványokat Amerikai Nemzeti Szabványként elfogadta az ANSI, kormányzati szabványként a NIST, valamint nemzetközi szabványként (ISO 8802 néven) az ISO. Meglepően jól olvashatók (egyébként a szabványoknak ilyennek kellene lenniük).

A szabványokat részekre osztották, és ezeket külön könyvekben publikálták. A 802.1 szabvány bevezetést nyújt a szabványhalmazba és meghatározza az interfész-primitíveket. A 802.2 szabvány az adatkapcsolati réteg felső részét definiálja, amely az ún. **LLC (Logical Link Control – logikai kapcsolatvezérlés)** protokollt használja. A 802.3, 802.4 és 802.5 részek a három LAN szabványt, sorra a CSMA/CD-t, a vezérjeles sínt (token bus) és a vezérjeles gyűrűt (token ring) írják le. Mindhárom szabvány a fizikai réteget és a MAC alréteget definiálja. A következő három szakasz e három rendszert mutatja be. További információk: (Stallings, 1993b).

4.3.1. Az IEEE 802.3 szabvány és az Ethernet

Az IEEE 802.3 szabvány egy 1-perzisztens CSMA/CD LAN-t definiál. Az alapgondolat már ismert. Mielőtt egy állomás adni akar, belehallgat a kábelbe. Ha a kábel foglalt, akkor addig vár, amíg az üressé nem válik, különben pedig azonnal adni kezd. Ha egy szabad kábelen egyszerre kettő vagy több állomás kezd el adni, akkor ütközés következik be. Az ütközésben részt vevő összes állomásnak be kell fejeznie adását, véletlenszerű ideig várnia kell, majd az egész eljárást meg kell ismételnie.

A 802.3 szabványnak érdekes története van. Az igazi kezdetet az ALOHA rendszer jelentette, amelyet arra terveztek, hogy rádiós kommunikációt tegyen lehetővé Hawaii szigeteken elszórt gépek között. A módszer később csatornafigyeléssel egészült ki, és a Xerox megépített egy 2,94 Mb/s-os CSMA/CD rendszert, amely 1 km-es kábelen 100 személyi munkaállomást kötött össze (Metcalfe és Boggs, 1976). Ezt a rendszert a *luminiferous éter* után, amelyet valamikor az elektromágneses sugárzás közvetítő közegének hittek, **Ethernet-nek** nevezték el. (Amikor a XIX. században élt brit fizikus, James Clerk Maxwell felfedezte, hogy az elektromágneses sugárzást hullámegyenlet formájában is le lehet írni, akkor a kor tudósai feltételezték, hogy a tér valamilyen éterszerű anyaggal van kitöltve, amelyben a sugárzás terjedni képes. Csak a híres, 1887-es Michelson–Morey-féle kísérlet után fedezték fel a fizikusok, hogy az elektromágneses sugárzás vákuumban is képes terjedni.)

Az Ethernet olyan sikeres volt, hogy a Xerox, a DEC és az Intel összefogva létrehozta a 10 Mb/s-os Ethernet szabványt. Ez a szabvány alkotja a 802.3 szabvány alapját is. A publikált 802.3 szabvány abban különbözik az Ethernet specifikációjától, hogy egy teljes 1-perzisztens CSMA/CD rendszercsaládot ír le, 1-től 10 Mb/s-os sebességig, különböző közegeken működve. Eltérés mutatkozik a két szabvány által definiált fejrészek egyik mezőjének tartalmában is (a 802.3 által hosszmezőnek kezelt mező helyén az Ethernet a csomag típusát jelölte). A kezdeti szabvány egy 10 Mb/s-os, 50 Ω-os koaxiális kábelen futó alapsávú rendszer paramétereire is javaslatot ad. Az egyéb sebességekre, illetve közegekre vonatkozó paramétereket később részletezzük.

Sokan helytelenül „Ethernet" néven hivatkoznak az összes CSMA/CD protokollra, még akkor is, ha valójában egy konkrét 802.3-at megvalósító termékről van szó. A könyvben általában a „802.3" vagy a „CSMA/CD" kifejezéseket használjuk, kivéve, amikor egyértelműen az Ethernet termékre hivatkozunk, mint pl. a következő néhány bekezdésben.

802.3 kábelezés

Mivel az „Ethernet" név a kábelre (az éterre) utal, kezdjük a kábelezéssel az ismerkedést. A 4.17. ábrának megfelelően négyféle kábelezést használnak elterjedten. Sorrendben a **10Base5**, vagy közkedveltebb nevén a **vastag Ethernet (thick Ethernet)** kábelezés volt az első. Megjelenésében egy sárga kerti öntözőtömlőre emlékeztet, amelyen a csapok lehetséges csatlakoztatási pontjait 2,5 méterenként megjelölték. (A 802.3 szabvány nem *írja elő*, hogy a kábelnek sárgának kell lennie, de *javasolja*.) A csatlakozás ún. **vámpír csatlakozókon** keresztül lehetséges, amelyekben egy apró tüskét nyomnak óvatosan a koaxiális kábel középső vezetékébe. A 10Base5 megnevezés azt jelenti, hogy 10 Mb/s sebességgel üzemel alapsávú (baseband) jelekkel, és legfeljebb 500 méter hosszú szegmensek kialakítását teszi lehetővé.

Sorrendben a második kábelezési típus a **10Base2**, más néven a **vékony Ethernet** volt, amely már sokkal könnyebben hajlítható, ellentétben az öntözőcsőhöz hasonlító vastag Ethernettel. Vámpír csatlakozók helyett a már gyári szabványként létező BNC csatlakozókat és T elosztókat használták csatlakozásra. Ezek egyszerűbben használhatóak, és sokkal megbízhatóbbak. A vékony Ethernet sokkal olcsóbb és egyszerűbben is telepíthető, de csak 200 méter a megengedett legnagyobb szegmenshossz, és csupán 30 állomás csatlakozhat rá szegmensenként.

Mindkét közegnél nagy problémát okoz a kábelhiba, a rossz megcsapolás, vagy a laza csatlakozó bemérése, megkeresése. Emiatt különböző módszereket dolgoztak ki ezek felkutatására. Alapvetően egy ismert alakú impulzust adnak a kábel egyik végébe. Ha a jel akadályba vagy a kábel végébe ütközik, akkor visszhang keletkezik, amely visszajut a kábel vizsgált végéhez. Pontos időméréssel, a kiadott impulzus és a visszhang megérkezése közti idő alapján bemérhető a visszhang kiindulási helye. Ezt a technikát **időbeli reflektometriának (time domain reflectometry)** nevezik.

A kábelezés hibáinak bemérésével kapcsolatos problémák egy olyan kábelezési rendszer kialakításához vezettek, amelyben minden állomástól egy kábel megy egy központi **elosztóhoz (hub).** Általában az irodaházakban a már kiépített csavart érpárú telefonkábeleket szokták ilyen hálózatokhoz felhasználni, mivel ezekben a kábelekben normális esetben elegendő számú szabad érpár áll rendelkezésre. Az ilyen sémát **10Base-T**-nek nevezik.

Megnevezés	Kábel	Max. szegmens	Csomópont/ szegmens	Előnyök
10Base5	Vastag koaxiális	500 m	100	Jó gerinchálózatnak
10Base2	Vékony koaxiális	200 m	30	A legolcsóbb rendszer
10Base-T	Csavart érpár	100 m	1024	Egyszerű kiépítés
10Base-F	Optikai	2000 m	1024	A legjobb épületek között

4.17. ábra. *Az alapsávú 802.3 LAN-ok legelterjedtebb típusai*

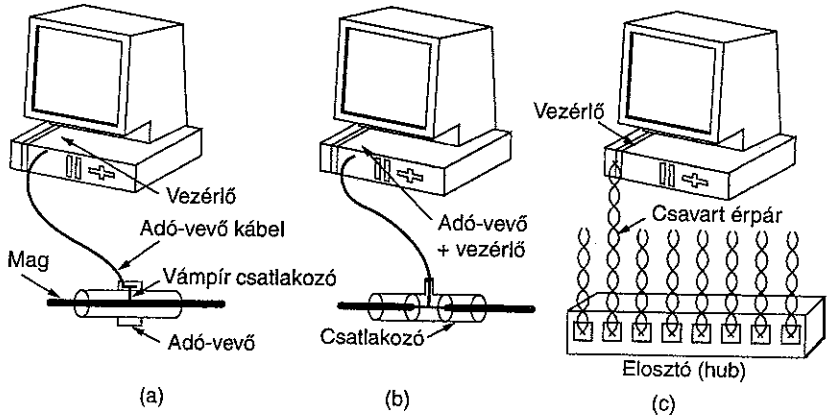
Ezt a három kábelezési sémát mutatja be a 4.18. ábra. 10Base5 esetén a kábelre egy **adó-vevő (transceiver)** csatlakozik, amelynek tüskéje valósítja meg az összeköttetést a kábel belső magjával. Az adó-vevőbe szerelik azt az elektronikát, amely a vivőjel érzékelést, illetve az ütközések érzékelését hivatott elvégezni. Ütközés érzékelése esetén az adó-vevők egy speciális jelet kezdenek el adni a kábelen, hogy az összes többi adó-vevő biztosan érzékelhesse az ütközés bekövetkeztét.

10Base5 esetén az adó-vevőt a számítógép csatolókártyájával egy **adó-vevő kábel** köti össze. Ez a kábel öt, egymástól függetlenül árnyékolt csavart érpárt tartalmaz. Hossza legfeljebb 50 méter lehet. Két érpár adatbemenetként, illetve adatkimenetként szolgál, két további érpárt vezérlőjelek ki- és bevitelére használnak, az ötödik érpár segítségével pedig (ezt nem mindig használják ki) az adó-vevő elektronikáját láthatja el a számítógép tápfeszültséggel. Vannak olyan adó-vevők is, amelyek akár nyolc közeli számítógépet is képesek kiszolgálni, így csökkentik a hálózat kialakításához szükséges adó-vevők számát.

Az adó-vevő kábel egy számítógépben elhelyezett illesztőkártyához csatlakozik. Ezen a kártyán egy olyan vezérlőáramkör található, amely kereteket küld az adóvevőnek, illetve kereteket fogad attól. A vezérlő feladata, hogy az adatokat megfelelő keretekbe szervezze, kiszámítsa hozzájuk az ellenőrző összegeket, illetve ellenőrizze a beérkezett keretek integritását. Néhány vezérlő egyéb funkciókat is képes ellátni. Például a beérkező és a továbbítandó keretek számára puffereket, illetve sorokat kezelhet, DMA átvitelt bonyolíthat le a gazda számítógéppel, vagy egyéb, a hálózat felügyeletével kapcsolatos funkciókat is támogathat.

A 10Base2 esetében a kábelhez való csatlakozást egy egyszerű, passzív BNC T-csatlakozó oldja meg. Az adó-vevő elektronika az illesztőkártyán található, és minden állomásnak mindig saját adó-vevője van.

10Base-T esetén egyáltalán nincs kábel, csak elosztó (hub), ami nem más, mint egy elektronikával teleépített doboz. Ilyen felépítésű rendszerekben az állomások beiktatása és kivétele sokkal egyszerűbb, szintúgy a hibás kábelszakaszok felderítése is. A csavart



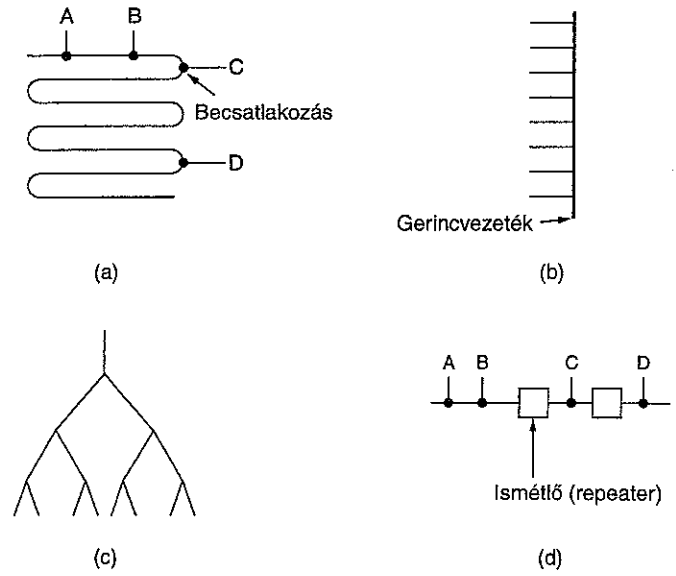
4.18. ábra. *Három különböző 802.3-as kábelezés. (a) 10Base5. (b) 10Base2. (c) 10Base-T*

érpáras megoldás hátránya mindössze az, hogy a vezeték, ami a gépek és az elosztó között fut legfeljebb csak 100, de jó minőségű (5-ös kategóriájú) kábelek esetén is csupán 150 méter hosszú lehet. Ráadásul a nagyméretű elosztók akár több százezer forintba is kerülhetnek. Ezek ellenére a 10Base-T architektúra rendületlenül egyre népszerűbbé válik egyszerű kiépíthetőségének köszönhetően. A 10Base-T egy gyorsabb változatával (100Base-T) is megismerkedünk nemsokára, még e fejezet folyamán.

A 802.3 által támogatott negyedik kábelezési lehetőség az üvegszál használata, a **10Base-F**. Ez a megoldás tagadhatatlanul nagyon drága a csatlakozók és végberendezések magas árai miatt, de kiváló a zajtűrése, és általában ezt a módszert használják épületek, vagy egymástól nagy távolságban levő elosztók összekötésére.

A 4.19. ábra egy épület különböző bekábelezési lehetőségeit szemlélteti. A 4.19.(a) ábra egy olyan megoldást jelképez, amelyben egyetlen kábel kígyózik végig az épületen szobáról szobára, miközben minden állomás a hozzá legközelebbi ponton csatlakozik rá. A 4.19.(b) ábrán egy függőleges gerincvezeték fut az alagsortól a tetőig, amelyhez speciális erősítőkön (repeater – ismétlő) keresztül vízszintes kábelek csatlakoznak minden emeleten. Néhány épületben a függőleges gerinckábel vastag, míg a vízszintes kábelek vékonyak. A legkedveltebb topológia a fa elrendezés, amelyet a 4.19.(c) ábra szemléltet. Ez a topológia azért előnyös, mert olyan hálózatokban, ahol bizonyos állomáspárok között két útvonal is létezik, a különböző útvonalakon haladó jelek között interferencia léphet fel.

Mindegyik 802.3 hálózat esetén a kábel szegmensenkénti maximális hossza előre meghatározott. Nagyobb hálózatok kialakítása érdekében a 4.19.(d) ábrán látható mó-



4.19. ábra. *Kábelezési topológiák. (a) Lineáris. (b) Gerincvezetékes. (c) Fa. (d) Szegmentált*

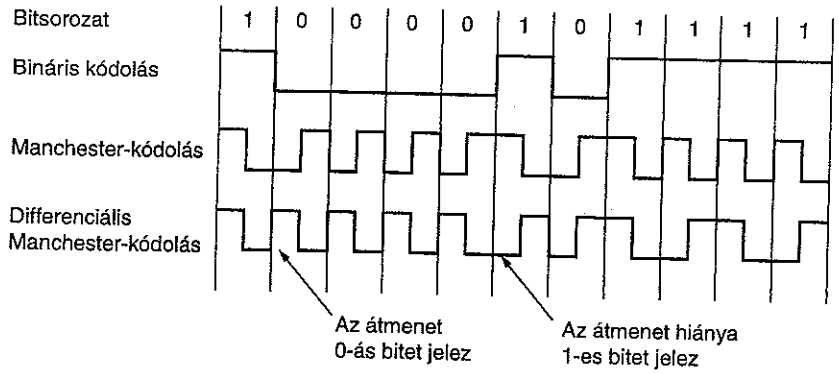
don több kábelszegmenst **ismétlőkkel (repeaters)** lehet összekötni. Az ismétlő egy olyan eszköz, amely a protokoll architektúrának kizárólag a fizikai rétegében működik. Veszi a jeleket, regenerálja, majd mindkét irányban továbbküldi azokat. Szoftveres szemszögből nézve, az ismétlőkkel összekötött kábelszegmensek ugyanúgy viselkednek, mintha egyetlen kábel lenne csak a rendszerben (leszámítva az ismétlők működése során keletkezett kisebb késleltetéseket). Egy rendszer tartalmazhat több kábelszegmenst és több ismétlőt is, de két adó-vevő pár között nem lehet 2,5 km-nél nagyobb távolság, és közöttük négy ismétlőnél több nem lehet.

Manchester-kódolás

A 802.3 szabványok egyik változata sem használ bináris kódolást, ahol 0 volt a 0 logikai szintnek, az 5 volt a logikai 1-es szintnek felel meg, mivel ez nem egyértelmű. Ha egy állomás például a 0001000 bitsorozatot küldi el, akkor a többiek helytelenül azt hihetik, hogy ez eredetileg 1000000 vagy 0100000 volt, mivel nem tudják megkülönböztetni az üres csatornán mérhető 0 voltot a 0 logikai szintnek megfelelő 0 volttól.

Amire szükség van az, hogy külső óra segítsége nélkül, minden állomás félreérthetetlenül el tudja dönteni, mikor kezdődik, mikor ér véget, és mikor tart éppen a felénél egy bit a csatornán. Két ilyen megoldás is létezik: a **Manchester-kódolás,** és a **differenciális Manchester-kódolás.** A Manchester-kódolás esetén minden bit-időt két egyenlő méretű részre (intervallumra) vágnak. Bináris 1 küldése esetén az első intervallum alatt magas feszültségszint, míg a második alatt alacsony jelenik meg a csatornán. Bináris 0 elküldése esetén mindez megfordul: az első intervallum alatt alacsony, a második alatt pedig magas feszültségszintet kényszerítenek a csatornára. Ez a módszer biztosítja, hogy minden bit-idő közepén legyen egy átmenet, így a vevő könnyedén összeszinkronizálódhat az adóval. A Manchester-kódolás hátránya azonban, hogy a bináris kódoláshoz szükséges sávszélesség kétszeresét igényli, mivel az impulzusok hossza a felére csökkent. A Manchester-kódolást a 4.20.(b) ábra mutatja be.

A 4.20.(c) ábrán bemutatott differenciális Manchester-kódolás az alap eljárásnak



4.20. ábra. *(a) Bináris kódolás. (b) Manchester-kódolás. (c) Differenciális Manchester-kódolás*

egy változata. Ennek során az 1-es bitet a bit-idő elején hiányzó átmenet jelzi, míg ennek az átmenetnek a megléte 0-s bitre utal. Mindkét esetben megtörténik a szokványos átmenet a bit-idő felénél. A differenciális megoldás bonyolultabb berendezést igényel, ám cserébe jobb zajvédettséget biztosít. Az összes alapsávú 802.3 rendszer, egyszerűsége miatt, Manchester-kódolást használ. A magas szintnek +0,85 volt, míg az alacsonynak –0,85 volt felel meg, amelynek köszönhetően az egyenáramú (DC) komponens 0 volt lesz.

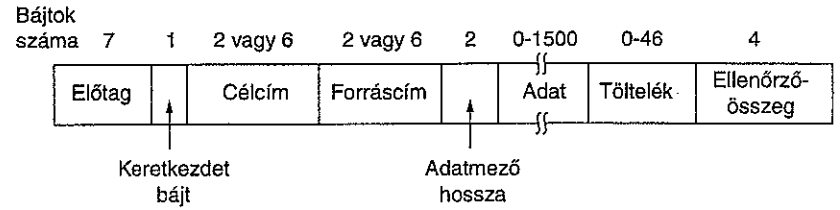
A 802.3 MAC-protokollja

A 802.3 (IEEE, 1985a) keretstruktúráját a 4.21. ábrán láthatjuk. Minden keret egy 7 bájtos *előtaggal (preamble)* kezdődik, amely 10101010 mintájú. E minta Manchester-kódolása, amely egy 10 MHz-es, 5,6 μs időtartamú négyszögjel, lehetőséget biztosít a vevő órájának, hogy az adó órájához szinkronizálódjon. Ezután következik a *keretkezdet (start of frame)* bájt, amely a keret kezdetét jelöli ki az 10101011 mintával.

A keret két címet tartalmaz: egy célcímet és egy forráscímet. A szabvány 2 és 6 bájtos címeket is megenged, de a 10 Mb/s-os alapsávú szabvány számára kijelölt paraméterek csak 6 bájtos címek használatát engedélyezik. A célcím legfelső helyi értékű bitje közönséges címek esetén 0, csoportcímek esetén viszont 1 értékű. A csoportcímek több állomás egyetlen címmel való megcímzését teszik lehetővé. Amikor egy keretet csoportcímmel küldünk el, akkor azt a csoport minden tagja veszi. Az állomások egy meghatározott csoportjának való keretküldést **többes küldésnek (multicast)** nevezik. A csupa 1-esekből álló cím az **adatszórás (broadcast)** esetén használatos. A célcímben csupa 1-est tartalmazó kereteket az összes állomás veszi.

A címzés további érdekessége a 46. bit (a legmagasabb helyi értékű bit szomszédja) használata, amely megkülönbözteti a helyi és a globális címeket. A helyi címeket a hálózat adminisztrátorai jelölik ki, és nincs jelentőségük a helyi hálózaton kívül. Ezzel szemben, a globális címeket az IEEE jelöli ki azért, hogy a világon ne fordulhasson elő két azonos globális cím. Mivel $48 - 2 = 46$ bit áll rendelkezésre, ezért megközelítőleg 7×10^{13} globális cím létezik. Az alapgondolat az, hogy 48 bitet használva már a világ bármely két állomása megcímezheti egymást. A célállomások megtalálásának módja már a hálózati rétegre tartozik.

A *hosszmező (length field)* az adatmezőben található adatbájtok számát adja meg. A minimum 0, a maximum 1500 bájt. Bár a 0 hosszúságú adatmező érvényes, mégis



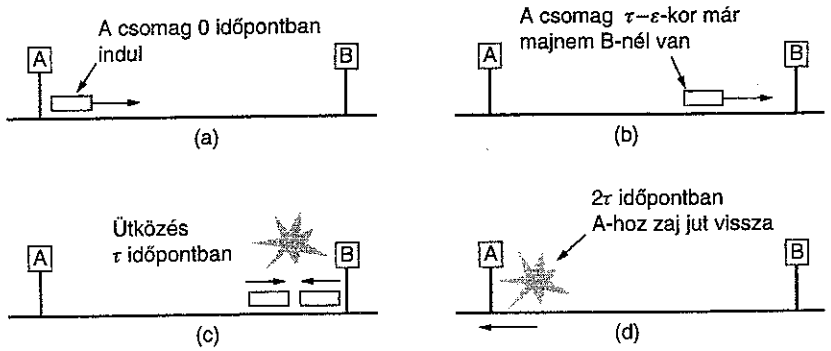
4.21. ábra. *A 802.3 keretformátuma*

problémákat okozhat. Amikor egy adó-vevő ütközést érzékel, csonkolja az aktuális keretet, ami azt jelenti, hogy kóbor bitek, keretdarabkák mindig jelen lehetnek a kábelen. Az érvényes keretek és a szemét megkülönböztetése érdekében a 802.3 szabvány szerint egy érvényes keretnek a célcímtől az ellenőrző összegig bezárólag, legalább 64 bájt hosszúnak kell lennie. Ha tehát egy keret adatrésze 46 bájtnál rövidebb, akkor a *töltelékmezőt (pad)* kell használni a keret minimális méretének eléréséhez.

A minimális kerethosszúság előírását más (sokkal fontosabb) indok is szükségessé teszi. Rövid keretek engedélyezése esetén előfordulhatna, hogy egy állomás még azelőtt befejezné a keretének küldését, mielőtt annak első bitje elérné a kábel legtávolabbi végét, ahol az még egy másik kerettel ütközhet. A problémát a 4.22. ábra illusztrálja. A 0 időpillanatban a hálózat egyik végén elhelyezkedő *A* állomás elküld egy keretet. Jelöljük azt az időtartamot, amíg ez a csomag elér a hálózat másik végéig, τ-val! Éppen azelőtt, hogy a keret elérte volna a vezeték másik végét (azaz $\tau - \varepsilon$ pillanatban), a legtávolabbi állomás, *B* szintén adni kezd. Amikor *B* észleli, hogy az általa vett jel erőssége nagyobb, mint amit maga sugárzott, rájön, hogy ütközés történt. Abbahagyja az adását és egy 48 bit hosszú zajlöketet (noise burst) állít elő, hogy a többi állomást is figyelmeztesse az ütközésre. Az adás megkezdése után csak körülbelül 2τ idő elteltével érzékeli a zajlöketet az eredeti küldő fél, amely hatására szintén leáll a forgalmazással. Ezután véletlenszerű ideig vár, majd újból próbálkozik.

Egyértelmű, hogy ha egy állomás egy nagyon rövid keretet küld el, akkor is bekövetkezik az ütközés, ám az átvitel be is fejeződik, mielőtt a zajlöket a 2τ késleltetés után megérkezik. Ezek alapján az adó fél tévesen azt a következtetést vonja le, hogy a keretet sikeresen küldte el. Az ilyen helyzetek elkerülése érdekében minden keretnek olyan hosszúnak kell lennie, hogy elküldése legalább 2τ időt igényeljen. Egy (a 802.3 specifikációja alapján) maximális, azaz 2500 méter hosszú, és négy ismétlőt tartalmazó LAN esetén az engedélyezett legrövidebb keret továbbítása sem fejeződhet be 51,2 μs-on belül. Ennyi időhöz 64 bájt tartozik. Az ennél rövidebb kereteket pedig a kitöltő mező segítségével 64 bájtra kell növelni.

A hálózatok sebességének növekedésével arányosan kell a minimális keretméretnek is növekednie, vagy a megengedett maximális kábelhossznak csökkennie. Egy



4.22. ábra. *Az ütközésérzékelés 2τ időt is igénybe vehet*

2500 méteres, 1 Gb/s-os sebességen működő LAN mellett a minimális keretméretnek 6400 bájtnak kell lennie. Alternatív módon az is lehetséges, hogy a minimális keretméret csak 640 bájt, de ekkor a maximális távolság egyetlen állomáspár között sem lehet több 250 méternél. Ezek a megkötések egyre inkább kényelmetlenek lesznek, amint a gigabites hálózatok irányába haladunk.

Az utolsó 802.3-as keretmező az *ellenőrző összeg (checksum)*. Ez tulajdonképpen az adatok egy 32 bites hash-kódja. Ha néhány adatbit (a vezetéken megjelenő zaj miatt) sérülten érkezik meg, akkor az ellenőrző összeg majdnem biztosan rossz lesz, így a hiba felfedezhető. Az ellenőrző összeg algoritmusa a harmadik fejezetben ismertetett ciklikus redundanciaellenőrzésen (CRC) alapul.

A kettes exponenciális visszalépéses algoritmus

Most nézzük meg, hogyan jelenik meg a véletlenszerűség, amikor ütközés történik. A modell a 4.5. ábrán látható. Egy ütközés után az időt diszkrét résekre osztva képzelhetjük el, ahol az időrések hossza akkora, mint amennyi idő legrosszabb esetben ahhoz kell, hogy egy jel visszaérhessen az éteren keresztül (vagyis 2τ). A 802.3 szabvány által megengedett maximális méretekhez (2500 méter és 4 ismétlő) igazodva ezeknek az időréseknek a hosszát 512 bit-időre, azaz 51,2 μs-ra választották meg.

Az első ütközés után minden állomás véletlenszerűen vagy 0, vagy 1 időrésnyit várakozik, mielőtt újra próbálkozna. Ha két állomás ugyanúgy sorsolt volna, akkor ismét ütköznek. A második ütközés után véletlenszerűen 0-t, 1-et, 2-t vagy 3-at sorsolnak, és ennyi időrésnyit várakoznak. Ha harmadszor is ütköznek (ennek valószínűsége 0,25), akkor a 0 és a $2^3 - 1$ közé eső intervallumból választják ki, hogy mennyi időrésnyit várakoznak.

Általánosan igaz, hogy az i-edik ütközés után a véletlen szám a 0 és $2^i - 1$ közötti intervallumból kerül kiválasztásra, és az állomások ennek megfelelő számú időrést hagynak ki. Mindazonáltal a tizedik ütközés után már nem nő tovább a tartomány, hanem az 1023 marad a felső korlátja. 16. ütközés után a vezérlő bedobja a törülközőt, és hibajelzést küld a számítógépnek. A további hibajavítás már a felsőbb rétegek feladata.

Ezt az algoritmust **kettes exponenciális visszalépésnek (binary exponential backoff)** nevezik. Azért erre az algoritmusra esett a választás, mert dinamikusan képes az adni kívánó állomások számához igazodni. Ha a véletlenszám-generálás felső határa minden ütközés esetén 1023 lenne, akkor két állomás újbóli ütközésének valószínűsége valóban elhanyagolható volna, de a várakozási idő várható értéke több száz rés körül alakulna, amely megengedhetetlenül nagy késleltetéseket okozna. Másfelől viszont, ha az állomások örökösen csak a 0 és az 1 közül választanának csak, akkor 100 egyszerre adni kívánó állomás keretei addig ütköznének, amíg végre 99 állomás 0-t, míg a maradék egy az 1-est (vagy fordítva) választaná. Ez akár évekig is eltarthatna. Azáltal, hogy a véletlenszám-generálás intervalluma az egymást követő ütközések hatására exponenciálisan nő, az algoritmus biztosítja azt, hogy kevés ütköző állomás esetén viszonylag kis késleltetés következzen be, ugyanakkor nagyszámú állomás esetén az ütközés még belátható időn belül feloldódjon.

Ahogy az eddigiekből kiderült, a CSMA/CD nem biztosít nyugtázást. Mivel az ütközés puszta hiánya nem garantálja azt, hogy a bitek a kábelen levő zajtüskék miatt nem sérülnek meg, ezért a megbízható átvitel érdekében a célállomásnak ellenőriznie kell az ellenőrző összeget, és ha az hibátlan, akkor erről a tényről egy nyugtakeret küldésével értesítenie kell a forrást. Rendes körülmények között egy protokollban ez a nyugtázás egy másik keretet igényelne, amelynek elküldése érdekében, akárcsak egy adatkeret esetén, meg kellene szereznie a csatorna hozzáférési jogát. A versenyalgoritmus egy egyszerű módosításával azonban ez elkerülhető, és a keret sikeres vételéről a küldőnek gyorsan nyugta küldhető (Tokoro és Tamaru, 1977). Ehhez mindössze az kell, hogy a sikeres adásokat követő versengési rések közül az elsőt a célállomás számára kell fenntartani.

A 802.3 teljesítménye

Vizsgáljuk most meg a 802.3 teljesítményét nagy és állandó terhelés mellett! Tegyük fel, hogy k állomás folyamatosan adásra kész állapotban van! A bináris exponenciális visszalépés algoritmusának teljes vizsgálata nagyon bonyolult, ezért Metcalfe és Boggs (1976) példáját követve, minden résben állandó újraküldési valószínűséget feltételezünk. Ha minden egyes állomás p valószínűséggel ad egy versengés során, akkor annak valószínűsége (A), hogy valamelyik állomás meg is tudja szerezni a csatornát:

$$A = kp(1-p)^{k-1} \tag{4.6}$$

A akkor maximális, ha $p = 1/k$ és ha $k \to \infty$, akkor $A \to 1/e$-hez. Annak valószínűsége, hogy a versengési intervallum pontosan j időrést tartalmaz $A(1-A)^{j-1}$, így tehát a versengésenkénti rések számának középértéke:

$$\sum_{j=0}^{\infty} jA(1-A)^{j-1} = \frac{1}{A}$$

Mivel minden rés időtartama 2τ, ezért a versengési intervallum átlagos hossza $w = 2\tau/A$. Optimális p-t feltételezve a versengési rések számának középértéke soha nem nagyobb mint e, így w legfeljebb $2\tau e \approx 5{,}4\tau$ lehet.

Ha egy átlagos keret elküldéséhez P másodpercre van szükség, akkor sok küldeni kívánó állomás esetén:

$$\text{Csatornahatékonyság} = \frac{P}{p + 2\tau/A} \tag{4.7}$$

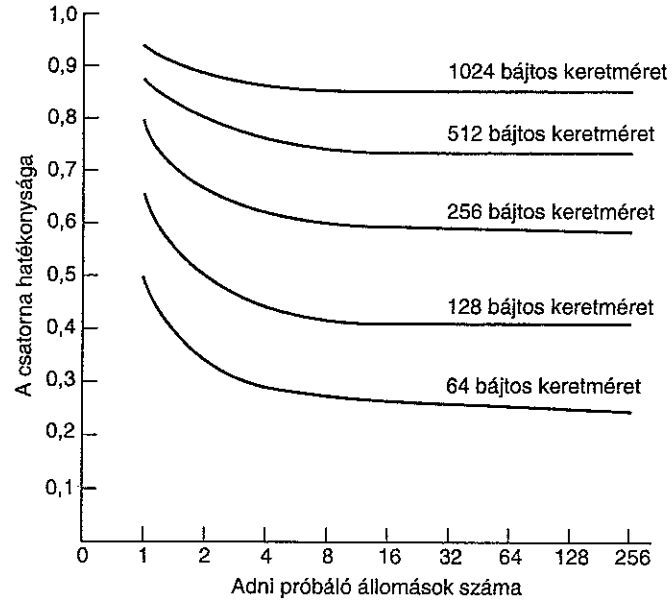
Itt látható, hogy a két állomás közti maximális kábelhosszúság hol játszik szerepet a teljesítmény alakulásában, így adva esélyt a 4.19.(a) ábrán látható topológiáktól eltérő elrendezések számára. Minél hosszabb a kábel, annál hosszabb a versengési intervallum hosszúsága is. 2,5 km-nél nem hosszabb kábelt és két adó-vevő között nem több mint 4 ismétlőt engedélyezve a körbejárási idő 51,2 μs-ra korlátozható, amely 10 MHz-nél 512 bitnek, vagyis 64 bájtnak felel meg, amely éppen a minimális keretméret.

Tanulságos a (4.7) egyenlőséget az F kerethossz, a B hálózati sávszélesség, az L kábelhosszúság és a c jelterjedési sebesség segítségével az optimális e keretenkénti versengési rés esetére átalakítani. $P = F/B$ teljesülése esetén a (4.7) egyenlet így alakul:

$$\text{Csatornahatékonyság} = \frac{1}{1 + 2BLe/cF} \tag{4.8}$$

Amikor a nevező második tényezője nagy, a hálózat hatékonysága kicsi. Konkrétan, ha a hálózati sávszélesség és a távolság nő (BL szorzat), akkor ez csökkenti az egy adott keretméretre számított hatékonyságot. Sajnos azonban a legtöbb hálózati hardver kutatás éppen ennek a szorzatnak a növelésére irányul. Nagy távolságokon nagy sávszélességet akarnak elérni (pl. üvegszálas MAN-ok), ami azt sugallja, hogy a 802.3 nem a legalkalmasabb az ilyen alkalmazások számára.

A 4.23. ábrán a (4.8) egyenlőség alapján $2\tau = 51{,}2$ μs és 10 Mb/s-os adatátviteli sebesség mellett az adni kész állomások függvényében rajzoltuk fel a csatornahatékonyság görbéjét. 64 bájtos résidő mellett nem meglepő, hogy a 64 bájtos keretek nem hatékonyak. Másfelől, 1024 bájtos kereteket és versengési intervallumonként e darab



4.23. ábra. *A 802.3 hatékonysága 10 Mb/s-os sebesség, és 512 bites résidő esetén*

(amely csak aszimptotikusan közelíthető) 64 bájtos rést feltételezve, a hatékonyság 0,85, míg a versengési periódus 174 bájt hosszú lesz.

Az adásra kész állomások számának középértékét nagy terhelés esetén a következő (durva) megfontolások alapján határozhatjuk meg. Minden keret a csatornát egy versengési periódus és egy keretátviteli idő erejéig tartja fel, azaz összesen $P + w$ másodpercig. A keretek száma ezért másodpercenként $1/(P + w)$. Ha minden állomás λ keret/s átlagsebességgel állítja elő a kereteket, és a rendszer k állapotban van (ennyi adásra kész állomás van éppen), akkor a nem blokkolt állomások egyesített bemeneti sebessége $k - \lambda$ keret/s. Mivel állandósult állapotban a bemeneti és kimeneti sebességnek meg kell egyezniük, ezért ezeket a kifejezéseket egyenlővé tehetjük, és az egyenletet megoldhatjuk k-ra. (Vegyük észre, hogy w a k függvényében változik!) Ennél sokkal részletesebb vizsgálatok is léteznek (Bertsekas és Gallager, 1992).

Valószínűleg megéri megemlíteni, hogy a 802.3-mal (és más hálózatokkal) kapcsolatban rengeteg elméleti teljesítményelemzés létezik. Mindegyik munka azt feltételezi, hogy a forgalom Poisson-folyamat szerint változik. Ahogy a kutatók azonban elkezdték a valódi forgalmi adatokat vizsgálni, kiderült, hogy a hálózatok forgalma aligha Poisson, hanem inkább egészen sajátságos (Paxson és Floyd, 1994; Willinger és mások, 1995). Ez azt jelenti, hogy a hosszú időintervallumokra számított átlagértékek nem egyenlítik ki a forgalmat. Egy óra minden percére számított csomagszám átlagának ugyanolyan szórása van, mint egy perc másodpercei alatt számított átlagértékeknek. Ennek a felfedezésnek az a következménye, hogy a hálózati forgalomról alkotott legtöbb modell nem alkalmazható a valóságban, és (jókora) fenntartással kezelendő.

Kapcsolt 802.3 LAN-ok

Minél több állomást csatlakoztatunk egy 802.3-as LAN-hoz, a forgalom annál nagyobb lesz. Idővel a hálózat telítődni fog. Az egyik megoldás az, ha magasabb sebességre állunk át, mondjuk 10 Mb/s-ról 100 Mb/s-ra. Ez a megoldás azonban maga után vonja az összes 10 Mb/s-os csatolókártya leselejtezését, és az újabb, drágább berendezések beszerzését. Ha a 802.3 áramkörök a számítógép alaplapján helyezkednek el, akkor lehetséges, hogy a csere nem is valósítható meg.

Szerencsére van másik, kevésbé drasztikus megoldás is: a 4.24. ábrán látható kapcsolt 802.3-as LAN. Egy ilyen rendszer lelke a kapcsoló (switch), amely egy nagy sebességű hátlapot (backplene) és az ebbe dugható 4–32 darab vonali illesztőkártyát tartalmaz. A vonali kártyák 1–8 csatlakozóval rendelkeznek. Minden csatlakozótól leggyakrabban egy 10Base-T típusú csavart érpár vezet egy hoszt számítógéphez.

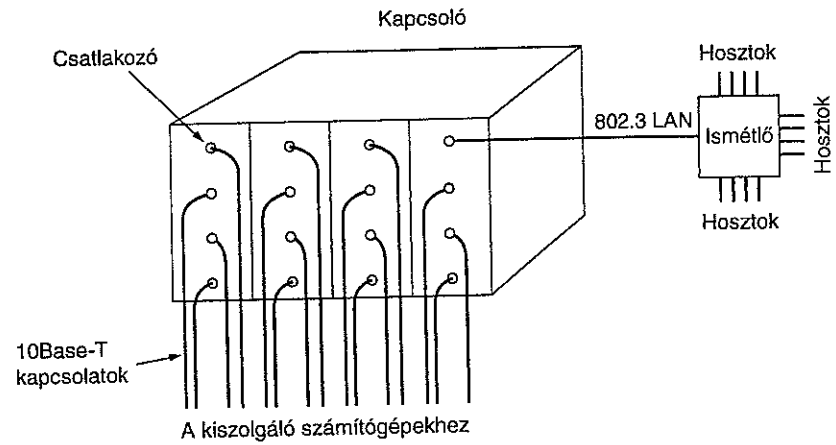
Amikor egy állomás 802.3-as keretet szeretne továbbítani, akkor egy teljesen szokványos keretet küld a kapcsoló felé. A kapcsoló vonali kártyája, miközben veszi a keretet, először megvizsgálja, hogy a célállomás ugyanahhoz a kártyához csatlakozik-e. Ha igen, akkor a keretet odamásolja. Ha nem, akkor a keretet a nagy sebességű hátlapon keresztül elküldi annak a kártyának, amelyikhez a címzett csatlakozik. Az átviteli panel általában 1 Gb/s-os átviteli sebességgel dolgozik, és saját belső protokollt használ.

Mi történik akkor, ha két olyan állomás forgalmaz egyszerre, amelyek ugyanahhoz

a vonalhoz csatlakoznak? Ez attól függ, hogy a kártyát hogyan alakították ki. Az egyik lehetőség az, hogy a kártya portjai össze vannak kötve, így egy kis kártyára integrált LAN-t alkotnak. Ilyen esetben a kártyán bekövetkezett ütközések pontosan úgy érzékelhetők és kezelhetők le, mint a normál CSMA/CD hálózatok esetén – a bináris visszalépéses algoritmus használatával újra kell küldeni a keretet. Az ilyen kártyáknál minden pillanatban kártyánként csak egyetlen átvitel valósulhat meg, viszont a kártyák párhuzamosan, egymás zavarása nélkül működhetnek. Ezzel a tervezéssel minden kártya saját **ütköztetési tartományt (collision domain)** képez, amely független az összes többitől.

A másik fajta vonali kártya használata esetén mindegyik bemeneti port pufferelt, így a beérkező keretek a kártya RAM-jában tárolódhatnak el megérkezésük pillanatában. Az ilyen tervezés lehetővé teszi, hogy a portok egyidejűleg fogadjanak (és továbbítsanak) kereteket a párhuzamos, duplex működés érdekében. Miután egy keret teljes egészében megérkezett, a kártya eldöntheti, hogy a címzett ugyanahhoz a kártyához csatlakoztatott gép, vagy egy távolabbi port felé irányítsa a keretet. Az első esetben a keret egyenesen a célállomás felé továbbítható, míg a második esetben a hátlapon keresztül kell a megfelelő kártyának elküldeni. Ennél a tervezésnél minden port külön képez egy-egy ütközési tartományt, így a kapcsolóban egyáltalán nem fordulnak elő ütközések. A teljes rendszer átbocsátóképességét akár egy nagyságrenddel is növelhetjük a 10Base-5 hálózathoz képest, amely az egész hálózatot egyetlen ütköztetési tartományként kezeli.

Mivel a kapcsoló csak annyit vár el, hogy szabványos 802.3 keretek érkezzenek a bemeneti portjaira, néhány portját koncentrátorként lehet használni. A 4.24. ábrán, a jobb felső portra nem egy különálló számítógép, hanem egy 12 portos elosztó (hub) csatlakozik. Az elosztóra érkező keretek a szokványos módon versenyeznek a 802.3-as LAN használatáért, beleértve az ütközéseket és a bináris visszalépéses algoritmus használatát. A sikeresen továbbított keretek folytatják útjukat a kapcsoló felé, ahol



4.24. ábra. *Kapcsolt 802.3 LAN*

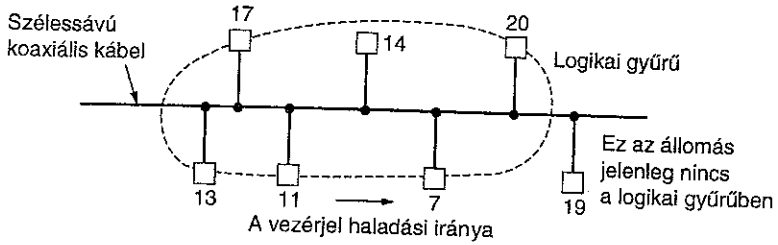
ugyanolyan elbánásban részesülnek, mint bármilyen egyéb beérkező keret: továbbításra kerülnek a nagy sebességű hátlapon keresztül a megfelelő kimeneti vonal felé. Ha a kapcsoló összes portjára elosztókat csatlakoztatnak, akkor a kapcsoló átalakul egy 802.3– 802.3 típusú híddá. A hidakról a fejezet során, később még lesz szó.

4.3.2. Az IEEE 802.4 szabvány: vezérjeles sín

Bár a 802.3-at elterjedten használják hivatalokban, a 802. szabvány kidolgozásakor a General Motors más, gyártásautomatizálásban érdekelt vállalatok szakemberei komoly fenntartással kezelték e szabványt. Ennek egyik oka az volt, hogy valószínűségi alapon működő MAC protokolljából következően, ha egy állomásnak nincs szerencséje, akkor esetleg nagyon sokáig nem képes keretet küldeni (vagyis a legrosszabb esetnek nincs korlátja). Egy másik ok az volt, hogy a 802.3 keretek nem rendelkeznek prioritással, ami alkalmatlanná teszi azokat valós idejű rendszerekben való használatra, mivel ott egy fontos keretet nem tarthatnak fel kevésbé fontosak.

Egy egyszerű, kiszámítható legrosszabb esettel rendelkező rendszer a gyűrű, amelyben az állomások egymás után következve küldik el kereteiket. Ha n állomás van, és T másodpercig tart egy keret elküldése, akkor egyetlen állomásnak sem kell nT másodpercnél többet várnia egy keret elküldésére. A 802-es bizottság gyártásautomatizálással foglalkozó szakemberei a gyűrű koncepcióját kedvelték, nem kedvelték viszont annak fizikai megvalósítását, mivel gyűrűkábelben előforduló hiba az egész hálózatot megbéníthatja. Továbbá nyilvánvaló, hogy a gyűrű-topológia kevéssé illeszkedik a futószalagok egyenes vonalú topológiájához. Végeredményben egy új szabványt fejlesztettek ki, amely a 802.3 adatszórásos kábelének megbízhatóságával, ugyanakkor a gyűrű kiszámítható legrosszabb esetbeli viselkedésével rendelkezik.

Ez a 802.4 szabvány (Dirvin és Miller, 1986; IEEE, 1985b), amelyet **vezérjeles sínnek (token bus)** is neveznek. Fizikailag a vezérjeles sín egy lineáris vagy fa elrendezésű kábel, amelyre állomásokat csatlakoztatnak. Az állomások, a 4.25. ábrának megfelelően, logikailag gyűrűbe szervezettek, amelyben mindegyik állomás ismeri bal, illetve jobb oldali szomszédjának címét. Amikor a gyűrűt üzembe helyezik, elsőként a legmagasabb sorszámú állomás küldhet. Miután megtette, a küldés jogát továbbadja a közvetlen szomszédjának. Ezt egy speciális vezérlőkeret, az ún. **vezérjel (token)** elküldésével végzi el. A vezérjel a logikai gyűrű mentén körbejár. Küldési jo-



4.25. ábra. *Vezérjeles sín*

ga csak a vezérjelet aktuálisan birtokló állomásnak van. Mivel egyszerre csak egyetlen állomás birtokolhatja a vezérjelet, ezért ütközés sohasem fordulhat elő.

Fontos megértenünk, hogy az állomások fizikai sorrendje lényegtelen. Mivel a kábel (tulajdonságából fakadóan) egy adatszórásos közeg, ezért minden állomás minden keretet vesz, de nem veszi figyelembe azokat, amelyek nem neki szólnak. Amikor egy állomás továbbadja a vezérjelet, akkor úgy küldi el a közvetlen logikai gyűrűszomszédjának címzett vezérjelkeretet, hogy valójában nem ismeri annak fizikai helyét. Azt is érdemes megjegyeznünk, hogy amikor az állomásokat bekapcsolják, nem kerülnek azonnal a gyűrűbe (pl. a 14-es és 19-es állomások a 4.25. ábrán), az állomások gyűrűbe juttatása, illetve az onnan való törlése a MAC protokoll hatáskörébe tartozik.

A 802.4 MAC protokollja nagyon összetett, amelyben minden állomásnak 10 különböző időzítést és több mint két tucat belső állapotváltozót kell nyilvántartania. A 802.4 szabvány sokkal hosszabb, mint a 802.3, több mint 200 oldalas. A két szabvány stílusában is jelentősen eltér. A 802.3 a protokollokat Pascal nyelvű eljárások formájában adja meg, míg a 802.4 véges állapotú automatákkal írja le úgy, hogy a tevékenységet Ada®-ban fogalmazza meg.

A fizikai réteghez a vezérjeles sín a kábeltelevíziózásban alkalmazott 75 Ω-os szélessávú koaxiális kábelt használja. Mind az egykábeles, mind a kétkábeles rendszer engedélyezett, főállomással vagy anélkül. Három különböző analóg modulációs módszer engedélyezett: fázisfolytonos frekvenciabillentyűzés, fáziskoherens frekvenciabillentyűzés és a duobináris amplitúdóbillentyűzés. A lehetséges sebességek: 1, 5 és 10 Mb/s. A modulációs módszerek a logikai 0 és 1 ábrázolásának, illetve a kábel tétlenségének nem kizárólagos kifejezői, létezik ugyanis további három szimbólum, amelyek hálózatvezérlési célokat szolgálnak. Mindent összevetve a fizikai réteg teljesen inkompatíbilis a 802.3-mal, és sokkal bonyolultabb.

A vezérjeles sín MAC protokollja

A logikai gyűrű üzembe helyezésekor az állomások cím szerint csökkenő sorrendben kerülnek be sorban egymás után a gyűrűbe. A vezérjelet szintén a magasabb című állomásoktól az alacsonyabb című állomások felé küldik. Amikor egy állomás megszerzi a vezérjelet, egy meghatározott ideig adhat, majd a vezérjelet tovább kell küldenie. Ha a keretek elegendően rövidek, akkor több, egymást követő keret elküldésére is lehetségessé válik. Ha egy állomás nem rendelkezik elküldendő adatokkal, akkor a vezérjelet azonnal továbbítja.

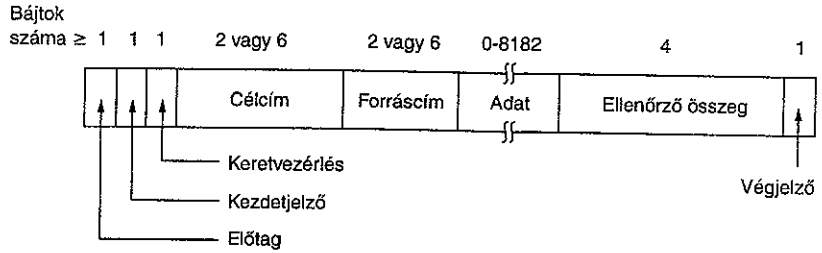
A vezérjeles sín négy prioritási osztályt definiál a forgalom számára: 0-st, 2-est, 4-est és 6-ost, ahol a 0-s a legalacsonyabb, a 6-os a legmagasabb prioritású. A legkönnyebben ez úgy képzelhető el, hogy minden állomás belül négy alállomást tartalmaz, amelyek az egyes prioritási osztályoknak felelnek meg. A MAC alrétegbe érkező bemeneti adatok, prioritásuk szerint szétválogatva a négy alállomás közül a megfelelőhöz kerülnek. Így minden egyes alállomásnak saját sora van az elküldendő keretek számára.

Amikor a kábelen keresztül a vezérjel megérkezik egy állomáshoz, annak azonnal

a 6-os alállomása aktivizálódik, így megkezdheti kereteinek elküldését, ha van. Amikor végzett (vagy amikor az időzítés lejárt), a vezérjelet belül átadja a 4-es alállomásnak, amely időzítésének lejártáig küldhet. Ezután az is továbbadja a vezérjelet a 2-es prioritású alállomásnak. Ez a folyamat addig ismétlődik, amíg a 0-s alállomás is el nem küldi összes keretét, vagy amíg időzítése le nem jár. Ekkor a vezérjelet a következő állomásnak kell elküldeni.

Anélkül, hogy a különböző időzítések felügyeletének részleteibe belemennénk, világosnak kellene lennie, hogy az időzítések megfelelő beállításával elérhető, hogy a teljes vezérjel-birtoklási idő egy jól meghatározott része a 6-os prioritású forgalomé legyen. Az alsóbb prioritásoknak azzal az idővel kell élniük, ami marad. Ha a magasabb prioritású alállomásnak nincs szüksége a rendelkezésre álló időre, akkor az alacsonyabb prioritású alállomások felhasználhatják az így felszabadult részt, tehát az nem vész kárba.

Ez a prioritási séma, amely tehát a hálózati sávszélesség egy ismert részét a 6-os prioritású forgalom számára tartja fenn, hang vagy más valós idejű forgalom lebonyolítására használható. Tegyük fel, hogy egy 50 állomásos, 10 Mb/s sebességű hálózat paramétereit úgy állítjuk be, hogy a 6-os prioritású forgalom a teljes sávszélesség 1/3-át foglalja le. Ekkor az egyes állomások 67 kb/s garantált sebességgel rendelkeznek a 6-os prioritású adatok átviteléhez. Ez a sávszélesség állomásonként egy ISDN hangcsatorna megvalósításához elegendő, egy kis, a vezérlőinformációk átvitelére alkalmas maradék sávszélességgel együtt.



4.26. ábra. *A 802.4 keretformátuma*

A vezérjeles sín keretformátumát a 4.26. ábrán láthatjuk. Sajnos eléggé különbözik a 802.3 keretformátumától. Az *előtag (preamble)*, akárcsak a 802.3-ban, a vevő órajelének szinkronizálását segíti elő, a különbség csupán annyi, hogy itt megengedett az 1 bájt előtaghosszúság is. A *kezdetjelző (starting delimiter)* és a *végjelző (ending delimiter)* mező a keret határait jelzik. Mindkét mező analóg kódolású szimbólumokat tartalmaz, amelyek a digitális 0 és 1 kódolástól jelentősen különböznek. Így kerülik el, hogy ezek a jelek véletlenül a felhasználói adatok között is előforduljanak. A határoló jeleknek köszönhetően nincs szükség adathossz mezőre.

A *keretvezérlő* mező az adat- és vezérlőkereteket különbözteti meg egymástól. Adatkeretek esetén a keretek prioritását hordozza. Tartalmazhat olyan jelzést is, amely a célállomást a keret hibátlan vagy hibás vételének nyugtázására kötelezi. Enélkül a

jelzés nélkül a célállomás nem küldhetne semmit, hiszen nincs nála a vezérjel. Ez a jelzés a vezérjeles sínt, a Tokoro és Tamaru által javasolt nyugtázási sémára emlékeztető rendszerré alakítja.

Vezérlőkeretek esetén a *keretvezérlő* mező a keret típusát jelöli ki. A megengedett típusok halmazát a vezérjel-átadási és a különböző gyűrű-karbantartási keretek alkotják. Ez utóbbiak között találjuk az állomások gyűrűbeléptetését vagy azok gyűrűből való kilépését jelző kerettípusokat stb. Vegyük észre, hogy ezzel ellentétben a 802.3 protokoll nem rendelkezett vezérlőkeretekkel. Ott a MAC alréteg csak a keretek kábelre juttatását biztosította; azok tartalmával nem foglalkozott.

A *célcímet* és *forráscímet* hordozó mezők ugyanolyanok, mint a 802.3-ban. Akárcsak ott, egy adott hálózatban vagy csak 2 bájtos, vagy csak 6 bájtos címeket használhatnak az állomások, a kevert használat nem engedett. A kezdeti 802.4 szabványban ugyanakkor még mindkét címméret használható volt egyidejűleg. Az egyedi és csoportcímek, valamint a lokális és globális címek kijelölésére ugyanazok vonatkoznak, mint a 802.3-nál.

Amikor a címek 2 bájtosak, az *adatmező* legfeljebb 8182 bájt hosszú, amikor a címek 6 bájtosak, akkor legfeljebb 8174 bájt hosszú lehet. Ez több mint ötször olyan hosszú, mint a legnagyobb 802.3-beli keret, amelyet azért választottak annyira rövidnek, hogy egy állomás ne tarthassa fel túl hosszú ideig a többi állomást. A vezérjeles sín esetén az időzítéseket fel lehet használni ilyen „önző" állomásokkal szemben, egyébként viszont nagyon kényelmes hosszú kereteket küldeni akkor, ha nem valós idejű követelményekkel állunk szemben. Az átviteli hibák kiszűrésére szolgál az *ellenőrző összeg*. Ugyanazt a polinom alapú algoritmust használja, mint a 802.3.

A vezérjeles sín vezérlőkeretei a 4.27. ábrán láthatók. Ezeket a következőkben fogjuk tárgyalni. Eddig csak a vezérjelállomások közötti átadására szolgáló *vezérlőkerettel* ismerkedtünk meg. A többi vezérlőkeret elsősorban az állomásoknak a logikai gyűrűbe való beléptetésével és kiléptetésével kapcsolatos.

Keretvezérlő mező	Név	Jelentése
00000000	Claim_token	Vezérjel igénylése a gyűrű beindítása során
00000001	Solicit_successor_1	Állomások belépésének engedélyezése
00000010	Solicit_successor_2	Állomások belépésének engedélyezése
00000011	Who_follows	Helyreállítás elveszett vezérjel esetén
00000100	Resolve_contention	Versenyhelyzet feloldása, amikor egyszerre több állomás is be akar lépni
00001000	Token	Vezérjel-átadás
00001100	Set_successor	Állomás kilépésének engedélyezése

4.27. ábra. *A vezérjeles sín vezérlőkeretei*

A logikai gyűrű karbantartása

Az állomások be- és kikapcsolása folytonosan előforduló esemény, így a gyűrűbe való be- és kiléptetést meg kell oldani. Ennek módját a MAC protokoll részletekig menően definiálja úgy, hogy közben fenntartja az előírt legrosszabb esetbeli vezérjel-körbefordulási időt is. Itt csak a használt mechanizmus vázát ismertetjük.

Miután a gyűrű létrejött, minden állomás nyilvántartja a két logikailag szomszédos állomás címét. A vezérjel birtokosa az egyik SOLICIT_SUCCESSOR keret elküldésével rendszeresen ajánlatot kér a gyűrűbe nem tartozó állomásoktól. A keret a küldő, és az azt követő állomás címét tartalmazza. Azok az állomások, amelyek címe ebbe az intervallumba esik (hogy megmaradjon a gyűrűben az állomások címük szerinti csökkenő sorrendje), kérhetik beléptetésüket a gyűrűbe.

Ha az adott résidő alatt (2τ, mint a 802.3-nál) egyetlen állomás sem jelentkezik, akkor a **válaszablak (response window)** bezárul, és a vezérjel birtokosa folytatja tovább a forgalmazást. Ha pontosan egy állomás kér beléptetést, akkor az végrehajtódik, és ez az állomás lesz a vezérjel birtokosának az új, a gyűrűben következő szomszédja.

Ha egyszerre kettő vagy több állomás is jelzi belépési szándékát, akkor kereteik, a 802.3 hálózatokhoz hasonlóan ütközni fognak és összekeveredenek. A vezérjel birtokosa ekkor egy RESOLVE_CONTENTION keret elküldésével kezdeményezi a versenyfeloldási algoritmus végrehajtását. Ez az algoritmus a bináris visszaszámlálás egy olyan változata, amely egyszerre mindig csak két bitet használ.

Minden állomás csatolója további két véletlenszerű bitet kezel még. Ezek értéke határozza meg azt a 0, 1, 2 vagy 3 résnyi késleltetést, amely a versengés további csökkentésére hivatott. Összefoglalva: két bejelentkezni kívánó állomás csak akkor ütközik, ha ugyanazt a címet használják, és véletlenbitjeik is megegyeznek. A 3 résidőnyit várakozni kénytelen állomások hátrányos helyzetben maradásának elkerülése érdekében a véletlenbiteket minden használat után, vagy legalább 50 ms-onként periodikusan újragenerálják.

Az új állomások beléptetési kérelmei nem befolyásolhatják a vezérjel körbefutási idejének legrosszabb esetre számított értékét. Minden állomásban van egy időzítő óra, amely nullázódik mindig, amikor az állomás megszerzi a vezérjelet. Amikor a vezérjel beérkezik, az óra újbóli nullázása előtt az állomás megvizsgálja az óra értékét (vagyis az előző vezérjel-körbejárási időt). Ha ez meghalad egy bizonyos értéket, akkor arra következtet, hogy a forgalom túl nagy, ezért ebben a körben az állomás nem fog belépési ajánlatot küldeni. Akárhány ajánlat is érkezik, egyszerre mindig csak egy állomás beléptetése valósulhat meg. Ennek az a célja, hogy korlátozni lehessen a gyűrűkarbantartásra felhasználható időt. Nagy forgalom esetén nem garantálható egy állomás gyűrűbe kerülésének maximális ideje, de ez a gyakorlatban nem lehet több néhány másodpercnél.

A gyűrű elhagyása könnyű. Egy X állomás, amelyet a P állomás előz meg, és az S követ, úgy lép ki a gyűrűből, hogy SET_SUCCESSOR üzenetet küld P-nek, amelyben tudatja, hogy ezentúl P-nek nem X, hanem S a követője. Ezután X egyszerűen abbahagyja a forgalmazást.

A gyűrű üzembe helyezése az új állomás beléptetésének egy speciális esete. Tekintsünk most egy tétlen rendszert, amelyben az összes gép ki van kapcsolva! Amikor

bekapcsolják az első állomást, az észreveszi, hogy egy bizonyos ideje már nincs forgalom. Ezután elküld egy CLAIM_TOKEN üzenetet. Mivel nem érzékel más, a vezérjelért versengő társat, létrehoz egy vezérjelet, és felépít egy olyan gyűrűt, amelynek egyetlen tagja lesz, éspedig saját maga. Rendszeres időközönként lekérdezi, hogy van-e újabb állomás, amelyik be akar lépni a gyűrűbe. Ahogy egy új állomást feszültség alá helyeznek, válaszolni fog ezekre a kérésekre, és az előzőekben leírt módon beléphet a gyűrűbe. Végül tehát minden állomás be tud kerülni a gyűrűbe. Ha két állomást egyszerre helyeznek feszültség alá, akkor a vezérjel létrehozásáért folytatott küzdelemben a protokoll a módosított kettes visszaszámlálási algoritmust és a két véletlenbitet használja.

Az átviteli és hardverhibák következtében probléma lehet a gyűrűvel és a vezérjellel is. Például, mi történik akkor, ha egy állomás a vezérjelet egy, már működésképtelenné vált állomásnak adja tovább? A megoldás magától értetődő. Miután az állomás elküldte a vezérjelet, elkezdi figyelni szomszédját, hogy ad-e ki vezérjelet vagy keretet. Ha nem küld semmit, akkor az állomás újabb vezérjelet küld.

Ha ez sem jár sikerrel, akkor az egy WHO_FOLLOWS keretet küld el, amely a szerinte soron következő állomás címét tartalmazza. Amikor a gyűrűben a meghibásodott állomás mögött álló észrevesz egy WHO_FOLLOWS keretet, amelyben a saját megelőző szomszédjának címét találja, akkor a keret küldőjének egy SET_SUCCESSOR keretet küld, amelyben magát nevezi meg új szomszédként. A meghibásodott állomás így kikerül a gyűrűből.

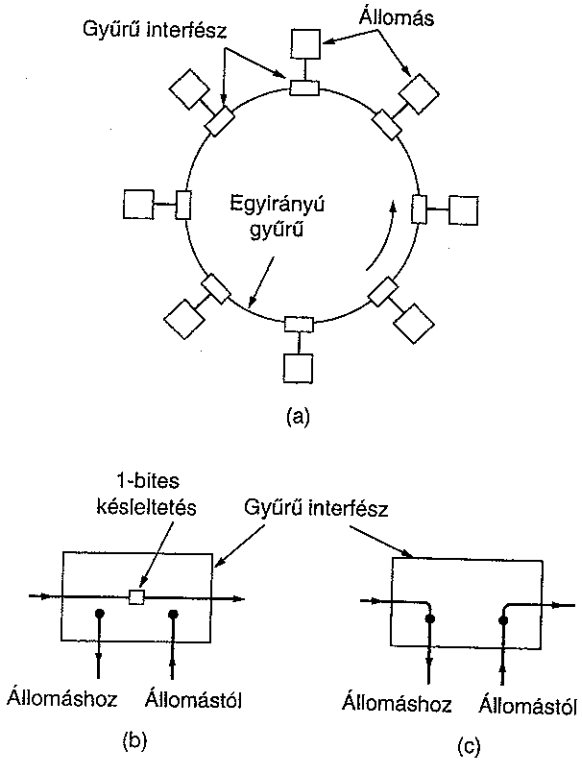
Most tegyük fel, hogy az állomásunknak nem csak a vezérjelet nem sikerült továbbadnia, de még a hibás állomás után következő állomást sem sikerült felkutatnia, mert az szintén működésképtelenné vált. Ilyen esetben új stratégiát kell alkalmazni, ezért elküld egy SOLICIT_SUCCESSOR_2 keretet, hogy kiderítse, van-e még egyáltalán működő állomás rajta kívül a rendszerben. Ezt követően ismét a szabályos versenyprotokoll kerül végrehajtásra, amelyben minden olyan állomás részt vehet, amely be akar kerülni az újonnan kialakuló gyűrűbe. Végül felépül újra a gyűrű.

Megint egy más típusú probléma, amikor a vezérjel birtokosa megy tönkre, és így nem képes eleresztení a vezérjelet. Ezt a problémát a gyűrű beindítási algoritmusa oldja meg. Minden állomás rendelkezik egy aktivitásidőzítővel, amely egy keretnek a hálózaton történő megjelenésével nullázódik. Amikor ez az időzítő lejárt, az állomás egy CLAIM_TOKEN üzenetet bocsát ki, és egy új vezérjel létrehozásának jogáért verseny indul, amelynek végeredménye a módosított bináris visszaszámlálási algoritmusnak és a két véletlen bitnek megfelelően alakul. További probléma az, ha egyszerre több vezérjel jelenik meg. Ha a vezérjelet birtokló állomás észrevesz egy másik állomástól származó vezérjelet, akkor a sajátját azonnal eldobja. Ha két vezérjel volt, akkor most már csak egy van. Ha több mint két vezérjel lenne, akkor ez a folyamat addig folytatódik, amíg végül csak egy vezérjel marad. Ha az állomások véletlenül az összes vezérjelet eldobnák, akkor az aktivitás hiánya egy vagy több állomást arra késztetne, hogy vezérjel-generálási folyamatot indítson el, amelynek lefolyását már korábban láthattuk.

4.3.3. Az IEEE 802.5 szabvány: vezérjeles gyűrű

Gyűrűhálózatok már jó ideje működnek (Pierce, 1972), régóta használják ezeket mind helyi, mind nagy kiterjedésű hálózatok kialakítási formájaként. Számos vonzó tulajdonságuk ellenére az igazság az, hogy a gyűrű nem túl jól használható adatszórásos átvitelre, mivel tulajdonképpen kör alakba rendezett, pontpárok közötti kapcsolatok halmazából áll. A kétpontos kapcsolatok jól ismert és kipróbált technikán alapulnak, és sodrott érpárt, koaxiális kábelt vagy optikai szálat használnak. A gyűrűtechnológia majdnem teljesen digitális, szemben pl. a 802.3-mal, amely jelentős mennyiségű analóg komponenst tartalmaz az ütközések érzékeléséhez. A gyűrű egyenlő esélyű és kiszámítható felső határú csatorna-hozzáférést biztosít. Ezen jó tulajdonságai miatt az IBM saját LAN-ját a gyűrű topológiára alapozta, és az IEEE is felvette 802-es szabványai közé a **vezérjeles gyűrű** szabványt, ez lett a 802.5-ös szabvány (IEEE, 1985c: Latif és mások, 1992).

Gyűrűhálózatok tervezésénél és elemzésénél alapvető kérdés az, hogy mekkora egy bit „fizikai hossza". Ha egy gyűrű R Mb/s-os adatátviteli sebességgel rendelkezik, akkor $1/R$ μs-onként kerül ki egy-egy bit az átviteli közegre. Tipikus, 200 m/ms-os jelterjedési sebességgel számolva egy bit megközelítőleg $200/R$ métert foglal el a gyűrűn. Ez azt jelenti, hogy egy 1 Mb/s-os gyűrű, amelynek kerülete 1000 m, egyszerre csak 5 bitet tartalmazhat. A gyűrűn egyidejűleg megjelenő bitek számának jelentősége és hatása a későbbiekben válik majd világossá.



4.28. ábra. *(a) Gyűrűhálózat. (b) Vételi üzemmód. (c) Adási üzemmód*

Ahogy fent már említettük, egy gyűrű valójában pontpárok közötti kapcsolatokkal összekötött gyűrű-interfészek gyűjteménye. Minden, illesztőhöz érkező bit egy ideiglenes 1 bites pufferbe kerül, ahonnan ismét kimásolódik a gyűrűre. A pufferben levő bitet a gyűrűre való visszaírás előtt az állomás megvizsgálja, és szükség szerint módosíthatja. A bitek interfészenkénti tárolása és másolása 1 bites késleltetést eredményez minden egyes állomásnál. A 4.28. ábra egy gyűrűt és annak illesztőit mutatja be.

Az állomások tétlensége esetén, a vezérjeles gyűrűn egy speciális bitminta, a **vezérjel (token)** jár körbe. Amikor egy állomás keretet akar küldeni, akkor még az adás megkezdése előtt meg kell szereznie a vezérjelet, és el is kell távolítania azt a gyűrűből. Mindezt úgy teheti meg, hogy a 3 bájtos vezérjel egyik bitjét invertálja, és e három bájtot az elküldendő normál adatkeret első 3 bájtjává változtatja. Mivel csak egyetlen vezérjel van, ezért egyszerre csak egyetlen állomás adhat, így tehát a csatorna-hozzáférés ugyanúgy oldódik meg, mint a vezérjeles sín esetén.

A vezérjeles gyűrű tervezésének további gondja az, hogy magának a gyűrűnek is elegendő késleltetéssel kell rendelkeznie ahhoz, hogy tétlen állomások esetén is képes legyen a teljes vezérjel befogadására és keringetésére. A késleltetés két komponensből áll: az egyes állomások által okozott 1 bites tárolási késleltetésből, valamint a jelterjedési késleltetésből. A tervezőknek majdnem minden gyűrűben számolniuk kell az állomások különböző időkben történő, különösen éjszakai kikapcsolásával. Ha az illesztők a gyűrűtől kapják áramellátásukat, akkor az állomások leállításának nincs hatása rájuk. Ha azonban az interfészek kívülről kapják az áramot, akkor úgy kell azokat megtervezni, hogy kikapcsolt állapot esetén a bemenetet és a kimenetet fixen össze kell kapcsolni. Ez nyilvánvalóan megszünteti az 1 bites késleltetést. Emiatt rövid gyűrűk esetén éjszakára mesterséges késleltetéseket ültetnek be, így teszik képessé a gyűrűt a vezérjel további fenntartására és keringetésére.

A gyűrűillesztőknek két üzemmódja van: vételi és adási. Vételi üzemmódban, ahogyan azt a 4.28.(b) ábra is szemlélteti, a bemeneti bitek 1 bites késleltetéssel a kimenetre másolódnak. Adási üzemmódban, amely csak a vezérjel megszerzése után következhet be, az interfész megszakítja a kapcsolatot a bemenet és a kimenet között, és saját adatait írja ki a gyűrűre. Azért, hogy a vételi módból az adási üzemmódba való átkapcsolás egyetlen bit-idő alatt megvalósítható legyen, az interfészeknek rendszerint pufferelniük kell az állomástól jövő egy vagy több elküldendő keretet ahelyett, hogy ilyen rövid idő alatt kellene lekezelniük azokat.

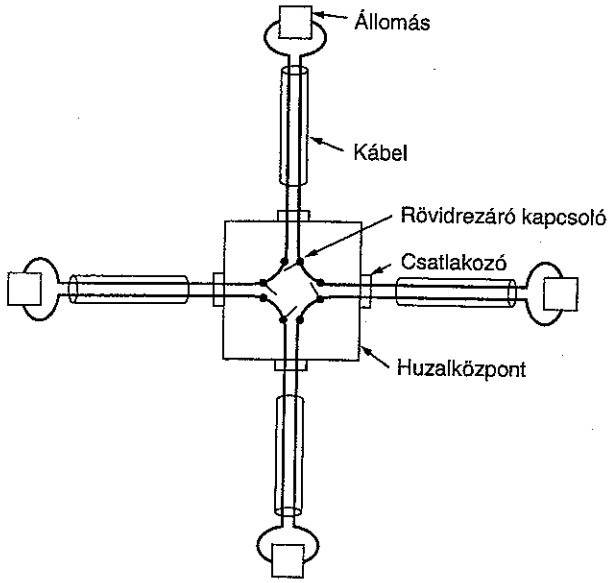
A gyűrűben körbeterjedő biteket a küldő állomások távolítják el a gyűrűről, mikor azok visszaérkeznek. A küldő állomás vagy tárolja azokat és összehasonlítja az eredetileg küldött adatokkal, hogy a gyűrű megbízhatóságára következtethessen, vagy eldobja azokat. Ez a gyűrűszerkezet nem korlátozza a keretek maximális méretét, hiszen az egész keret egyszerre úgysem jelenik meg a gyűrűben. Miután egy állomás elküldte utolsó keretének utolsó bitjét is, vissza kell helyeznie a vezérjelet a gyűrűbe. Az utolsó bit visszaérkezése és a gyűrűből való kivétele után az interfésznek azonnal vételi

állapotba kell visszaállnia, nehogy ismét leszedje a vezérjelet, amely az utolsó bitet követően érkezik.

A nyugtázás magától értetődően megoldható a gyűrűn. A keretformátumnak csak egy 1 bites mezőt kell tartalmaznia, amely kezdetben nulla. Amikor a célállomás megkapja a keretet, ezt a bitet 1-be állítja. Természetesen, ha a nyugta jelentése az, hogy az ellenőrző összeg helyes, akkor ennek a mezőnek az ellenőrző összeg után kell következnie, és a gyűrűinterfésznek képesnek kell lennie arra, hogy az utolsó adatbit beérkezése után az ellenőrző összeget azonnal leellenőrizze. Ha a keret adatszórásos, azaz több állomásnak szól, akkor ennél sokkal bonyolultabb nyugtázási mechanizmust használnak (ha használnak egyáltalán).

Amikor a forgalom kicsi, akkor a működési idő legnagyobb részében a vezérjel fut körbe-körbe a gyűrűben. Alkalomszerűen egy-egy állomás kivonja a gyűrűből, elküldi kereteit, majd ismét visszahelyezi. Ha azonban a forgalom olyan nagy, hogy az egyes állomásoknál sorok keletkeznek, akkor ahogy egy állomás befejezi adását és a vezérjelet visszahelyezi, a következő állomás, figyelve azt, azonnal lecsap rá, és újból kivonja a gyűrűből. Ily módon az adási engedély egyenletesen, minden állomást érintve körbejár a gyűrűben. Nagy terhelés esetén a hálózat hatékonysága akár a 100%-ot is elérheti.

A vezérjeles gyűrűk általános ismertetése után, most fordítsuk figyelmünket a 802.5 szabványra! A fizikai rétegben a 802.5 1 vagy 4 Mb/s-os sebességre alkalmas árnyékolt sodrott érpárt (STP – shielded twisted pair) használ, igaz az IBM később bejelentett egy 16 Mb/s-os változatot is. A jeleket a 4.20.(c) ábrán bemutatott differenciális Manchester-kódolással ábrázolják. A magas és alacsony logikai értékeket 3,0–4,5 V közötti pozitív, illetve ugyanilyen szintű negatív jelek képviselik. Amíg a differenciális Manchester-kódolás alacsony-magas és magas-alacsony átmenetekkel jelzi a 0-s és 1-es biteket, addig a 802.5 szabvány a speciális vezérlő bájtokban (pl. keret kezdet, keret vég jelzésére) fenntart alacsony-alacsony, illetve magas-magas átmeneteket is. Ezek a nem adat jellegű bitek csak egymást követő ellentétes párokban fordulnak elő, hogy ne idézzenek elő egyenfeszültségű komponenst a gyűrűn.



4.29. ábra. *Huzalközponttal összekötött négy állomás*

A gyűrűhálózatokkal szemben felmerülő egyik kritika az, hogy a kábelen bárhol előforduló megszakadás esetén az egész gyűrű leáll. Ez a probléma nagyon elegánsan oldható meg egy ún. **huzalközpont (wire center)** felhasználásával, amelyet a 4.29. ábra illusztrál. Míg logikailag az állomások gyűrűt alkotnak, addig fizikailag állomásonként (legalább) két sodrott érpárral a huzalközpontba csatlakoznak. Az egyik érpár az állomások felé, a másik pedig az állomásoktól a központ felé irányuló adatok továbbítását végzi.

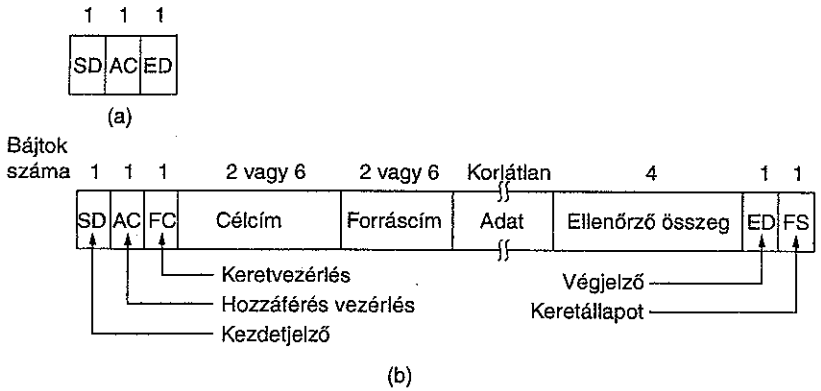
A huzalközponton belül terelő relék vannak, amelyeket az állomások látnak el árammal. Ha a gyűrű megszakad, vagy egy állomás meghibásodik, akkor a tápáram kiesése miatt a relé elenged, így az állomás kikerül a gyűrűből. A reléket szoftver is működtetheti, így lehetőség adódik olyan diagnosztikai programok írására, amelyekkel az állomások egyenkénti kiiktatása révén a hibás állomásokat, illetve gyűrűszegmenseket fel lehet térképezni. A gyűrű a rossz szegmens kiiktatása után tovább működik. Bár a 802.5 szabvány formálisan nem írja elő ennek a gyakran **csillag alakú gyűrűnek (star-shaped ring)** nevezett gyűrűtípusnak (Saltzer és mások, 1983) a használatát, de a gyakorlatban a legtöbb 802.5 LAN rendelkezik huzalközponttal a megbízhatóság és a jobb karbantarthatóság elérése érdekében.

Amikor a hálózat több, egymástól messze fekvő állomáscsoportból áll, akkor több huzalközpontból álló topológia is előállítható. Ezt úgy képzelhetjük el, hogy a 4.29. ábrán levő állomáskábelek közül az egyiket egy távoli huzalközpontba tartó kábel váltja fel. Bár logikailag az összes állomás továbbra is egyetlen gyűrűben marad, a huzalozási követelmények nagymértékben csökkenthetők. Egy huzalközpontot használó 802.5-ös gyűrű hasonló topológiával rendelkezik, mint egy 802.3-as 10Base-T elosztás hálózat, a keretformátumok és a protokollok azonban teljesen különböznek.

A vezérjeles gyűrű MAC protokollja

A MAC alréteg alapműködése nagyon egyszerű. Amikor nincs forgalom, akkor a gyűrűn egy 3 bájtos vezérjel kering körbe-körbe addig, amíg valamelyik állomás meg nem szerzi a második bájtjának egy adott bitjét, 0-ból 1-be állítva. Ezáltal az első két bájt keretkezdet szekvenciává alakul át. Az állomás ezután a 4.30. ábra szerint egy normál adatkeret további részeit kezdi el küldeni.

Rendes körülmények között a keret első bitje a gyűrűn körbefutva még azelőtt visszaér küldőjéhez, mielőtt az a teljes keretet el tudta volna küldeni. Csak egy nagyon hosszú gyűrű képes egy teljes keretet felvenni. Emiatt az adó állomásnak már küldés közben meg kell kezdenie a gyűrű „lecsapolását", vagyis az útjukat befejező bitek 4.28.(c) ábrán látható módon történő kivonását a gyűrűből.



4.30. ábra. *(a) Vezérjel felépítése. (b) Adatkeret formátuma*

Egy állomás a vezérjelet legfeljebb az ún. **vezérjeltartási ideig (token-holding time)** birtokolhatja, amely 10 ms, hacsak az üzembe helyezéskor nem állítanak be más értéket. Ha az első keret elküldése után még elegendő idő marad, akkor az állomás további kereteket is küldhet. Ha az összes keret elküldése befejeződött, vagy lejárt a vezérjeltartási idő, akkor az állomásnak vissza kell állítania a 3 bájtos vezérjelet, és vissza kell helyeznie a gyűrűre.

A 4.30.(b) ábrán látható *Kezdetjelző (Starting delimiter)* és *Végjelző (Ending delimiter)* mezők a keretek elejét és végét jelzik. Az adatbájtoktól való megkülönböztethetőség érdekében, ezek olyan átmeneteket tartalmaznak, amelyek érvénytelenek a differenciális Manchester-kódolásban (magas-magas és alacsony-alacsony). A *Hozzáférés-vezérlő (access control)* mező tartalmazza a vezérjelbitet (token bit), a *Figyelőbitet (monitor bit)*, a *Prioritásbiteket*, valamint a *Lefoglalási biteket (reservation bits)*, amelyeket a következőkben ismertetünk. Az adatkereteket a vezérlő keretektől a *Keretvezérlés (frame control)* mező különbözteti meg.

Ezeket a *Célcím* és a *Forráscím* mezők követik, amelyek ugyanolyanok, mint a 802.3-ban és 802.4-ben. Ezután következhetnek az adatok, amelyek tetszőleges hosszúságúak lehetnek. Az adatmező hosszát csak a vezérjeltartási idő korlátozza. Az *Ellenőrző összeg* mező *(checksum)* tartalma megegyezik a 802.3-aséval és a 802.4-esével.

Egy érdekes, a másik protokollokban nem szereplő mező a *Keretállapot (frame status)* mező. Ez az *A* és *C* biteket tartalmazza. Amikor egy keret megérkezik a célcímmel megegyező állomás csatolójára, a keret áthaladása során az interfész logikai 1-be billenti az *A* bitet. Ha még be is másolja a keretet az állomás memóriájába, akkor a *C* bitet is bebillenti. A bemásolás pufferhiány, vagy egyéb okokból meghiúsulhat.

Amikor a célállomás kivonja az általa küldött keretet a gyűrűről, megvizsgálja az *A* és *C* biteket. Három kombináció lehetséges:

1. $A = 0$ és $C = 0$: a célállomás nem létezik, vagy nincs bekapcsolva.

2. $A = 1$ és $C = 0$: az állomás létezik, de nem tudta fogadni a keretet.

3. $A = 1$ és $C = 1$: az állomás létezik és bemásolta a keretet.

Ez a megvalósítás a keretek egyidejű nyugtázását is jelenti. Ha egy keretet visszautasítanak, de a cél létezik, akkor a küldő opcionálisan egy kis idő elteltével újból próbálkozhat. A *Keretállapot* bájt az ellenőrző összeg hatáskörén kívül van, ezért az *A* és *C* bitek megkettőzésével kompenzálták a megbízhatóság csökkenését.

A *Végjelző* mező egy *E* bitet tartalmaz, amelyet akkor billent be az illesztő, ha hibát érzékel (pl. egy nem engedélyezett Manchester-mintát fedez fel). Tartalmaz még egy olyan bitet is, amelynek segítségével egy logikailag összetartozó keretsorozat utolsó keretét lehet megjelölni, azaz hasonló jellegű, mint egy állomány-vége (end-of-file) jel.

A 802.5-ben van egy kidolgozott eljárás, amely lehetővé teszi a többszintű prioritáskezelést. A 3 bájtos vezérjel középső bájtjának egyik mezője a vezérjel prioritását adja meg. Amikor egy állomás egy *n* prioritású keretet akar küldeni, akkor addig kell várnia, amíg el nem tud kapni egy olyan vezérjelet, amelynek prioritása legfeljebb *n*-nel egyenlő. Ezen kívül egy állomás a következő vezérjel lefoglalását megkísérelheti úgy is, hogy az éppen áthaladó keret *lefoglalásbitjeit* olyan prioritásúra írja át, amilyen prioritású keretet el szeretne küldeni. Ha azonban ezekbe a bitekbe már nagyobb prioritást jegyeztek be, akkor az állomás lefoglalási kísérlete sikertelen lesz. Az aktuális keret elküldését követően a visszahelyezett vezérjel prioritásának meg kell egyeznie azzal a prioritással, amelyet már korábban lefoglaltak.

Kis gondolkodással belátható, hogy ez a mechanizmus egyre följebb és följebb emeli a lefoglalási prioritást. A probléma megoldására a protokoll néhány összetettebb szabályt fogalmaz meg. A gondolat lényege az, hogy egy prioritást növelő állomás, az emelés végrehajtását követően a prioritáscsökkentés felelősévé válik.

Vegyük észre, hogy ez a prioritási rendszer alapvetően eltér a vezérjeles sín sémájától, amelyben minden állomás azonos módon osztozik az elérhető sávszélességen, nem számít, hogy a többi állomás mit csinál. A vezérjeles gyűrűben az alacsony prioritású keretekre a kiéheztetés veszélye leselkedik, hiszen sóvárogva várják, hogy végre feltűnjön egy alacsony prioritású vezérjel is. Világos, hogy a két bizottságnak eltérőek voltak a céljaik, amikor a szolgálatok kialakításánál a magas prioritású forgalom jó kiszolgálása, és az állomások egyenlő esélye között kellett dönteniük.

Gyűrűkarbantartás

A vezérjeles sín protokollja jelentős lépés a gyűrű decentralizált karbantartása terén. A vezérjeles gyűrű protokollja a gyűrű karbantartását teljesen másképpen oldja meg. Minden gyűrűben van egy **felügyelő állomás (monitor station)**, amely a gyűrű karbantartásáért felelős. Ha a felügyelő állomás meghibásodik, akkor a helyébe egy versenyprotokoll alapján gyorsan megválasztott másik állomás lép. (Minden állomásnak meg van az esélye, hogy felügyelővé váljon.) Amíg azonban a felügyelő állomás megfelelően működik, egyedül felelős a gyűrű helyes működéséért.

Amikor a gyűrű beindul, vagy bármelyik állomás észreveszi, hogy nincs felügyelő, egy CLAIM TOKEN vezérlőjelet küldhet el. Ha ez a keret anélkül ér vissza a feladóhoz,

hogy másik állomás ugyancsak CLAIM TOKEN keretet küldött volna, maga a küldő válik felügyelővé (minden állomásba beépítik a felügyelővé válás képességét). A vezérjeles gyűrű vezérlőkereteit a 4.31. ábra sorolja fel.

A felügyelő felelős többek között a vezérjel elvesztésének figyeléséért, a gyűrűszakadáskor elvégzendő teendők elvégzéséért, a hibás keretek eltávolításáért és az árván maradt keretek kiszűréséért. Árvakeret akkor keletkezik, amikor egy állomás egy rövid keretet teljes egészében kibocsát, de annak kivonását már nem tudja elvégezni, mert időközben meghibásodott vagy kikapcsolták. Ha erre a rendszer nem figyelne, akkor a keret a végtelenségig körözne.

A vezérjel elvesztését a felügyelő állomás egy, a lehetséges legrosszabb vezérjel nélküli intervallum értékére beállított időzítéssel ellenőrzi. Ezt abból a feltételezésből számítja ki, hogy minden állomás teljes vezérjeltartási idejét kihasználva forgalmaz. Ha ez az időzítés lejár, akkor a felügyelő megtisztítja a gyűrűt, és egy új vezérjelet állít elő.

A hibás kereteket érvénytelen formátumuk, vagy helytelen ellenőrző összegük révén lehet felismerni. A felügyelő ekkor felnyitja, majd kipucolja a gyűrűt. A gyűrű megtisztítása után új vezérjelet bocsát ki. Végül az árva kereteket úgy szűri ki, hogy minden áthaladó keret *hozzáférés-vezérlés* mezőjében bebillenti a *felügyelő bitet (monitorbit)*. Ha egy bejövő keretben ez a bit már be van állítva, akkor ez arra hívja fel a figyelmet, hogy a keret eltávolításáért felelős állomás valószínűleg hibás, hiszen csak így fordulhat elő, hogy a keret már másodszor kerül a felügyelőhöz. A felügyelő állomás ekkor maga távolítja el a keretet.

Az egyik monitorfunkció a gyűrű hosszával kapcsolatos. A vezérjel 24 bit hosszúságú, ami azt jelenti, hogy a gyűrűnek elég hosszúnak kell lennie ahhoz, hogy 24 bitet egyszerre tartalmazhasson. Ha az állomások 1 bites késleltetése, plusz a kábel terjedési késleltetése kisebb mint 24 bit, akkor a felügyelő külön késleltetésekkel biztosítja a vezérjel keringethetőségét.

A gyűrű szakadási helyének behatárolását a felügyelő állomás nem képes egyedül megoldani. Amikor egy állomás valamelyik szomszédját működésképtelennek érzékeli, akkor egy BEACON keretet bocsát ki, amelyben megadja a feltételezhetően hibás állomás címét. E keret azután körbeérve tartalmazni fogja azoknak az állomásoknak a

Keretvezérlő mező	Név	Jelentése
00000000	Duplicate address test	Ellenőrzi, hogy két állomásnak azonos-e a címe
00000010	Beacon	A gyűrűszakadások megkeresésére szolgál
00000011	Claim token	Próbálkozás felügyelővé válásra
00000100	Purge	A gyűrű újraindítása
00000101	Active monitor present	A felügyelő periodikusan bocsátja ki
00000110	Standby monitor present	Potenciális felügyelő jelenlétét jelzi

4.31. ábra. *Vezérjeles gyűrű vezérlőkeretei*

címeit, amelyek feltehetően hibásak, így ezek a huzalközpontban, a terelőrelék segítségével, emberi beavatkozás nélkül iktathatók ki a gyűrűből.

Tanulságos összehasonlítani a vezérjeles gyűrű és a vezérjeles sín vezérlési filozófiáját. A 802.4 bizottság félt bármilyen központi komponens alkalmazásától, amelynek bármilyen véletlenszerű meghibásodása miatt az egész rendszer működésképtelenné válhat. Emiatt olyan rendszert terveztek, amelyben a vezérjel aktuális birtokosa különleges jogokkal van felruházva (pl. új állomásokat vehet fel a gyűrűbe), egyébként azonban egyik állomás sem különbözhet a többitől (pl. a karbantartás érdekében aktuálisan hozzárendelt adminisztrációs felelősség).

A 802.5 bizottság ellenben úgy érezte, hogy a vezérjelvesztés, árvakeretek stb. kezelése központi felügyelő állomással sokkal egyszerűbb. Továbbá egy normál rendszerben az állomások csak nagyon ritkán mennek tönkre, ezért az új felügyelő állomás alkalomszerű versengéses megválasztása nem okoz nagy nehézséget. Ennek ára azonban az, hogy ha a felügyelő állomás egyszer valóban meghibásodik, de ugyanakkor ACTIVE MONITOR PRESENT jeleket küld periodikusan, akkor nincs olyan állomás, amely ezt megkérdőjelezhetné. A felügyelő állomást nem lehet felelősségre vonni.

Ez a megközelítésbeli különbség annak köszönhető, hogy a bizottságok eltérő alkalmazási területekben gondolkodtak. A 802.4 bizottság nagy gyárakban gondolkodott, amelyekben hatalmas fémtömegek számítógépek által felügyelt mozgatása folyik. A hálózati hibák súlyos károkat okozhatnak, amelyeket mindenáron meg kell akadályozni. A 802.5 bizottság ezzel szemben az irodaautomatizálásban volt érdekelt, ahol a néha előforduló hibákat az egyszerűbb rendszer alacsonyabb ára kompenzálja. Az azonban, hogy a 802.4 tényleg megbízhatóbb-e mint a 802.5, vita tárgyát képezi.

4.3.4. A helyi hálózatok összehasonlítása

A három különböző, és egymással nem kompatíbilis LAN különböző tulajdonságokkal rendelkezik, így sok szervezet kerül szembe azzal a kérdéssel: „Melyiket válasszuk?" Ebben a részben végignézzük a három 802-es LAN szabványt, kiemeljük előnyeiket és hátrányaikat, összehasonlítjuk és szembeállítjuk őket.

Mindjárt induláskor érdemes megjegyezni, hogy a három LAN szabvány durván ugyanazt a technológiát használja, és közel ugyanazt a teljesítményt nyújtja. Bár a számítógép-tudomány jeles képviselői és mérnökei órákat vitatkozhatnak a koaxiális kontra sodort érpár kérdésen, ez aligha érdekli a piaci felhasználókat, legyenek azok egyének, hivatalok vagy gyárak.

Kezdjük a 802.3 előnyeivel! Jelenleg a legelterjedtebben használt hálózattípus, rengeteg üzemel szerte a világon, és működtetésével kapcsolatban nagy mennyiségű tapasztalat halmozódott fel. Az algoritmus egyszerű. Az állomásokat üzem közben, a hálózat leállása nélkül lehet a hálózatba kötni. A használt kábel passzív, modemre nincs szükség. Ráadásul kis terhelés mellett a késleltetés gyakorlatilag nulla (az állomásoknak nem kell vezérjelre várniuk, a keret összeállítása után azonnal adhatnak).

Másrészről azonban a 802.3-nak alapvető analóg komponensei vannak. Minden állomásnak képesnek kell lennie a leggyengébben adó állomás jeleinek érzékelésére még adás közben is. Az adó-vevőkben levő összes, ütközést érzékelő áramkör analóg.

Mivel a keretek küldése az ütközések következtében megszakadhat, ezért a keretek minimális hossza 64 bájt, amely 1 bájtos adatok átvitelekor (pl. terminálok forgalma) jelentős felesleges terhelést jelent.

Mindezeken túl a 802.3 nem determinisztikus, ami sok esetben alkalmatlanná teszi valós idejű feladatok elvégzésére [bár bizonyos valós idejű feladatok futtathatók a vezérjeles gyűrű szoftverének szimulálásával (Venkatramani és Chiueh, 1995)]. Nincsenek prioritások definiálva. A kábel hossza legfeljebb 2,5 km lehet (10 Mb/s esetén), mivel a jel körbejárási késleltetése meghatározza a résidőt, és ezáltal a teljesítőképességet is. A CSMA/CD hálózatok, mint amilyen a 802.3 is, nagy sebesség mellett történő üzemeltetése nehéz, mert a sebesség növelésével a hatékonyság csökken, mivel a keret átviteli ideje csökken, de a versengési intervallum hossza nem (az adatátviteli sebességtől függetlenül 2τ a rés szélessége). Az is lehetséges, hogy a kábel hosszát csökkentik. Alapvető problémává válik nagy terhelés mellett az ütközések jelenléte, amely komolyan befolyásolhatja a hálózat átbocsátóképességét.

Tekintsük most a 802.4-et, a vezérjeles sínt! Kábeltelevíziós eszközöket használ, amelyek nagyon megbízhatóak és könnyen, raktárról beszerezhetőek. Sokkal determinisztikusabb a 802.3-nál, bár a vezérjel kritikus időpillanatban történő elvesztése több bizonytalanságot idézhet elő, mint amennyit támogatói elismernek. Kis minimális keretméreteket is képes kezelni.

A vezérjeles sín a prioritásokat is támogatja, és úgy konfigurálható, hogy a sávszélesség egy részét a magas prioritású forgalomnak lehet lefoglalni (pl. digitális hangátvitel). Nagy terhelés mellett is kitűnő az áteresztőképessége és a hatékonysága, mivel gyakorlatilag TDM rendszerré válik. Végül, a szélessávú kábel több csatornát is biztosít, így nem csak adat, hanem hang- és televíziós csatornákat is kialakíthatunk rajta.

A másik oldalról viszont a szélessávú rendszerek sok analóg áramköri tervezést igényelnek, valamint modemeket és nagy sávszélességű erősítőket tartalmaznak. A protokoll rendkívül összetett, és kis terhelés esetén jelentős késleltetéssel rendelkezik (az állomásoknak még tétlen rendszer esetén is meg kell várniuk a vezérjelet). Végül, nem nagyon alkalmasak üvegszálas átvitel megvalósítására, és felhasználói tábora is kicsi.

Most vegyük szemügyre a vezérjeles gyűrűt! Kétpontos összeköttetéseket használ, ami azt jelenti, hogy tervezése könnyű és teljesen digitális lehet. A gyűrű felépítéséhez bármilyen átviteli közeg alkalmazható, a postagalambtól egészen az üvegszálig. A szabványos sodrott érpár olcsóvá és egyszerűvé teszi a gyűrűk telepítését. A huzalközpont bevezetésével a vezérjeles gyűrű az egyetlen LAN, amely a kábelhibákat ki tudja mutatni, és azokat automatikusan képes kijavítani.

A vezérjeles sínekhez hasonlóan prioritások kijelölése itt is lehetséges, bár a megvalósítás nem biztosít egyenlő esélyeket a különböző prioritási szinten levő állomásoknak. A rövid keretek itt is átvihetők, de eltérően a vezérjeles síntől, akármilyen hosszúak is lehetnek, hosszukat mindössze a vezérjel-átadási idő korlátozza. Végül, nagy terhelés mellett kiválóan alakulnak átbocsátó képességi és hatékonysági jellemzői éppúgy, mint a vezérjeles síné.

Legnagyobb negatívuma a centralizált felügyelő állomás, amely a rendszer legkritikusabb alkotóeleme. Még ha egy hibás felügyelő állomás helyettesíthető is, működésképtelenné válása sok fejfájást okozhat. Továbbá, a többi vezérjeles rendszerhez hasonlóan, alacsony terhelés esetén a késleltetés jelentős lehet, mert a küldőnek meg kell várnia a vezérjelet.

Érdemes megjegyezni, hogy a három LAN-nal kapcsolatban rengeteg tanulmány jelent meg. A legfontosabb következtetés, amelyet ezekből a tanulmányokból levonható az az, hogy nem vonhatunk le semmilyen következtetést belőlük. Bárki találhat olyan paramétereket, amelyek tekintetében az egyik LAN jobbnak látszik a másiknál. A legtöbb környezetben mindhárom jól szerepel, ezért a választás során a teljesítőképesség és hatékonyság helyett inkább más paraméterek játsszák a döntő szerepet.

4.3.5. Az IEEE 802.6 szabvány: kettős sín osztott várakozási sorral

Az eddig tanult 802-es LAN-ok közül egyik sem alkalmas MAN kialakítására. A kábelhossz korlátozottsága, valamint a teljesítőképességbeli problémák, amikor több ezer állomás kapcsolódik egymáshoz, a LAN-okat legfeljebb egyetemi méretű hálózatok kialakítására teszik alkalmassá. Olyan hálózatok számára, amelyek egy teljes város kiszolgálását tűzik ki célul, az IEEE a 802.6 szabványt definiálta, amelyet **DQDB-nek (Distributed Queue Dual Bus – kettős sín osztott várakozási sorral)** neveznek. A következőkben megnézzük, hogyan működik. További információk (Kessler és Train, 1992). Az irodalomjegyzék 171 cikket sorol fel a DQDB-vel kapcsolatban (Sadiku és Arvind, 1994).

Egy 802.6 hálózat alapvető felépítését az 1.4. ábra hivatott szemléltetni. Két párhuzamos, egyirányú sín kígyózik keresztül a városon, amelyekhez párhuzamosan csatlakoznak az állomások. Mindkét vezetékhez egy-egy fejállomás csatlakozik, amely 53 bájtos cellák folyamát állítják elő. A cellák a fejállomásoktól kezdve végighaladnak a kábeleken. Amint elérik a sín végét, lepottyannak és eltűnnek.

Mindegyik cella egy 44 bájtos adattömböt szállít, aminek köszönhetően több AAL modellhez is biztosított az illeszthetőség. Mindegyik cellában található két protokollbit is: a *Foglalt (Busy)* bit jelzi, hogy foglalt-e a cella, és a *Kérelem (Request)*, amelylyel az állomások a kéréseiket jelezhetik.

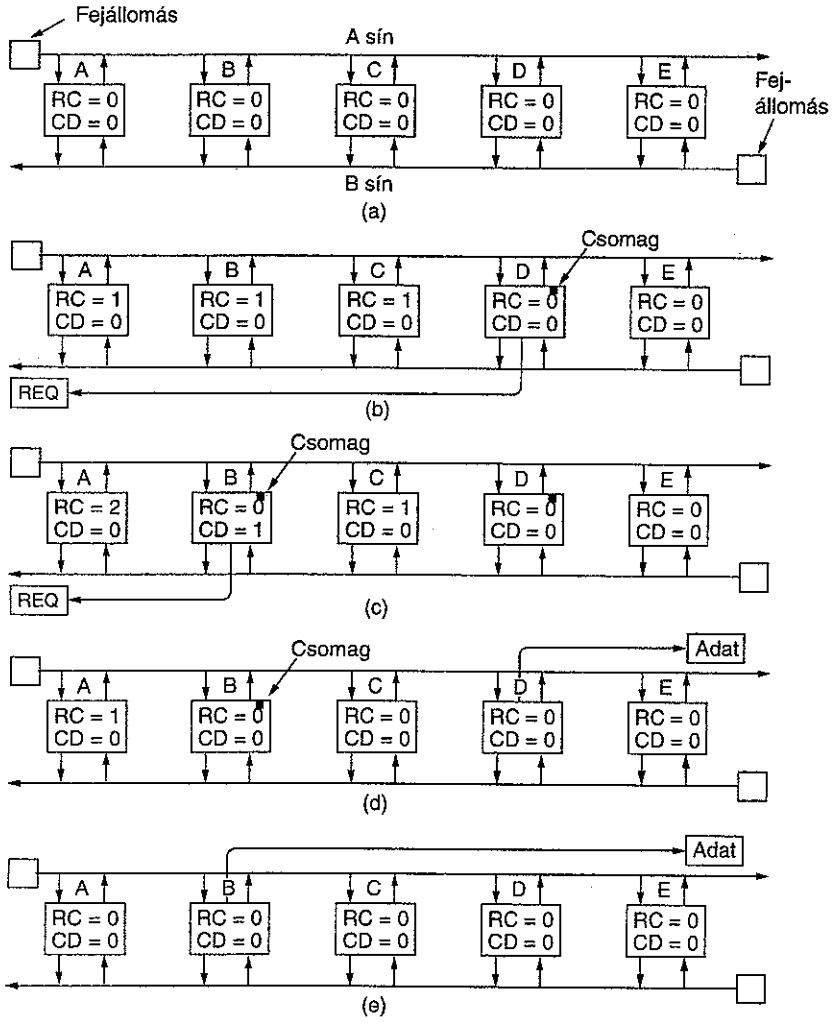
Ha egy állomás el akar küldeni egy cellát, akkor tudnia kell, hogy tőle jobbra vagy balra található-e a célállomás. Ha jobbra van, akkor a küldő az *A* sínt fogja használni, míg ha balra, akkor a *B* sínt. Bármelyik csatornára is ír adatot, az logikai VAGY kapcsolatba lép a csatorna aktuális jelével, így az egyes állomások meghibásodása nem veszélyezteti a teljes rendszer működését.

A többi 802-es LAN protokollal ellentétben a 802.6 nem mohó. A többi esetben ha egy állomásnak lehetősége adódott adatok küldésére, akkor élt is a lehetőséggel. Itt, az állomások sorba rendeződnek aszerint, hogy mikor kerültek küldésre kész állapotba, és FIFO rendszer szerint forgalmaznak. A protokoll igazán érdekes része az, hogy hogyan állítják elő a FIFO sorrendet, miközben a rendszerben nincs központi sor.

Az alapszabály az, hogy az állomásoknak udvariasnak kell lenniük: alkalmazkodnak azokhoz az állomásokhoz, amelyek az adatfolyam szemszögéből tőlük lejjebb vannak. Az illedelmességnek azért van fontos szerepe, mert a fejállomáshoz legközelebb levő állomás egyszerűen lecsaphatna az összes üres cellára, majd feltöltve azokat,

a tőle lejjebb levőket teljesen kiéheztethetné. Az egyszerűség kedvéért csak az *A* sín forgalmát fogjuk vizsgálni, de pontosan ugyanazok igazak a *B*-re is.

A FIFO sor szimulálásához minden állomás két számlálót kezel, *RC*-t és *CD*-t. *RC (Request Counter – igényszámláló)* azt számolja, hogy addig, amíg az adott állomáson meg nem jelenik egy elküldendő keret, mennyi olyan függőben maradt kérés keletkezik, amelyek feladója a folyamon lejjebb helyezkedik el. Ekkor *RC* tartalmát az állomás átmásolja *CD*-be, majd nullázza *RC*-t, amelyben már azt fogja számolni, hogy mennyi kérés érkezik az állomás adásra kész állapotba kerülése után. Például, ha a *k* állomás számlálói *CD* = 3 és *RC* = 2 értékeken állnak, akkor a következő három üres cella tovább mehet az állomás mellett, fenntartva a hátrébb levő állomások számára. Ezután a soron következő üres cellában ez az állomás is adhat, majd pedig az ezt követő két üres cella ismét a lejjebb levő állomásoké lesz. Az egyszerűség kedvéért tegyük fel, hogy minden állomásnak egyszerre csak egyetlen küldésre váró cellája lehet.



4.32. ábra. *(a) Kezdetben a MAN üres. (b) D igénybejelentése után. (c) B igénybejelentése után. (d) Miután D küldött. (e) Miután B küldött*

Ahhoz, hogy egy állomás cellát küldhessen, az ellentétes irányú sín (vagyis a *B* sínen, ha aztán az *A*-n szeretne forgalmazni) egyik cellájának *Igénybejelentés (Request)* bitjét be kell állítania. Ahogy ez a cella végigfut az ellentétes irányú sínen, minden olyan állomás, amely mellett elhalad észleli és megnöveli eggyel *RC* számlálóját. Az elképzelés illusztrálásához vegyünk egy példát! Kezdetben legyen minden *RC* számláló értéke 0, és egyetlen cella se álljon sorba. Ezt az állapotot mutatja a 4.32.(a) ábra. Ekkor a *D* állomás bejelenti igényét, amely a 4.32.(b) ábrának megfelelő módon arra készteti a *C*, *B* és *A* állomásokat, hogy növeljék meg eggyel *RC* számlálóikat. Miután *B* is elküldte igénybejelentését elmentve aktuális *RC* értékét *CD*-be, a 4.32.(c) ábrának megfelelő helyzet alakul ki.

Ez lesz az a pont, amikor az *A* sín fejállomása előállít egy üres keretet. Amikor az elhalad *B* mellett, az állomás látja, hogy a *CD* > 0, így nem használhatja fel a cellát. (Amikor egy állomás küldésre kész cellával rendelkezik, *CD* a sorban elfoglalt sorszámát jelképezi, amelynek 0 értéke esetén áll az állomás a sor legelején.) Ehelyett inkább csökkenti eggyel *CD*-jének értékét. Amikor a még mindig üres cella eljut a *D* állomáshoz, az a *CD* = 0 értéket látva tudja, hogy már senki sincs a sorban előtte, így adatait logikai VAGY kapcsolat használatával beírja a cellába és bebillenti annak *Foglalt (Busy)* bitjét. Az átvitel lebonyolítása után a 4.32.(d) ábrán látható helyzet fog kialakulni.

Amikor előáll a következő üres cella, *B* fogja látni, hogy a sor elejére került, így használatba veszi a cellát (bebillentve *Foglalt* bitjét), ahogyan azt a 4.32.(e) ábra mutatja. Ilyen módon az állomások úgy képesek sorba állni, hogy ténylegesen nem is létezik központi sorkezelő.

A mostanában telepített DQDB rendszerek már több vivővel is rendelkeznek, és egész városokat ölelnek fel. Tipikusan 160 km hosszúak, és 44,736 Mb/s (T3) sebességgel üzemelnek.

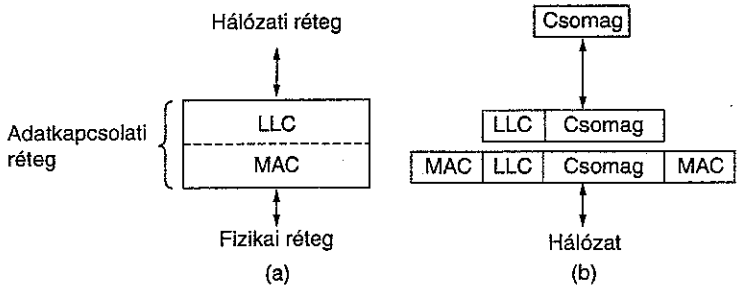
4.3.6. Az IEEE 802.2 szabvány: logikai kapcsolatvezérlés

Talán most elérkezett az ideje, hogy egy lépést visszalépve összehasonlítsuk az ebben a fejezetben eddig tanultakat az előző fejezettel. A 3. fejezetben megvizsgáltuk, hogy két állomás hogyan tud egymással megbízható módon kommunikálni egy megbízhatatlan vonalon keresztül különböző adatkapcsolati protokollok használatával. Ezek a protokollok hibajavítást (nyugtázások használatával) és forgalomszabályozást (csúszóablak használatával) biztosítottak.

Ezzel szemben, ebben a fejezetben egy árva szót sem szóltunk eddig a megbízható kommunikációról. A 802-es LAN és MAN hálózatok mindössze annyit ígérnek, hogy mindent elkövetnek a datagram szolgálat biztosításáért. Sokszor ennyi elegendő is.

Például IP csomagok továbbításához nincs szükség semmilyen garanciára, de még csak nem is számít erre senki. Elegendő, ha az IP csomagot beletesszük egy 802-es keret adatmezőjébe, és útjára bocsátjuk. Ha elveszik, hát elveszik.

Léteznek azonban rendszerek, amelyek olyan protokollt igényelnek, amelyik rendelkezik hibajavítással és forgalomszabályozással. Az IEEE definiált egy ilyet, amely a 802-es LAN és MAN protokollokra épül. Ráadásul, ez a protokoll, amelyet **LLC-nek (Logical Link Control – logikai kapcsolatvezérlés)** neveznek, teljesen eltakarja a különböző 802-es hálózatokat azzal, hogy egységes formátumot és felületet biztosít a hálózati rétegek számára. Ez a formátum, felület, illetve protokoll erőteljesen az OSI-ra épül. Ahogyan a 4.33. ábra is mutatja, az LLC az adatkapcsolati réteg felső felét tölti ki, amely alatt a MAC alréteg helyezkedik el.



4.33. ábra. *(a) Az LLC elhelyezkedése. (b) Keret formátumok*

Az LLC-t tipikusan a következő módon használják. A küldő számítógép hálózati rétege az LLC szolgálati primitívjeinek felhasználásával átad egy csomagot az LLC rétegnek. Az LLC réteg kiegészíti ezt egy LLC fejrésszel, amely sorszámot és nyugtaszámot tartalmaz. Az így nyert adatstruktúra ezután bekerül egy 802.x keret adatmezejébe, és továbbításra kerül. A vevőnél a folyamat megfordítottja zajlik le.

Az LLC három szolgálati lehetőséget biztosít: megbízhatatlan datagram szolgálat, visszaigazolt datagram szolgálat és megbízható, összeköttetés alapú szolgálat. Az LLC fejrésze a korábbi HDLC protokollon alapul. Több, egymástól különböző formátum is használatos az adatok és vezérlőinformációk átadására. A visszaigazolt datagram és összeköttetés alapú szolgálatok esetén az adatkeretek tartalmazzák a cél- és forráscímeket, egy sorszámot, egy nyugtaszámot, valamint néhány egyéb bitet. Megbízhatatlan datagram szolgálat mellett a sorszámot és a nyugtaszámot elhagyják.

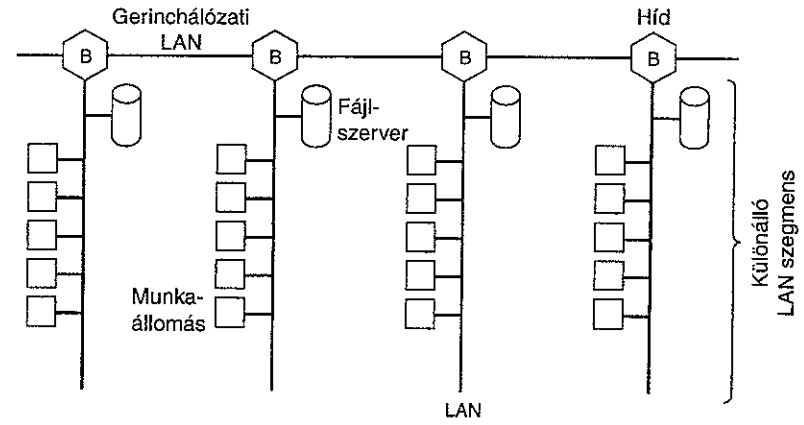
4.4. Hidak

Sok szervezet rendelkezik több LAN-nal, amelyeket szeretne egymással összekötni. Különböző helyi hálózatokat **hidakkal (bridges)** lehet összekötni, amelyek az adatkapcsolati rétegben működnek. Ez annyit jelent, hogy a hidak nem vizsgálják a hálózati réteg fejrészeit, így egyaránt képesek IP, IPX és OSI csomagok átjátszására. Ezzel szemben egy egyszerű IP, IPX vagy OSI router csak a saját csomagjait képes kezelni.

A következő részekben végignézzük a hidak felépítését, és különös figyelmet fogunk szentelni a 802.3, 802.4 és 802.5 LAN-ok összekapcsolásának. A hidak részletes leírásáról és az idecsatlakozó témákról bővebb információ a szakirodalomban található (Perlman, 1992). Mielőtt megismerkednénk a hidak technológiájával, érdemes végignézni néhány olyan helyzetet, amelyek esetében hidakat használnak. Hat okot is felsorolunk arra, hogy miért is születnének különálló helyi hálózatok egy-egy szervezeten belül. Először is, sok egyetem és egyéb szervezet egységeinek saját LAN-ja van, elsősorban azért, hogy számítógépeiket, munkaállomásaikat és szervereiket összekösse. Mivel a különböző egységek más és más feladatokat látnak el, elképzelhető, hogy különböző típusú LAN-okat választanak maguknak függetlenül attól, hogy más egységek hogyan döntenek. Előbb vagy utóbb szükség lesz az együttműködésre, így hidakra lesz szükség. Ebben a példában a hálózatok tulajdonosainak autonómiája okozta a többféle helyi hálózat létrejöttét.

A második esetben a szervezet szétszórtan, egymástól akár nagy távolságokra elhelyezkedő épületekben helyezkedhet el. Olcsóbb lehet, ha az épületekben különálló LAN-okat telepítenek, és ezeket hidak és infravörös átvitel segítségével kötik össze ahelyett, hogy egy koaxiális kábelt vezetnének végig a teljes területen.

Harmadik eset az, amikor a terhelés megosztása indokolja az egyetlen LAN több szegmensre bontását. Több egyetemen például több ezer munkaállomás áll a hallgatói és oktatói célokra. A fájlokat általában fájlszervereken tárolják, ahonnan igény szerint tölthetők le. A rendszer hatalmas mérete egyszerűen lehetetlenné teszi egyetlen LAN használatát, mivel a szükséges sávszélesség óriási lenne. Ehelyett a 4.34. ábrán bemutatott struktúra szerint, kisebb méretű helyi hálózatokat alakítanak ki, amelyeket hidak segítségével kapcsolnak össze. Mindegyik különálló LAN a munkaállomások egy cso-



4.34. ábra. *Több, gerinchálózattal összekapcsolt LAN, amelyen a teljes terhelés nagyobb lehet, mint egy különálló LAN kapacitása*

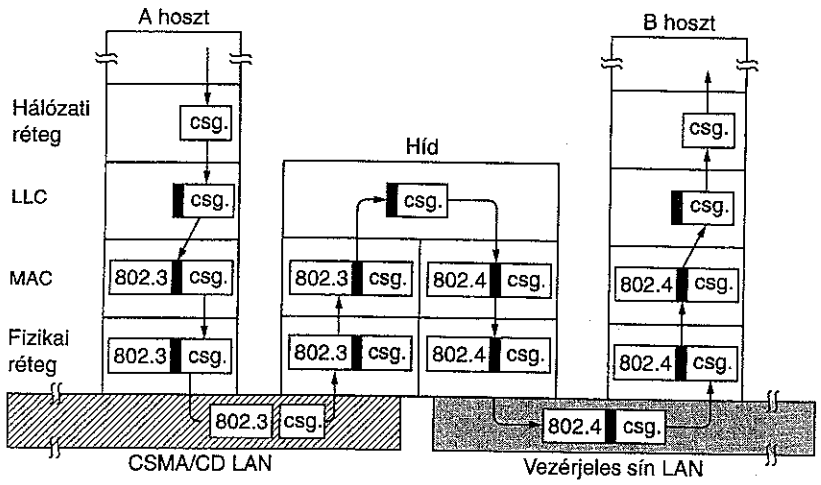
portjából áll, és saját fájlszerverrel rendelkezik, így a forgalom nagy része a különálló szegmensekre korlátozódik anélkül, hogy a forgalom megjelenne a gerinchálózaton.

Negyedszer, elképzelhető, hogy a különálló LAN megfelelő megoldást jelentene a terhelés szempontjából, ám a két, egymástól legmesszebb eső állomás között túl nagy lehet a távolság (pl. 802.3 esetén nagyobb, mint 2,5 km). A kábel lefektetése egyszerű feladat is lehet, de a hálózat működésképtelen lesz a túl nagy körbefutási idő miatt. Az egyetlen megoldás, ha feldaraboljuk a LAN-t, és a szegmenseket hidakkal kötjük össze. Hidak használatával a teljes hálózat által lefedett távolság megnövelhető.

Ötödiknek vegyük a megbízhatóság problémáját. Egy különálló LAN esetén egy meghibásodott állomás, amely folyamatosan szemetet küldözget, megbéníthatja a teljes rendszert. Ha a kritikus pontokon hidakat iktatunk be úgy, mint egy épületben a tűzzáró ajtókat, megelőzhető a hálózatnak egy hibás állomás miatt történő összeomlása. Szemben az ismétlőkkel, amelyek mindent után átmásolnak, a hidak felprogramozhatók, hogy bizonyos kritériumokat megvizsgálva döntsék el, hogy mit továbbítanak, és mit nem.

A hatodik, és egyben utolsó példánk szerint a hidak részét képezhetik egyes szervezetek védelmi rendszerének. A legtöbb LAN illesztő rendelkezik azzal a képességgel, hogy ún. **válogatás nélküli üzemmódban (promiscuous mode)** üzemeljen, amikor is minden keretet továbbít a számítógépnek, és nem csak azokat, amelyeket annak az állomásnak címeztek. A kémek és a minden lében kanál emberek nagyon szeretik ezt a funkciót. Ha hidakat iktatunk be a hálózat különböző pontjaira, és azokat úgy állítjuk be, hogy az érzékeny forgalmat ne továbbítsák, akkor megoldható a hálózat egyes részeinek az izolálása, amelynek bizalmas forgalma nem tud kiszökni, így nem juthat rossz kezekbe.

Most, hogy láttuk miért hasznosak a hidak, vizsgáljuk meg, hogyan működnek! A 4.35. ábra egy egyszerű, kétportos híd működését szemlélteti. A csomag leereszkedik az LLC alrétegig, ahol LLC fejrésszel látják el. Ekkor átkerül a MAC alrétegbe, ahol



4.35. ábra. *Egy 802.3 és egy 802.4 hálózatot összekötő híd működési elve*

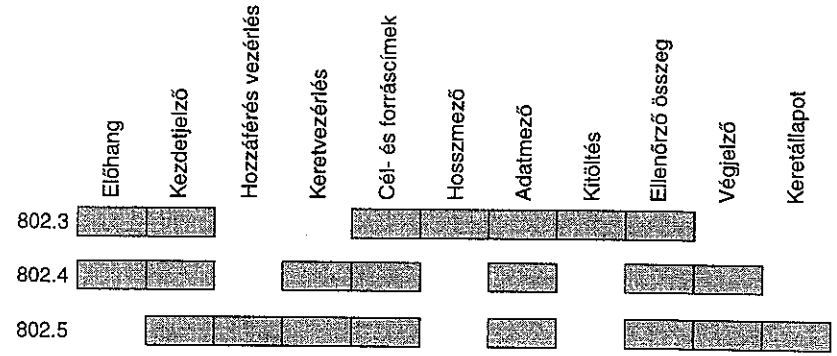
egy 802.3-as fejrésszel (és lábrésszel is, amelyet nem ábrázoltunk) látják el. Az üzenetegység kikerül a kábelre és történetesen a híd MAC alrétegében köt ki, ahol eltávolítják róla a 802.3-as fejrészt. A lecsupaszított csomag (az LLC fejrésszel együtt) átkerül a híd LLC alrétegébe. Példánkban a csomag címzettje egy 802.4-es hálózatban található, így a híd 802.4-es oldalán folytatja útját lefelé, majd kikerül a másik hálózatra. Vegyük észre, hogy egy híd, amely *k* különböző LAN-t köt össze, *k* különböző MAC alréteggel, és *k* különböző fizikai réteggel kell rendelkezzen, mindegyik felé eggyel.

4.4.1. Hidak 802.x és 802.y hálózatok között

Naivan azt gondolhatnánk, hogy teljesen triviális két 802-es LAN közé hidat készíteni. Sajnos nem így áll a helyzet. A szakasz további részeiben azt tárgyaljuk, hogy milyen nehézségek merülhetnek fel, amikor két különböző 802-es LAN számára próbálunk hidat készíteni.

A kilenc darab 802.x–802.y párosítás közül mindegyik egyedi problémákkal rendelkezik. Mindazonáltal nézzük meg először azokat a problémákat, amelyekkel minden híd esetén szembekerülhetünk, és csak ezután vegyük sorra a kombinációk speciális problémáit. Kezdjük azzal, hogy minden LAN saját keretszerkezettel rendelkezik (lásd 4.36. ábra). Ennek az inkompatibilitásnak nincs igazi technikai magyarázata, egyszerűen csak az, hogy a három nagy cég (Xerox, GM és IBM) közül egyik sem akarta megváltoztatni a *saját* szabványát. Ennek az eredménye az, hogy a különböző LAN-ok közti másolás során a kereteket újra kell formálni, ami CPU időt igényel, és szükségessé válik új ellenőrző összeg kiszámítása, valamint behozza azt a lehetőséget, hogy a híd memóriájában esetleg előforduló néhány helytelen bit miatt észrevehetetlen hibák keletkezzenek. Ezek közül egyik sem lenne szükségszerű, ha a három bizottság meg tudott volna állapodni egy közös keret formátumban.

Egy másik probléma az, hogy az összekapcsolt hálózatok nem feltétlenül ugyanazzal az adatsebességgel működnek. Ha egymást sűrűn követő keretek hosszú sorozatát kell egy gyorsabb hálózatról egy lassabbra továbbítani, a híd nem lesz képes olyan



4.36. ábra. *Az IEEE 802 keret formátumok*

gyorsan megszabadulni a keretektől, mint ahogyan azok beérkeznek. Kénytelen pufferelni azokat reménykedve, hogy nem fogy ki a memóriából. A probléma fennáll akkor is, ha 802.4-ről 802.3-ra akarunk átjátszani 10 Mb/s-on, mivel a 802.3-as hálózat sávszélességének egy részét az ütközések emésztik fel, így valójában nem rendelkezik 10 Mb/s kapacitással, míg a 802.4 igen (legalábbis többnyire). A három vagy több LAN-t összekötő hidak szintén szembetalálhatják magukat ezzel a problémával, ha egyszerre több LAN is ugyanazt a kimenetét próbálja elárasztani.

Egy kisebb jelentőségű, de fontos probléma az, hogy a híd torlódási pontot (bottleneck) képezhet, amelyhez a felsőbb rétegek időzítései nem alkalmazkodhatnak. Gondoljunk arra, hogy egy 802.4-es LAN hálózati rétege egy nagyon hosszú üzenetet, mint keretek sorozatát szeretné továbbítani. Miután elküldte az utolsót is, beindít egy időzítőt a nyugtára való várakozáshoz. Ha az üzenetnek keresztül kell haladnia egy hídon egy lassabb 802.5 hálózat felé, akkor fennáll annak a veszélye, hogy az időzítés még azelőtt lejár, hogy az utolsó keretet továbbítani tudná a lassabb hálózat felé. A hálózati réteg azt fogja feltételezni, hogy csomagvesztés történt, ezért egyszerűen újraküldi a teljes sorozatot. Miután *n*-szer próbálkozott, fel fogja adni, és azt fogja jelenteni a szállítási rétegnek, hogy a célállomás halott.

A harmadik, és egyben potenciálisan legveszélyesebb probléma mind közül az, hogy mindhárom 802-es LAN más értékben szabja meg a maximális keretméretet. Egy 802.3 hálózat esetén a paraméterezéstől függ, de a 10 Mb/s-os szabvány szerint az adat mező maximális hossza 1500 bájt lehet. 802.4 hálózatok esetében ez fix 8191 bájt, míg 802.5 esetén nincs is felső korlát, csupán az, hogy egyetlen állomás sem adhat tovább, mint a maximális vezérjeltartási idő. Az alapértelmezett 10 ms-mal számolva a maximális keretméret 5000 bájtra adódik.

Egy nyilvánvaló probléma abból származik, ha egy olyan hosszú keretet kell továbbítani, amelyet a másik LAN nem fogadhat el. A keret feldarabolása ebben a rétegben nem oldható meg. Minden protokoll azt feltételezi, hogy a keretek vagy célba érkeznek, vagy nem. Nincs arra lehetőség, hogy a kereteket kisebb méretűekből rakjuk össze. Ez persze nem jelenti azt, hogy ilyen protokollok nem kezelhetők le. A probléma lekezelhető, és már meg is oldották, egyszerűen csak a 802 nem rendelkezik ezzel a képességgel. Alapvetően tehát nincs megoldás: azokat a kereteket, amelyek túl nagyok ahhoz, hogy továbbítani lehessen őket, el kell dobni. Ennyit az átlátszóságról (transparency).

Most vegyük sorra a kilenc 802.x–802.y párosítást, hogy megláthassuk, milyen problémák rejtőznek még az árnyékban. A 802.3-ról 802.3-ra történő továbbítás egyszerű. Az egyetlen probléma, ami felmerülhet az, ha a cél LAN annyira túlterhelt, hogy a beérkező csomagoktól nem tud a híd időben megszabadulni. Ha az ilyen helyzet elég sokáig fennáll, akkor a híd puffere lassan betelhet, és elkezdheti eldobálni a kereteket. Mivel ez a probléma mindig fennállhat, amikor 802.3-as LAN felé továbbítunk, a későbbiekben meg sem említjük többször. A másik két LAN esetén minden állomás, így a híd számára is biztosított, hogy időről időre megkapja a vezérjelet, így biztosan nem szorulhat hosszú ideig háttérbe.

802.4-ről 802.3-ra való átjátszás esetén két probléma merül fel. Először is a 802.4-es keretek rendelkeznek prioritással, míg a 802.3-asok nem. Emiatt, ha két 802.4-es LAN kommunikál egymással egy 802.3-as LAN-on keresztül, akkor a keretek prioritása el fog veszni.

A második problémát a 802.4 egy sajátságos szolgáltatása okozza: az ideiglenes vezérjelátadás. A 802.4 keretszerkezete lehetőséget nyújt egy bit beállításával, hogy a vezérjelet ideiglenesen a célállomás kapja meg azért, hogy nyugtázást küldhessen a keret vételéről. Mit tehet a híd, ha egy ilyen keretet kell továbbítania? Ha saját maga küld egy nyugtát, akkor hazudik, hiszen a keretet tulajdonképpen még nem kézbesítette. Az is elképzelhető, hogy a célállomás üzemen kívül van.

Másrészről viszont, ha nem küld nyugtát, akkor a küldő fél ismételten azt a következtetést vonhatja le, hogy a célállomás halott, így hibaüzenetet küld. Úgy tűnik, hogy ez a probléma sehogyan sem oldható meg.

Hasonló gonddal kerülünk szembe, ha 802.5-ről szeretnénk 802.3-ra átjátszani, mivel ezekben a keretekben szerepel az *A* és a *C* keretállapot bit. Ezeket a biteket a célállomás állítja be, hogy jelezze az adónak, hogy eljutott-e hozzá a keret, illetve sikeresen be tudta-e másolni azt az állomás memóriájába. Ez esetben a híd szintén hazudhat, hogy sikeres volt az átvitel, de óriási gondok adódhatnak, ha később kiderül, hogy a célállomás üzemen kívül van. A híd beiktatása teljesen megváltoztatta a bitek jelenté-

		Célhálózat		
		802.3 (CSMA/CD)	802.4 (Token bus)	802.5 (Token ring)
Forráshálózat	802.3		1, 4	1, 2, 4, 8
	802.4	1, 5, 8, 9, 10	9	1, 2, 3, 8, 9, 10
	802.5	1, 2, 5, 6, 7, 10	1, 2, 3, 6, 7	6, 7

Teendők:

1. A keret újraformálása, és az ellenőrző összeg újraszámítása.
2. Bitsorrend megfordítása.
3. Prioritás másolása értelemszerűen, vagy automatikusan.
4. Prioritás kitalálása.
5. Prioritás eldobása.
6. A gyűrű lecsapolása (valahogy).
7. Az A és C bitek beállítása (hazudni kell).
8. Aggódni kell a torlódás miatt (gyorsabb LAN-ról lassabbra való átalakítás).
9. Ideiglenes vezérjelátadás esetén a visszajelzés késhet vagy lehetetlen.
10. Pánikhelyzet, ha a keret túl nagy méretű a célhálózat számára.

Feltételezett paraméterek:

802.3: 1500 bájtos keretek, 10 Mb/s (mínusz az ütközések)
802.4: 8191 bájtos keretek, 10 Mb/s
802.5: 5000 bájtos keretek, 4 Mb/s

4.37. ábra. *802.x-ről 802.y-ra történő átjárás során felmerülő problémák*

sét. Elég nehéz elképzelni, hogy milyen megfelelő megoldást lehetne találni erre a problémára.

Ha 802.3-ról alakítunk át 802.4-re, abba az akadályba ütközünk, hogy nem tudjuk, mit írjunk a keret prioritási mezejébe. Jó elgondolás, ha a hidat úgy állítjuk be, hogy a kereteket a lehetséges legmagasabb prioritással továbbítsa, hiszen azok valószínűleg már így is elég késleltetést szenvedtek.

802.4-es hálózatok közé helyezett hidak csak az ideiglenes vezérjelátadás problémájának megoldásával kerülnek szembe. Mindenesetre a lehetőség megvan arra, hogy a keretet olyan gyorsan továbbítsuk, hogy a nyugta még időben megérkezhessen, azonban ez a megoldás ténylegesen egy lutri. Ha a híd a keretet maximális prioritással teszi át a másik hálózatra, akkor csal egy kicsit, de megnövelheti a nyugta időben történő visszaérkezésének esélyét.

Amikor 802.5-ről 802.4-re alakítunk, akkor ismételten az *A* és *C* bitek kitöltésének problémájával kerülünk szembe. Fennáll a prioritások definíciói közötti különbség problémája is, de aki éhes, ne válogasson. Örüljünk, hogy a két szabvány ugyanannyi prioritásbitet használ! A híd nem tehet többet, mint átmásolja a prioritási biteket és bizakodik.

Ha a híd 802.3-ról 802.5-re vált, akkor az egyetlen problémát a prioritásbitek kitöltése jelentheti. 802.4-ről 802.5-re alakítás esetén potenciális probléma jelentkezhet a túl hosszú keretekkel, valamint az ideiglenes vezérjelátadással kapcsolatban. Végül a 802.5-ről 802.5-re történő átjátszás esetén ismételten csak az *A* és *C* bitek kitöltése okoz problémát. A 4.37. ábra foglalja össze a különböző problémákat, amelyekről eddig beszéltünk.

Amikor az IEEE 802 bizottsága kihozta LAN szabványát, nem tudta magát egyetlen szabvány mellett elkötelezni, ezért *három,* egymással nem kompatíbilis szabványt készített, ahogy azt már részletesen láthattuk. Emiatt a hiba miatt rengeteg kritika érte a szervezetet. Amikor aztán megkapták a feladatot, hogy tervezzenek szabványt a három inkompatíbilis LAN-t összekapcsolni képes hidakhoz, már jobb munkát végeztek. Komolyan! Mindössze *kettő,* egymással inkompatíbilis hídtervezettel álltak elő. Igaz ugyan, hogy eddig még senki sem kérte fel őket, hogy készítsenek egy átjáró (gateway) szabványt, amely a két inkompatíbilis hidat képes összekapcsolni, de legalább a tendencia jó irányba halad.

Ez a szakasz arról szólt, hogy milyen problémákkal kerülünk szembe, ha két IEEE 802-es LAN-t egyetlen híddal próbálunk meg összekapcsolni. A következő két szakasz azt részletezi, hogy a sok LAN-t és sok hidat tartalmazó, összekapcsolt hálózatokkal kapcsolatban milyen problémák merülhetnek fel, valamint megvizsgáljuk azt a két IEEE megoldást, amelyet ezen hidak kialakítására kidolgoztak.

4.4.2. Transzparens hidak

Az első 802-es hídtípus a **transzparens híd (transparent bridge)** vagy **feszítőfás híd (spanning tree bridge)** (Perlman, 1992). Azoknak az embereknek, akik ezt a felépítést támogatták az elsődleges célja a teljes átlátszóság volt. Nézőpontjuk szerint biztosítani kell a több LAN-t használók számára, hogy megvásárolva egy, az IEEE

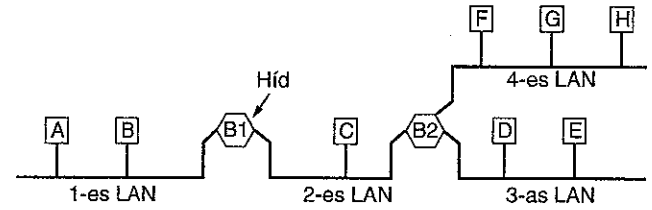
szabvány szerint készített hidat, azt egyszerűen becsatlakoztatva a hálózataik közé, minden azonnal tökéletesen működjön. Nem szabad, hogy bármilyen hardver vagy szoftver módosítást kelljen eszközölni, címzési kapcsolókat kelljen beállítgatni, forgalomirányító táblákat vagy egyéb paramétereket kelljen letölteni, semmit. Egyszerűen csak a kábeleket kelljen csatlakoztatni, és már menjen is a rendszer. Ráadásul a már meglevő hálózat működését a hidak semmilyen formában sem befolyásolhatják. Meglepő, de sikerült teljesíteniük a kikötéseket.

Egy transzparens híd mindig ún. válogatás nélküli üzemmódban működik, azaz minden keretet elkap, amely a hozzá csatlakoztatott hálózatokon megjelenik. Példaképpen vegyük a 4.38. ábrán látható elrendezést. A B1 híd az 1-es és 2-es LAN-okkal van összeköttetésben, míg a B2 a 2-es, 3-as és 4-es hálózatokkal. Ha egy keret érkezik B1-hez az 1-es LAN felől *A*-nak címezve, akkor a keret eldobható, hiszen máris a megfelelő LAN-on van, viszont ha az 1-es LAN felől *C*-nek vagy *F*-nek érkezne kerete, akkor azokat továbbítania kellene.

Amikor egy keret beérkezik egy hídhoz, annak el kell döntenie, hogy eldobhatja-e azt, vagy továbbítania kell, valamint ez utóbbi esetben azt, hogy melyik LAN irányába küldje tovább a keretet. A döntést a híd úgy hozhatja meg, hogy a memóriájában levő nagy (hash struktúrájú) listából kikeresi a célállomást. A tábla felsorolja a lehetséges célállomásokat és mindegyikhez megadja a megfelelő kimenetet (LAN-t). Például B2 táblája szerint *A* a 2-es LAN-hoz tartozik, hiszen B2-nek csak annyit kell tudnia, hogy merre továbbítsa az *A*-nak küldött kereteket. Az nem érdekli, hogy a keretet útja során kell-e még többször is továbbadni, vagy sem.

Amikor a hidakat először kapcsolják be, a tábláik üresek. Egyik híd sem tudja, hogy a célállomások merre helyezkednek el, így az elárasztásos algoritmust használják: minden bejövő keretet, amelynek címzettje ismeretlen, továbbadják az összes hozzá kapcsolódó LAN-hoz, kivéve azt, amelyiktől érkezett. Ahogy telik az idő, az alább leírtak alapján a hidak lassan megtanulják, hogy a célállomások merre találhatóak. Miután egy célállomás ismertté vált, a felé irányuló keretek már nem képezik elárasztás tárgyát, hanem csakis a megfelelő LAN felé kerülnek továbbításra.

Az az algoritmus, amit a transzparens hidak használnak, a **hátrafelé tanulás (backward learning).** Mint már említettük, a hidak válogatás nélküli üzemmódban működnek, így minden keretet látnak, amely a LAN-jaikon megjelenik. Megvizsgálva a forráscímet megállapíthatják, hogy mely LAN-okon mely állomások érhetőek el. Például, ha a 4.38. ábrán a B1 híd lát egy keretet a 2-es LAN-on, amelyik a *C* állomástól származik, rájön, hogy *C* a 2-es LAN-on keresztül érhető el, így készít a táblá-



4.38. ábra. *Egy konfiguráció, amely négy LAN-t és két hidat foglal magában*

jában egy bejegyzést, amely szerint a *C*-nek küldött kereteket a 2-es LAN felé kell irányítani. Minden olyan rákövetkező keretet, amely az 1-es LAN felől *C*-nek érkezik, továbbítani fog, míg a 2-es LAN-on *C*-nek érkezőket eldobja.

A hálózat topológiája változhat, ahogy állomásokat és hidakat üzembe vagy üzemen kívül helyeznek, az üzemelésük helyét megváltoztatják. A dinamikusan változó topológia kezelése érdekében minden alkalommal, amikor létrejön egy táblabejegyzés, eltárolják a keret beérkezésének időpontját is. Amikor egy olyan keret érkezik, amely feladójáról helyes bejegyzés szerepel a táblában, az időinformációt az aktuális időponttal írják felül. Ilyen módon a tábla bejegyzéseihez rendelt időpontok megadják, hogy a híd mikor vett utoljára keretet az adott állomástól.

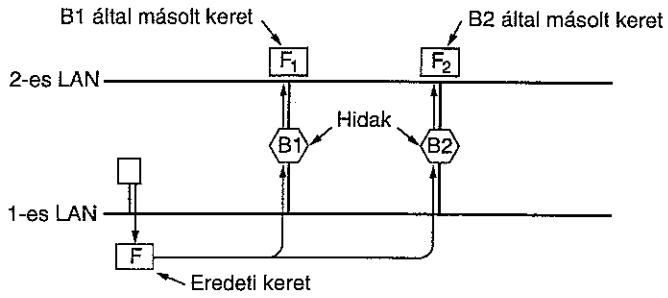
Egy folyamat a hídban időről időre végignézi a táblát, és törli onnan a néhány percnél régebbi bejegyzéseket. Ily módon elérhető, hogy kézi beavatkozás nélkül, néhány percen belül ismét visszaálljon egy olyan állomás normális működése, amelyet levettek a hálózatról, majd annak egy eltérő pontjára csatlakoztattak vissza. Az algoritmus azonban azt is maga után vonja, hogy ha egy állomás néhány percig csendben marad, a felé irányuló forgalom elárasztja a teljes rendszert addig, amíg az állomás maga nem küld végre egy keretet.

Egy beérkező keret forgalomirányítása tehát az alábbiak szerint függ attól, hogy melyik LAN felől érkezett (forrás LAN), valamint melyik LAN irányában (cél LAN) van a célja:

1. Ha a forrás és cél LAN azonos, akkor a keretet el kell dobni.
2. Ha a forrás és a cél LAN különböző, akkor a keretet továbbítani kell.
3. Ha a cél LAN ismeretlen, akkor elárasztást kell alkalmazni.

A keretek beérkezésekor mindig ezt az algoritmust kell használni. A táblában való keresésre, valamint a bejegyzések frissítésére speciális VLSI áramkörök léteznek, amelyek néhány μs alatt oldják meg ezeket a feladatokat.

A megbízhatóság növelése érdekében néhány hálózaton egymással párhuzamosan bekötött hidakat is használnak, ahogyan ezt a 4.39. ábra is szemlélteti. Ez az elrendezés azonban újabb problémákat vet fel, mivel hurkot képez a hálózat topológiájában.



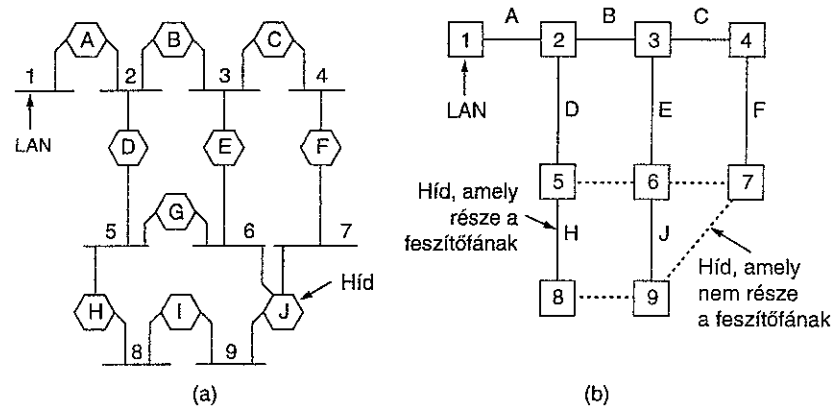
4.39. ábra. *Párhuzamosan elhelyezett transzparens hidak*

A 4.39. ábrán bemutatunk a problémára egy egyszerű példát. Nézzük meg mi történik az *F* kerettel, amelynek címzettje ismeretlen! A normál szabályokat alkalmazva az ismeretlen címzett esetére, mindkét híd elárasztást kezdeményez, ami jelen esetben mindössze azt jelenti, hogy a keretet átmásolják a 2-es LAN-ra. Nemsokára az 1-es híd felfedezi az F_2 keretet, melynek címzettje ismeretlen, így átmásolja azt az 1-es LAN-ra, így állítva elő az ábrán már nem jelölt F_3 keretet. Hasonlóan, a 2-es híd az F_1 keretet másolja át az 1-es LAN-ra, így előáll az F_4 keret is, amelyet szintén nem ábrázoltunk már. Ezután B1 az F_4, B2 pedig az F_3 keretet fogja továbbítani, és ez a körforgás sosem fog leállni.

Feszítőfás hidak

A fenti nehézség úgy küszöbölhető ki, hogy a hidak kommunikálnak egymással, és lefedik a hálózat aktuális topológiáját egy olyan feszítőfával (spanning tree), amely eléri az összes LAN-t. Tulajdonképpen annyi a teendő, hogy bizonyos kapcsolódási pontokat figyelmen kívül kell hagyni, hogy egy képzeletbeli, hurokmentes topológia jöjjön létre. Vegyük például a 4.40.(a) ábrát, amelyen kilenc LAN látható, tíz híddal összekötve. A hálózat átalakítható egy olyan gráffá, amelyben a LAN-ok képezik a csomópontokat. Egy-egy él köti össze azokat a csomópontokat, amelyek között híd helyezkedik el. A gráf átalakítható egy feszítőfává, ha eldobjuk a 4.40.(b) ábrán pontozott vonalakkal jelölt éleket. Ezt a feszítőfát használva minden LAN-tól csak egyetlen útvonal vezet az összes többihez. Miután a hidak megállapodtak a feszítőfában, minden LAN-ok közötti továbbítás ezt a fát követi. Mivel így minden forrás és cél között csak egyetlen útvonal létezik, nem jöhetnek létre hurkok.

A feszítőfa felépítéséhez a hidaknak először maguk közül egyet ki kell választaniuk, hogy az legyen a fa gyökere. A választást úgy végzik el, hogy mindannyian szét-



4.40. ábra. *(a) Összekötött LAN-ok. (b) Egy feszítőfa, amely lefedi a LAN-okat. A pontozott vonalak nem részei a feszítőfának*

küldik gyári sorozatszámukat, amely a gyártók által garantáltan az egész világon egyedi. Az a híd alkotja a fa gyökerét, amelyiknek a legkisebb a sorozatszáma. A következő lépésben a gyökértől minden egyes hídhoz és LAN-hoz menő legrövidebb útvonalat kijelölő fa meghatározása történik. Ez a fa lesz a feszítőfa. Ha valamelyik LAN vagy híd meghibásodik, újabb fa készül.

Az algoritmus eredménye az, hogy minden LAN számára egyetlen útvonal lett meghatározva a gyökér felé, és így a többi LAN felé is. Igaz, hogy a fa lefedi az összes LAN-t, de nem biztos, hogy az összes híd szerepel benne (a hurkok elkerülése miatt). Miután a feszítőfa elkészült, az algoritmus tovább fut, hogy a hálózat topológiai változásait automatikusan észlelhesse, és a fát bármikor frissíthesse. A feszítőfa kialakítására alkalmazott elosztott algoritmust Perlman fejlesztette ki, és munkájában részletesen tárgyalja (Perlman, 1992).

Hidak arra is használhatók, hogy egymástól nagy távolságban levő LAN-okat kössünk össze. Az ilyen modellben mindegyik helyszínen LAN-ok egy halmaza található, amelyeket hidak kötnek össze. Ezen hidak közül az egyik, egy WAN-nal áll kapcsolatban. Az egymástól távol levő LAN-ok közötti keretforgalom a WAN-on utazik keresztül. Az alapvető feszítőfa algoritmus itt is használható, előnyben részesítve bizonyos optimalizálást annak a fának a kiválasztására, amelyik a WAN forgalom mennyiségét minimalizálja.

4.4.3. Forrás által irányított hidak

A transzparens hidak előnye az, hogy egyszerűen telepíthetőek. Elegendő csatlakoztatni ezeket, és már működnek is. Másfelől azonban nem tudják a sávszélességet optimálisan kihasználni, hiszen a teljes topológiának csak egy részét (a feszítőfát) használják fel. E két (és néhány egyéb) szempont fontossága megosztotta a 802-es bizottságot (Pitt, 1988). A CSMA/CD és a vezérjeles sín hívei a transzparens hidakat választották, míg a vezérjeles gyűrű hívei (az IBM támogatásával) az ún. **forrás általi forgalomirányítás (source routing)** mellett tették le voksukat, amelyet a következőkben fogunk tárgyalni. További részletek (Dixon, 1987).

A végletekig leegyszerűsítve, a forrás általi forgalomirányítás azt feltételezi, hogy a küldő állomás pontosan tudja, hogy a célállomás ugyanazon a LAN-on van-e vagy sem. Amikor olyan állomásnak küld keretet, amely másik LAN-on található, akkor azt a forrásgép a forráscím legmagasabb helyi értékű bitjének 1-esre állításával jelöli meg. Ezen kívül a keret fejrészébe beszúrja a pontos útvonalat, amelyet a keretnek be kell járnia.

Ez az útvonal a következő módon áll össze. Mindegyik LAN rendelkezik egy egyedi, 12 bites címmel, valamint minden híd egyértelműen megcímezhető a hozzá kapcsolódó LAN-ok felől egy 4 bites azonosítószámmal. Emiatt két, egymástól távol eső hídnak lehet ugyanaz a száma, de olyan hidaknak, amelyek ugyanahhoz a LAN-hoz csatlakoznak, mindenféleképpen különböző számmal kell rendelkezniük. Az útvonal ezek után nem más, mint egy híd, LAN, híd, LAN, ... számsorozat. A 4.38. ábrán tehát az útvonalat *A* és *D* között az (L1, B1, L2, B2, L3) sorozat írja le.

Egy forrás általi hídnak tehát csakis azokkal a keretekkel kell foglalkoznia,

amelyek forráscímének legmagasabb helyi értékű bitje 1. Az összes ilyen keret esetében végig kell nézze az előírt útvonalat, és meg kell keresse annak a LAN-nak a sorszámát, amelyikről a keretet vette. Ha e mögött a LAN azonosító mögött a saját hídszámát találja, akkor a keretet továbbküldi arra a LAN-ra, amelyik száma a saját azonosítóját követi. Ha a keresett LAN azonosító mögött nem a saját sorszámát találja, akkor nem továbbítja a keretet.

Az algoritmus három különböző implementációs módszerrel is megvalósítható:

1. Szoftver: a híd válogatás nélküli üzemmódban fut, és az összes keretet bemásolja a memóriájába, hogy megvizsgálja a forráscím legmagasabb helyi értékű bitjét. Ha az 1, akkor tovább végzi a keret elemzését, egyébként pedig nem foglalkozik többet vele.

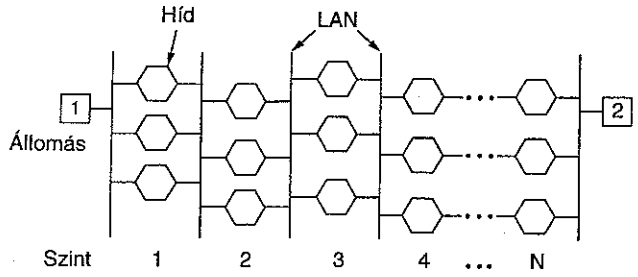
2. Hibrid: a híd LAN illesztője vizsgálja meg a forráscím legmagasabb helyi értékű bitjét, és csak azokat a kereteket fogadja el, amelyeknél ez a bit be van állítva. Ilyen interfész egyszerűen készíthető, és nagymértékben csökkenti a híd által megvizsgálandó keretek számát.

3. Hardver: a híd LAN illesztője nem csak a forráscím legfelső bitjét vizsgálja, hanem le is ellenőrzi, hogy a kijelölt útvonal alapján kell-e valamit tennie a hídnak a kerettel. Kizárólag azok a keretek kerülnek így el magához a hídhoz, amelyeket továbbítania kell. Ez az implementáció igényli a legbonyolultabb hardvert, de a híd CPU-jának egyetlen órajelciklusát sem vesztegeti el, hiszen az összes érdektelen keretet előre kiszűri.

A három implementáció mindössze költségében és teljesítményében különbözik egymástól. Az első nem támaszt semmilyen pluszkövetelményt az illesztő hardverével szemben, viszont nagyon gyors CPU-t igényel a rengeteg keret feldolgozásához. Az utolsó megoldás speciális VLSI áramköröket igényel, de rengeteg munkát vesz le a híd válláról, így lassabb processzor is elegendő, vagy akár több LAN kezelésére is lehetőség nyílik.

A forrás általi forgalomirányítás a háttérben feltételezi, hogy az összekapcsolt hálózatok összes állomása ismeri, vagy képes megtalálni az optimális útvonalat a többi állomás felé. Az algoritmus nagyon fontos része az, hogy hogyan találhatók meg ezek az útvonalak. Az alapötlet az, hogy amennyiben egy állomás nem ismeri a célt, akkor egy keretet adatszórással küld szét, amelyben keresi ezt a célállomást. Ezt a **felkutató keretet (discovery frame)** minden híd továbbítja, így eljut a rendszer összes LAN-jára. Amikor a válasz visszafelé halad, a hidak bejegyzik saját azonosítóikat, így az eredeti küldő fél láthatja a visszatérési útvonalat, és így végül kiválaszthatja a legjobb útvonalat.

Igaz, hogy ez az algoritmus biztosan megtalálja a legjobb útvonalat (az *összest* megtalálja), de sajnos keretrobbanást idézhet elő. Vegyük például a 4.41. ábrán látható hálózatot, amelyben *N* darab LAN van három-három híddal lineárisan összekötve. Minden felkutató keretet, amelyet az 1-es állomás elküld, mind a három híd átmásolja a 2-es LAN-ra. Mindezeket továbbítja a következő három híd a 3-as LAN-ra, ahol



4.41. ábra. *LAN-ok sora, amelyeket hídhármasok kötnek össze*

immár kilenc felkutató keret fog megjelenni. Mire elérjük az N-edik LAN-t, már 3^{N-1} keret fog bolyongani. Ha a keretek egy tucat hídhármason haladnak keresztül, akkor több mint félmillió felkutató keretet kell az utolsó LAN-ba belenyomni, ami óriási torlódáshoz vezetne.

Bizonyos mértékben hasonló folyamat játszódik le a transzparens hidak esetén is, bár ott nem ilyen vészes a helyzet. Amikor olyan keret érkezik, amelynek ismeretlen a címzettje, a hidak elárasztják vele a hálózatot. A különbség az, hogy csak a feszítőfa mentén történik az árasztás, így a létrejövő keretek száma lineárisan, és nem exponenciálisan arányos a LAN-ok számával.

Miután egy állomás kiderített egy adott célállomáshoz vezető útvonalat, tárolja azt egy gyors tárban, így nem kell legközelebb újból lefuttatni a kereső eljárást. Igaz, hogy ez a megközelítés jócskán lecsökkenti a keretrobbanás okozta kényelmetlenségeket, de bizonyos mértékű adminisztrációt ruház a gépekre, valamint egyáltalán nem teljesíti az átlátszóság elvárását, amit alapvető célként neveztek meg, mint azt már korábban említettük.

4.4.4. A 802-es hidak összehasonlítása

Mind a transzparens, mind a forrás általi hidaknak megvannak az előnyeik és hátrányaik. A következő részben ezek közül vesszük végig a fontosabbakat. Egy összefoglaló táblázat található a 4.42. ábrán, valamint további részletek találhatóak (Soha és Perlman, 1988; Zhang, 1988) műveiben. Vegyük azonban figyelembe, hogy a következő pontok közül mindegyik körül heves viták folynak!

A két hídtípus közötti különbségek alapja az összeköttetés nélküli és az összeköttetés alapú hálózatkezelés között fennálló eltérés. A transzparens hidak semmilyen virtuális áramkört nem kezelnek, hanem minden keretet külön, az összes többitől függetlenül irányítanak célba. Ezzel szemben a forrás általi hidak meghatároznak egy útvonalat a felkutató keretek segítségével, és ettől kezdve mindig ezt az útvonalat használják.

A transzparens hidak teljesen láthatatlanok az állomások számára, és teljesen kompatíbilisek az összes meglevő 802-es termékkel. A forrás általi hidak nem átlátszóak és nem is kompatíbilisek. Használatukhoz a hosztoknak a forgalomirányítás sémájára teljes mértékben fel kell készülniük, és tevékenyen részt kell vennük abban. Egy

Szempont	Transzparens híd	Forrásirányított híd
Orientáltság	Összeköttetés nélküli	Összeköttetés alapú
Átlátszóság	Teljesen átlátszó	Nem átlátszó
Konfiguráció	Automatikus	Kézi
Forgalomirányítás	Nem optimális	Optimális
Keresés	Hátrafelé tanulás	Felkutató keretek
Hibák	A hidak kezelik le	A hosztok kezelik le
Komplexitás	A hidakban	Az állomásokban

4.42. ábra. *A transzparens és forrás általi hidak összehasonlítása*

meglevő LAN forrás általi híddal történő kettéválasztásához az állomások szoftverét meg kell változtatni.

Ha transzparens hidakat használunk, nincs szükség hálózatfelügyeletre. A hidak automatikusan konfigurálják magukat a hálózat felépítéséhez. A forrás általi hidaknál a rendszer adminisztrátorának saját kezűleg kell megadnia a LAN-ok és hidak azonosítószámait. Az esetleges hibák, mint a duplázott LAN vagy hídazonosítók, csak nagyon nehezen vehetők észre, mivel ilyen esetben néhány keret hurokba kerülhet, míg az összes többi, amelyek más útvonalat használnak, probléma nélkül ér célba. Ezen felül, ha két különálló komplex hálózatot szeretnénk összekapcsolni transzparens híd használatával, akkor semmi más tennivalónk nincs, mint összekapcsolni azokat. Ezzel ellentétben, ha forrás általi hidat használunk, akkor több LAN azonosítóját is meg kell változtatnunk, hogy a teljes hálózat szemszögéből is egyediek maradjanak.

A forrás általi hidak néhány előnye közül az egyik az, hogy elméletileg lehetséges az optimális forgalomirányítás megvalósítása, míg a transzparens hidaknak ragaszkodniuk kell a feszítőfa éleinek használatához. Ezen felül, a forrás általi hidak ki tudják használni a LAN-ok között fennálló párhuzamos kapcsolatokat, hogy megosszák a terhelést. Az viszont már kérdéses, hogy a jelenleg meglevő hidak elég okosak-e ahhoz, hogy ezeket az elméleti előnyöket meg is valósítsák.

A célállomások felkutatása transzparens hidak esetén a hátrafelé tanulás, míg forrás általi hidak esetén felkutató keretek használatával valósul meg. A hátrafelé tanulás hátránya az, hogy a hidaknak meg kell várniuk, amíg az egyes állomásoktól egy-egy keret áthalad rajtuk, mielőtt megtanulhatnák, hogy merre is találhatók azok. A felkutató keretek használatának hátránya az exponenciális robbanás kialakulása, amely közepes és nagyméretű hálózatokban jelentkezhet, ha párhuzamos hidakat is használnak bennük.

A két séma esetén a hibakezelés módja teljesen eltérő. A transzparens hidak gyorsan és automatikusan tanulnak a bekövetkező híd-, illetve LAN hibákból, valamint topológiai változásokból egész egyszerűen azzal, hogy figyelik egymás vezérlőkereteit. A változásokat a hosztok észre sem veszik.

Forrás általi forgalomirányítás esetén a helyzet teljesen eltérő. Amikor egy híd meghibásodik, a számítógépek, amelyek ezen a hídon keresztül haladó útvonalakat használnak, először csak azt veszik észre, hogy nem érkezik több nyugta kereteikre,

így lejár az időzítésük és újra meg újra próbálkoznak. Végül elkönyvelik, hogy valami nincs rendben, de nem tudhatják, hogy a hiba a célállomásban vagy a használt útvonalban van-e. Kizárólag egy újabb felkutató keret küldésével győződhetnek meg arról, hogy a célállomás elérhető-e. Ha szerencsétlenségre egy központi híd megy tönkre, még akkor is rengeteg állomás időzítése fog lejárni, és rengeteg felkutató keret fog megjelenni a hálózaton a probléma megoldódásáig (amelyek a célállomások elérhetőségét vizsgálják), ha léteznek alternatív útvonalak is.

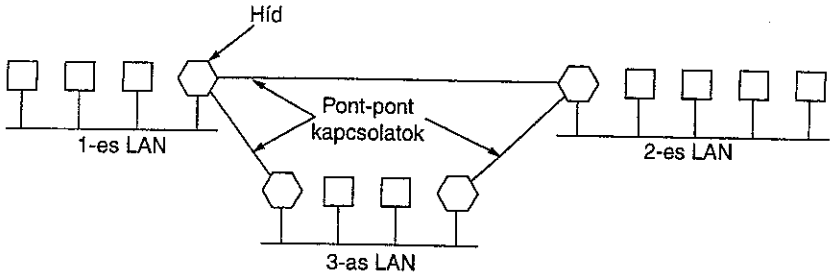
Végül elérkeztünk a komplexitás és költségek jócskán vitatott kérdéséhez. Ha a forrás általi hidakat olyan VLSI áramkörökkel építik fel, amelyek csak azokat a kereteket olvassák be, amelyeket továbbítani kell, akkor adott hardver költségek mellett ezek a hidak kisebb keretfeldolgozási terheléssel és jobb teljesítménnyel fognak üzemelni. Ilyen áramkör nélkül viszont rosszabb teljesítményt fognak nyújtani, mivel keretenként jóval több számítás elvégzésére van szükségük (a keret fejrészében található útvonal végignézése miatt).

Ezen túlmenően, a forrás általi forgalomirányítás növeli a hosztok komplexitását is: tárolniuk kell az útvonalakat, felkutató kereteket kell küldözgetniük, valamint minden keret fejrészébe be kell másolniuk az útvonal-információkat. Ezen feladatok közül mindegyik memóriát és CPU időt igényel. Mivel általában egy vagy két nagyságrenddel több állomás van egy hálózatban mint híd, talán jobb, ha az extra költséget és komplexitást a néhány hídba invesztáljuk, minthogy a rengeteg állomásba.

4.4.5. Távoli hidak

A hidakat sűrűn arra használják, hogy két vagy több, egymástól távoli LAN-t kössenek össze. Lehet például, hogy egy vállalatnak a telephelyei különböző városokban vannak, és mindegyiknek van saját LAN-ja. Ideális esetben az összes LAN-t össze kell kötni, hogy úgy viselkedhessenek, mint egyetlen nagy hálózat.

Ez a cél elérhető, ha mindegyik LAN-ba berakunk egy hidat, és a hidakat kétpontos kapcsolatokkal kötjük össze páronként (pl. a telefontársaságtól bérelt vonalakkal). Egy egyszerű rendszer látható a 4.43. ábrán, amely három LAN-t foglal magába. A megszokott forgalomirányítási algoritmusok ilyen esetben is használhatók. A legegyszerűbb, ha úgy képzeljük el a kétpontos kapcsolatokat, mint olyan LAN-okat, amelyeken egyetlen hoszt sincs. Ekkor egy normál rendszert kapunk, amelyben hat LAN-t köt össze négy híd. Az eddig tanultakban semmi sem állította azt, hogy egy LAN-on kellenek állomások.



4.43. ábra. *A távoli hidakkal egymástól távol eső LAN-ok köthetők össze*

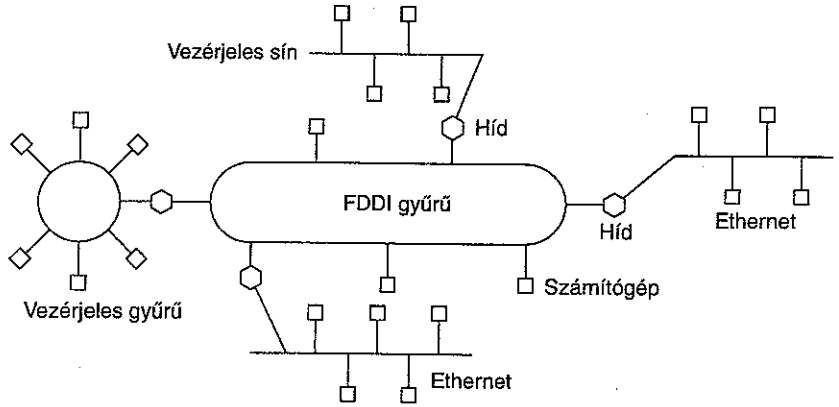
A pont-pont kapcsolatokon a legkülönbözőbb protokollok használhatóak. Az egyik lehetőség az, hogy a szabványos pont-pont adatkapcsolati protokollok közül választunk, és az adat mezőt a komplett MAC keretek képezik. Ez a stratégia akkor működik a legjobban, ha a LAN-ok egyformák, és ekkor a kizárólagos gondot csak az jelenti, hogy a kereteket a megfelelő LAN-ra továbbítsuk. Egy másik lehetőség az, hogy a forráshídnál eltávolítják a MAC fej- és lábrészt, és a pont-pont protokoll adat mezejébe az így kapottakat helyezik. Ezután a célhíd újabb MAC fej- és lábrészt állít elő számára. Ennek a megközelítésnek az a hátránya, hogy a célállomáshoz megérkező ellenőrző összeg nem azonos azzal, amelyet a küldő állomás számított ki, így a hidak memóriájában előforduló hibás bitek által okozott hibák felderítetlenül maradnak.

4.5. Nagy sebességű hálózatok

A 802 LAN és MAN hálózatok, amelyekről eddig tanultunk, mind egy rézvezetékre épülnek (a 802.6 két rézvezetéket igényel). Kis sebesség és távolságok mellett ez elegendő is, de nagy sebességű, nagy távolságokat felölelő LAN-oknak üvegszálra, vagy több párhuzamos rézvezetéket tartalmazó kábelekre kell épülniük. Az üvegszálnak nagy a sávszélessége, vékony, könnyű, és nincs rá hatással a nagy gépek környezetében kialakuló elektromágneses terek zavarása (nagyon fontos ez, amikor a kábelek liftaknákban futnak), a tápfeszültség ingadozása vagy a villámlás. Ráadásul biztonságtechnikailag is kiváló közeg, mivel szinte teljesen lehetetlen észrevétlenül megcsapolni. Mindezeknek köszönhetően a nagy sebességű LAN-ok rendszerint üvegszálat használnak. A következőkben megvizsgálunk néhány olyan helyi hálózatot, amely üvegszálat használ, és egy olyan nagyon nagy sebességű LAN-t, amely a régimódi rézvezetéket használja (igaz, abból jó sokat).

4.5.1. FDDI

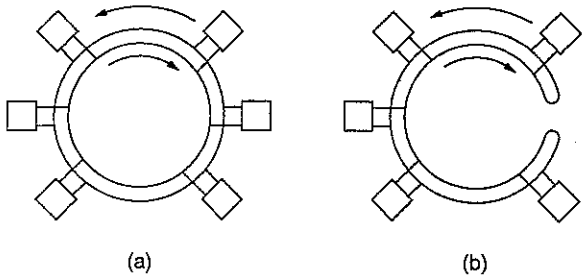
Az **FDDI (Fiber Distributed Data Interface – fényvezető szálas osztott adat interfész)** egy nagy teljesítményű üvegszálas vezérjeles gyűrűhálózat, amely 100 Mb/s sebességgel üzemel, maximálisan 200 km-es kiterjedésű is lehet, és akár 1000 állomás is csatlakozhat rá (Black, 1994; Jain, 1994; Mirchandani és Khanna, 1993; Ross és Hamstra, 1993; Shah és Ramakrishnan, 1994; Wolter, 1990). Ugyanúgy használható, mint bármelyik 802-es LAN, csak nagyobb a sávszélessége, így szívesen használják a 4.44. ábrán látható módon gerinchálózatként is, amely rézvezetékes LAN-okat köt össze. Az FDDI-II, az FDDI utódja, amelyet úgy módosítottak, hogy a közönséges adatok mellett képes legyen hang- vagy ISDN forgalom számára szinkron, vonalkapcsolt PCM adatokat is kezelni. A továbbiakban mindkettőre csak FDDI-ként hivatkozunk. Ez a szakasz a fizikai réteget és a MAC alréteget egyaránt részletezi.



4.44. ábra. *Gerinchálózatként használt FDDI gyűrű, amely LAN-okat és számítógépeket köt össze*

Az FDDI többmódusú üvegszálakat használ, mivel mindössze csak 100 Mb/s sebességgel működik, így nincs szükség a többletköltséget jelentő egymódusú szálakra. Lézer helyett inkább LED-eket használnak, nemcsak azért, mert olcsóbbak, hanem azért is, mert néha a munkaállomások közvetlenül csatlakoznak hozzá. Fennáll annak is a veszélye, hogy a kíváncsi felhasználók kihúzzák a csatlakozót és belenéznek, hogy megnézzék, hogyan röpködnek a bitek 100 Mb/s-os sebességgel. Ha lézert használnának, akkor a kíváncsi felhasználó retinája kilyukadhatna. A LED-ek túl gyengék ahhoz, hogy károsíthassák a szemet, viszont elég erősek ahhoz, hogy pontosan szállítsák az adatokat 100 Mb/s sebességgel. Az FDDI specifikációja $2{,}5 \times 10^{10}$ bitenként mindössze egyetlen hibát engedélyez, és sok implementáció még ennél is sokkal megbízhatóbb.

Az FDDI kábelezése két üvegszálas gyűrűből áll, az egyik az óramutató járásával megegyező, a másik azzal ellentétesen szállítja az adatokat, ahogyan ezt a 4.45.(a) ábra szemlélteti. Ha bármelyik megszakad, a másik tartalékként felhasználható. Ha például ugyanazon a helyen mindkettő megszakadna a kábelcsatornában tűz vagy más



4.45. ábra. *(a) Az FDDI két ellentétes irányú gyűrűből áll. (b) Ha egy ponton mindkét gyűrű megszakad, a két gyűrű összekapcsolható, hogy egy hosszú gyűrűt alkosson*

baleset miatt, akkor a két gyűrű összekapcsolható egy, körülbelül kétszer olyan hosszú gyűrűvé, amint az a 4.45.(b) ábrán látható. Minden állomásban relék vannak, amelyek összekapcsolhatják a gyűrűket, vagy kiiktathatják az állomást, ha az meghibásodna. A 802.5-höz hasonlóan huzalközpontokat is lehet használni.

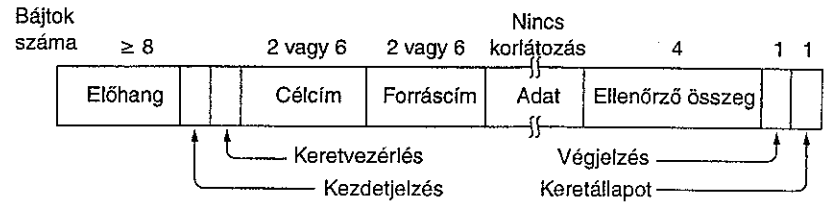
Az FDDI két állomásosztályt definiál, *A*-t és *B*-t. Az *A* osztályú állomások mindkét gyűrűvel össze vannak kapcsolva. Az olcsóbb *B* osztályú állomások csak az egyik gyűrűhöz csatlakoznak. A hibatűrés fontosságától függően üzembe helyezéskor felépíthetjük a hálózatot csupa *A* osztályú, csupa *B* osztályú, vagy mindkét osztályból keverten szereplő állomásokból.

A fizikai réteg nem Manchester-kódolású, mivel a 100 Mb/s kódolás 200 Mbaud-ot követelne meg, amit túl költségesnek ítéltek meg. Ehelyett az **5-ből 4 (4 out of 5) kódolást** használják. Minden 4 MAC szimbólumból (0-kat, 1-eket és egyéb nem adat szimbólum, mint a keretkezdetjelzés) álló csoport 5 bites csoporttá kódolva jelenik meg a közegen. A lehetséges 32 kombinációból 16 az adatok, 3 a határolók, 2 a vezérlés és 3 a hardverjelzések számára van fenntartva, míg 8 kihasználatlan (a protokoll későbbi verzióinak van fenntartva).

E kódolásnak az előnye az, hogy sávszélességet takarít meg, hátránya viszont az, hogy elveszíti a Manchester-kódolás önszinkronizáló tulajdonságát. Ennek kompenzálására az adó egy hosszú előtagot küld a keret elején, amely a vevő órájának szinkronba hozására alkalmas. További követelmény az is, hogy az összes órának legkevesebb 0,005%-on belüli pontosságúnak kell lennie. Ilyen stabilitás mellett legfeljebb 4500 bájt hosszú keret küldhető el annak veszélye nélkül, hogy a vevő órája hibát okozó mértékben elcsúszna az adatfolyam által meghatározott szinkrontól.

Az alap FDDI protokoll modellje a 802.5 protokollon alapul. Adatküldéshez az állomásoknak először a vezérjelet kell megszerezniük. Ezután elküld egy keretet, majd kivonja azt a gyűrűről, amikor az hozzá visszaérkezett. Egy különbség az FDDI és a 802.5 között az, hogy a 802.5-ben az állomások nem állítanak elő új vezérjelet addig, amíg a keretük körbejárva a gyűrűt vissza nem érkezett hozzájuk. Az FDDI esetén, amely potenciálisan 1000 állomásból és 200 km-nyi üvegszálból áll, az időveszteség, amit a keret visszaérkezésére való várakozás okozna, jelentős lenne. Emiatt megengedik, hogy az állomások a kereteik elküldését követően azonnal visszarakhassák a vezérjelet a gyűrűre. Egy nagyobb gyűrűn egyszerre akár több keret is elférhet.

Az FDDI adatkeretei nagyban hasonlítanak a 802.5 kereteire. Az FDDI formátumát a 4.46. ábra szemlélteti. A *Kezdetjelzés* és a *Végjelzés* mezők jelölik ki a keret határait. A *Keretvezérlés* mutatja meg, hogy milyen típusú keretről van szó (adat, vezérlő stb.).



4.46. ábra. *FDDI keret formátum*

A *Keretállapot* mező nyugtázó biteket hordoz, amelyek 802.5-höz hasonlóak. A többi mező analóg a 802.5 protokollnál használtakkal.

A szokványos (aszinkron) keretek mellett az FDDI speciális szinkron kereteket is engedélyez vonalkapcsolt PCM vagy ISDN adatok számára. Egy master állomás ezeket a szinkron kereteket, a PCM rendszerekhez szükséges 8000 minta/s sebesség fenntartásához, 125 μs-onként állítja elő. Mindegyik ilyen keret egy fejrészből, 16 bájt nem vonalkapcsolt, és legfeljebb 96 bájt vonalkapcsolt adatból (azaz keretenként legfeljebb 96 PCM csatornából) áll.

Azért választották a 96-os számot, mert ez vagy négy T1-es csatorna (4 × 24) 1,544 Mb/s-on, vagy három CCITT E1 csatorna (3 × 32) 2,048 Mb/s-on való keretbe foglalását teszi lehetővé, és így az egész világon mindenütt elfogadható. A 96 vonalkapcsolt csatorna számára az elérhető sávszélességből a 125 μs-onként kibocsátott szinkronkeretek 6,144 Mb/s-ot fogyasztanak el. A 125 μs alatt lehetséges maximum 16 szinkron keret legfeljebb 1536 PCM csatornát engedélyez, és 98,3 Mb/s-ot emészt fel.

Ha egy állomás egyszer már lefoglalt magának egy vagy több időrést egy szinkron keretben, akkor azok mindaddig foglaltak maradnak, amíg az állomás explicit módon fel nem engedi azokat. A szinkron keretek által fel nem használt teljes sávszélesség igény szerint osztható ki. Minden keretben található egy bitmaszk, amely jelzi, hogy mely időrések oszthatók ki az igénylők között. A nem szinkron forgalom prioritási osztályokra van osztva, így a fennmaradó sávszélességet először a legnagyobb prioritásúak használhatják fel.

Az FDDI MAC protokollja három időzítőt használ. A **vezérjeltartási időzítő (token holding timer)** azt határozza meg, hogy egy állomás mennyi ideig adhat, miután megszerezte a vezérjelet. Ez az időzítő akadályozza meg, hogy egy állomás végleg lefoglalhassa magának a gyűrűt. A **vezérjel-körbefutási időzítő (token rotation timer)** minden alkalommal újraindul, amikor a vezérjel feltűnik. Az időzítő lejárta azt jelenti, hogy a vezérjel túl hosszú ideje nem érkezett meg az állomásra. Lehetséges, hogy elveszett, így elindul a vezérjel-előállító eljárás. Végül, a **szabályos átvitel időzítő (valid transmission timer)** a gyűrű átmeneti hibáiból való felépülésre szolgál.

Az FDDI prioritási algoritmusa hasonló a 802.4-ével. Azt határozza meg, hogy a vezérjel egy adott átfutása alatt melyik prioritásosztályok adhatnak. Ha a vezérjel a vártnál korábban érkezik, akkor az összes prioritási osztály adhat, ha később érkezik, akkor csak a legmagasabb prioritásúak küldhetnek adatokat.

4.5.2. Gyors Ethernet

A LAN-ok következő generációjának szánták az FDDI-t, azonban sosem tudott kitörni a gerinchálózati piacról (ahol viszont továbbra is jól szerepel). Az állomások felügyelete túl komplikált volt, ami bonyolult áramkörökhöz vezetett, és magas árakat eredményezett. Az FDDI áramkörök túlzottan magas ára miatt a számítógépgyártók nem akarták hálózati szabványnak választani, így sosem került tömeggyártásra, és az FDDI sosem tört be a tömegpiacra. A lecke, amit itt ismerni kellett volna, röviden úgy nevezhető: KISS (Keep It Simple, Stupid – Tartsd meg egyszerűnek, butának).

Mindenesetre az FDDI sikertelensége óriási űrt hagyott a 10 Mb/s feletti sebességű

LAN-ok számára. Sok hálózat nagyobb sávszélességet igényel, ezért több LAN-ból állnak, amelyeket ismétlők (repeater), hidak (bridge), routerek és átjárók (gateway) egész hada kapcsol össze. A rendszergazdák sokszor már azt érzik, hogy az egész rendszert csupán a rágógumi és számítástechnika őrangyalai tartják egyben.

Ezen körülmények miatt az IEEE 1992-ben újból felkérte a 802.3 bizottságot, hogy készítsenek egy gyorsabb LAN szabványt. Az egyik javaslat az volt, hogy tartsák meg a 802.3 szabványt olyannak amilyen, csak tegyék gyorsabbá. A másik indítvány szerint viszont az egészet elölről kellene kezdeni, hogy olyan újabb szolgáltatásokkal ruházhassák fel, mint a valósi dejű forgalom és a digitális hangátvitel, és csak a régi nevet tartsák meg (üzleti okokból). Parázs viták után, a bizottság úgy döntött, hogy megtartják a 802.3 szabványt úgy, ahogy van, csak gyorsabbá teszik. A vesztes indítvány mögött álló emberek úgy tettek, ahogy a számítógépgyártók is tettek volna ilyen körülmények között, így tehát megalapították saját bizottságukat, és elkészítették a saját LAN szabványukat (tulajdonképpen a 802.12-t).

A három elsődleges ok, amiért a 802.3 bizottság a felgyorsított 802.3 LAN mellett döntött a következő volt:

1. Szükséges volt a hátrafelé kompatíbilitás a több ezer már meglevő LAN-nal.

2. Féltek, hogy az új protokoll előre nem látható problémákat hozhat magával.

3. Szerettek volna a munkával azelőtt végezni, hogy a technológia megváltozna.

A munkával hamar végeztek (a szabványosítási bizottságok normáihoz képest), és az eredményt, a **802.3u** szabványt 1995 júniusában elfogadta az IEEE. Technikailag a 802.3u nem új szabvány, hanem a 802.3 szabvány kiegészítése (a hátrafelé kompatíbilitás kihangsúlyozása érdekében). Mivel mindenki **gyors Ethernet** néven emlegeti 802.3u helyett, mi is így fogunk hivatkozni rá.

A gyors Ethernet alapötlete egyszerű: tartsunk meg minden régi keret formátumot, interfészt és eljárási szabályt, de csökkentsük le a bit-időt 100 ns-ról 10 ns-ra. Műszaki szempontból lehetséges lett volna a 10Base-5 vagy a 10Base-2 lemásolása, és még az ütközéseket is időben észlelhetnénk, ha tizedére csökkentenénk le a megengedett maximális kábelhosszt. A 10Base-T kábelezés előnyei azonban annyira ellenállhatatlanok voltak, hogy a gyors Ethernetet teljes egészében erre a megoldásra alapozták. Emiatt minden gyors Ethernet rendszer elosztókat (hub) használ; nem engedélyezi a vámpír csatlakozós többpontos kábeleket és a BNC csatlakozókat.

Néhány döntést azonban így meg kellett hozni. A legfontosabb ezek közül az, hogy milyen kábelfajtákat támogassanak. A verseny egyik résztvevője a 3-as kategóriájú csavart érpár. Az érv, ami mellette szólt az volt, hogy a nyugati világban minden irodát legalább négy 3-as kategóriájú (vagy jobb) csavart érpár köti össze a 100 méteren belül levő telefonos kábelrendezővel. Néha két ilyen kábel is létezik. Ilyen módon a 3-as kategóriájú csavart érpár lehetővé tenné, hogy az asztali számítógépeket gyors Ethernet hálózatba csatlakoztassák anélkül, hogy az egész épületet újra kéne kábelezni, ami óriási előnyt jelentene sok szervezet számára.

A 3-as kategóriájú csavart érpáros kábel fő hátránya az, hogy képtelen a 200 mega-

Név	Kábel	Max. szegmens	Előnyei
100Base-T4	Csavart érpár	100 m	3-as kategóriájú UTP-t használ
100Base-TX	Csavart érpár	100 m	Duplex 100 Mb/s
100Base-FX	Optikai szál	2000 m	Duplex 100 Mb/s, nagy távolság

4.47. ábra. *Gyors Ethernet kábelezési fajták*

baud-os jelek (100 Mb/s Manchester-kódolással) 100 méterre történő elszállítására, a maximális távolságra, amely a 10Base-T szabvány szerint a gépek és az elosztó (hub) között lehet (lásd. 4.47. ábra). Ezzel szemben az 5-ös kategóriájú csavart érpár a 100 méteres akadályt könnyedén veszi, az üvegszál pedig még sokkal távolabb is eljut. A kompromisszum az volt, hogy mind a három lehetőséget megengedik, ahogy ezt a 4.47. ábra is szemlélteti, de a 3-as kategóriás megoldást fel kellett javítani úgy, hogy megnövekedjen a szállítási kapacitása.

A 3-as kategóriájú UTP séma, amelyet **100Base-T4**-nek neveznek, 25 MHz-es jeleket használ, amely csak 25%-kal több a 802.3 szabvány 20 MHz-énél (emlékezzünk, hogy a Manchester-kódolás, a 4.20. ábrának megfelelően, két órajel-periódust igényel). A szükséges sávszélesség eléréséhez a 100Base-T4 négy csavart érpárt igényel. Miután már évtizedek óta négy csavart érpár fut a telefonkábelekben, a legtöbb irodának ez nem jelent különösebb problémát. Természetesen ez az irodai telefon feladását vonja maga után, de ez biztosan csekély ár, amit a gyorsabb elektronikus levelezésért kell fizetni.

A négy érpár közül egy állandóan az elosztó felé, egy állandóan az elosztó felől, a maradék kettő pedig átkapcsolható módon, az aktuális átvitel irányába szállítja az adatokat. A szükséges sávszélesség elérése érdekében nem használnak Manchester-kódolást, de ez a modern órák és a kis távolságok mellett nem is igazán szükséges. Emellett három jelszintű jeleket küldenek, így egyetlen órajel alatt a vezetéken megjelenhet egy 0, egy 1 vagy egy 2. Ha három csavart érpár megy a forgalom irányában, és mindegyiken három jelszintű jelzéseket használnak, akkor a 27 lehetséges szimbólum közül bármelyik átküldésre kerülhet, így lehetőség nyílik arra, hogy kis redundanciával 4 bitet is továbbítsanak. Ha a másodpercenként 25 millió órajelciklusonként 4 bitet küldenek el, akkor előáll a szükséges 100 Mb/s-os átvitel. Ráadásul állandóan készen áll egy 33,3 Mb/s-os szembejövő csatorna, amely a megmaradt csavart érpárt használja. Ez a megoldás, amelyet **8B6T (8 bits map to 6 trits – 8 bit leképezve 6 tercre)** néven ismernek, nem lehet az elegancia-díj nyertese, de legalább működik a már meglevő kábelezéssel.

Az 5-ös kategóriájú kábelezési eset, a **100Base-TX** sokkal egyszerűbb, mivel a kábel a 125 MHz-nél nagyobb frekvenciájú órajeleket is képes kezelni. Állomásonként elég mindössze két csavart érpár, egy az elosztó felé, egy másik pedig az elosztó felől. A normál bináris kódok helyett 125 MHz-en az ún. **4B5B kódolási** sémát használják. Az órajelek minden ötös csoportját 4 bit átvitelére használják, így egy kis redundanciát vihettek a rendszerbe, ami megkönnyíti az órák szinkronizálását, lehetőséget ad a kerethatár jelzések egyedi mintákkal történő megkülönböztetésére, valamint a fizikai réteg szintjén kompatíbilissé teszi a szabványt az FDDI-vel. Mindezeknek köszönhetően a 10Base-TX egy duplex rendszer: az állomások egyidejűleg adhatnak és vehetnek 100 Mb/s-mal. Ráadásul akár két telefon is lehet az irodában a tényleges kommunikáció biztosítására, ha a számítógép éppen a Weben való szörfözés miatt lenne teljesen foglalt.

Az utolsó lehetőség, a **100Base-FX**, két szabványos, többmódusú üvegszálat használ, egyet-egyet mindkét irányban, így ez is duplex üzemet biztosít mindkét irányban 100 Mb/s-mal. Mindezek mellett pedig a távolság az állomás és az elosztó között akár 2 km is lehet.

A 100Base-T4 és 100Base-TX (közös néven **100Base-T**) hálózatokban kétfajta elosztó (hub) lehetséges. Egy megosztott elosztóba (shared hub) befutó (vagy legalábbis az egy vonali kártyára befutó) vonalak logikailag össze vannak kapcsolva, így egy ütköztetési tartományt képeznek. Az összes alapszabály, beleértve a bináris visszalépési algoritmust is, ugyanúgy működik, mint a régi 802.3-nál. Lényegében tehát egyszerre mindig csak egy állomás adhat.

Egy kapcsolt elosztó (switched hub) esetében az összes vonalat pufferelik a vonali kártyán. Igaz ugyan, hogy ez a szolgáltatás drágábbá teszi az elosztót és a vonali kártyákat, de azt is eredményezi, hogy az összes állomás egyidejűleg adhat (és vehet), így a teljes rendszer sávszélessége általában egy, de esetenként akár több nagyságrenddel is megnő. A pufferelt kereteket egy nagy sebességű hátlapon keresztül továbbítják a forráskártyától a célkártyáig. A hátlap nincs szabványosítva, de erre nincs is szükség, hiszen teljes egészében mélyen a kapcsoló belsejében rejtőzik. Ha az eddigi tapasztalatok nem csalnak, akkor a kapcsológyártók erőteljes versengésbe fognak kezdeni, hogy minél gyorsabb hátlapokat készítsenek, és így minél jobban megnöveljék a rendszer átbocsátó képességét. Mivel a 100Base-FX kábelek túl hosszúak ahhoz, hogy az Ethernet normál ütköztetési algoritmusa működhessen rajtuk, kizárólag pufferelt, kapcsolt elosztókhoz szabad ezeket csatlakoztatni, hogy így önmagukban alkossanak egy-egy ütköztetési tartományt.

Végezetül érdemes megjegyezni, hogy a kapcsolók virtuálisan képesek 10 és 100 Mb/s-os állomások kevert kezelésére is, így megkönnyítik a hálózatfejlesztést. Amint egyre több és több 100 Mb/s-os állomásra van szükség, nincs más teendő, mint megvenni a megfelelő számú új vonali kártyát, és bedugni azokat a kapcsolóba.

A gyors Ethernetről bővebb információk (Johnson, 1996) dolgozatában található. A nagy sebességű hálózatok, történetesen az FDDI, a gyors Ethernet, az ATM és a VG-AnyLAN összehasonlítása (Cronin és mások 1994) munkáiban található.

4.5.3. HIPPI – Highy-Performance Parallel Interface (Nagy teljesítményű párhuzamos interfész)

A hidegháború alatt az Egyesült Államok kormányának nukleáris fegyvereket kifejlesztő központja, a Los Alamos-i National Laboratory rutinszerűen megvásárolt egyet-egyet minden megvásárlásra felkínált szuperszámítógépből. Los Alamos gyűjtötte még a különleges perifériákat is, mint amilyen a nagy kapacitású háttértár és a speciális, tudományos vizualizációra alkalmas grafikus munkaállomásokat. Akkoriban még minden gyártó különböző csatlakozási felületekkel rendelkezett szuperszámítógépeik

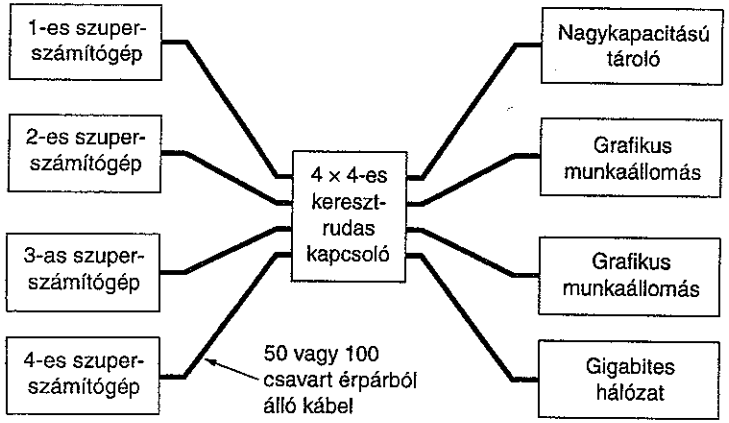
és a perifériák között, úgyhogy lehetetlen volt egymás perifériáinak használata, vagy a gépek összekapcsolása.

1987-ben Los Alamos kutatói elhatározták, hogy kifejlesztenek egy szabványos illesztő felületet a szuperszámítógépek számára, amelyet szabványként elfogadtatnak, majd ráveszik a gyártókat, hogy használják ezt a felületet. (Los Alamos számítástechnikai költségvetése akkora volt, hogy a gyártók adtak a szavukra.) Az interfész célja az volt, hogy mindenki gyorsan és hatékonyan implementálhassa azt. A KISS-t tartották vezérszabályként a szemük előtt: Keep It Simple, Stupid (Tartsd meg egyszerűnek, butának). Ez azt jelentette, hogy ne legyenek választási lehetőségek, ne igényeljen új áramköröket, és a teljesítménye akkora legyen, mint egy tűzoltótömlőnek.

Az eredeti specifikáció 800 Mb/s-ot írt elő, mivel a kilőtt bombákról készült felvételek 1024 × 1024 pontnyi képméretet, 24 bit/pont színmélységet és 30 keret/s képfrissítési sebességet igényeltek, amely összesen 750 Mb/s adatsebességet ad ki. Később egy másik lehetőség is belopódzott: az 1600 Mb/s-os második adatsebesség. Amikor ezt az indítványt, amelyet **HIPPI (HIgh-Performance Parallel Interface – nagy teljesítményű párhuzamos interfész)** névre kereszteltek, szabványosításra ajánlották fel az ANSI-nak, az indítványozókra úgy tekintettek, mint akik más bolygóról jöttek, hiszen az 1980-as években a LAN-ok alatt a 10 Mb/s-os Etherneteket értették.

A HIPPI-t eredetileg adatcsatornának, és nem LAN-nak szánták. Az adatcsatornák kétpontos üzemben működnek egyetlen mester (egy számítógép) és egyetlen szolga (egy periféria) között előre kijelölt vezetékeken, mindenféle átkapcsolás nélkül. Semmilyen verseny sem jöhet létre, és a környezet teljesen kiszámítható. Később szükségessé vált, hogy egy perifériát több szuperszámítógép között át lehessen kapcsolni, és a HIPPI szerkezetét kiegészítették, a 4.48. ábrán bemutatott módon, egy keresztrudas (crossbar) kapcsolóval.

Ahhoz, hogy a már meglevő áramköri elemekkel ilyen nagy teljesítményt érhessenek el, az alap interfész 50 bites volt, amelyből 32 adatbit és 18 vezérlőbit volt, így a HIPPI kábel 50 csavart érpárt tartalmaz. 40 ns-onként került átvitelre az interfészen egy-egy



4.48. ábra. *Keresztrudas kapcsolót használó HIPPI rendszer*

szó. Az 1600 Mb/s eléréséhez két kábelt használnak fel, és minden ciklusban két szó kerül továbbításra. Az összes átvitel szimplex. A kétirányú kommunikációhoz kettő (vagy négy) kábelre van szükség. Ilyen sebességek mellett a maximális kábelhossz 25 méter.

Miután túljutott az első ámulaton, az ANSI X3T9.3 bizottsága létrehozott egy HIPPI szabványt a Los Alamos-i javaslat alapján. A szabvány a fizikai és adatkapcsolati rétegeket fedi le. Minden, ami ezek fölött történik, a felhasználóra van bízva. Az alapvető protokoll szerint, ha egy hoszt kommunikálni szeretne, akkor először megkéri a keresztrudas kapcsolót, hogy hozzon létre egy kapcsolatot. Ezután (általában) elküldött egyetlen üzenetet és bontotta a kapcsolatot.

Az üzenetek struktúrája egy vezérlőszóból, egy maximum 1016 bájtos fejrészből és egy legfeljebb 2^{32}–2 bájt hosszúságú adatrészből áll össze. Forgalomszabályozási okokból az üzeneteket 256 szavas keretekre osztják. Amikor a vevő készen áll egy keret fogadására, jelzést küld az adónak, amely ezután elküld egy keretet. A vevők egyszerre akár több keretet is kérhetnek. A hibajavítás egy szavankénti vízszintes paritásbitből, és a keret végén egy függőleges paritásszóból áll. A tradicionális ellenőrző összegeket szükségtelennek és túl lassúnak tartották.

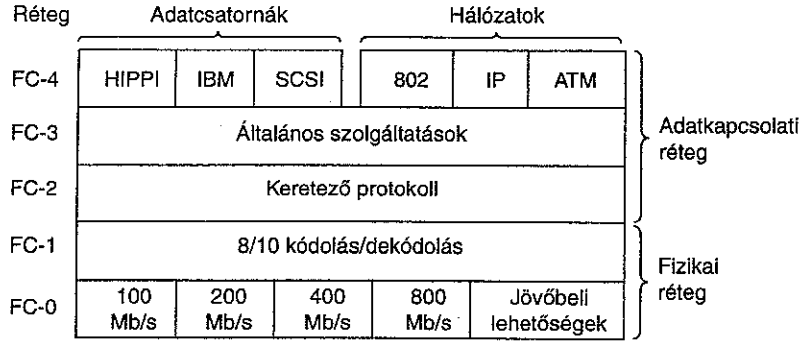
A HIPPI-t több tucat gyártó gyorsan implementálta, és évekig a szuperszámítógépek csatlakozási szabványa lett. További információk (Hughes és Franta, 1994; Tolmie, 1992; valamint Tolmie és Renwick, 1993) műveiben találhatók.

4.5.4. Fibre Channel

Mivel a HIPPI megtervezésének idején a fényvezető szálak nagyon drágák voltak, és megbízhatónak sem tartották, a valaha épített leggyorsabb LAN-t rossz minőségű telefonvezetékekből rakták össze. Az idő haladtával a fényvezető szálak olcsóbbak és megbízhatóbbak lettek, így természetes volt, hogy megpróbálták a HIPPI-t átalakítani úgy, hogy az 50 vagy 100 csavart érpár helyett egyetlen fényvezető szálat használjanak fel. Sajnálatos módon, az a Los Alamos által bevezetett fegyelemi előírás, miszerint az újítási indítványokat el kell fojtani, bármennyire is felütötte a fejét, valahogy mindig eltűnt az út mentén. A HIPPI utóda, a **fényvezető szálas csatorna (fibre channel)** sokkal komplikáltabb, és implementálása is drágább. Majd kiderül, hogy képes lesz-e elérni akkora kereskedelmi sikert, mint amilyet a HIPPI élvezett.

A fényvezető szálas csatorna egyaránt kezel adatcsatornákat és hálózati összeköttetéseket. Tulajdonképpen olyan adatcsatornák kiváltására használható, mint a HIPPI, SCSI és az IBM nagyszámítógépek által használt multiplexer csatornák. Képes ezek mellett hálózati csomagok szállítására is, beleértve az IEEE 802-t, az IP-t és az ATM-et. A HIPPI-hez hasonlóan a fényvezető szálas csatorna alapvető felépítése egy keresztrudas kapcsolóból áll, amely összekapcsolja a bemeneteket a kimenetekkel. Az összeköttetések egyetlen csomag időtartamára, vagy akár sokkal hosszabb időtartamra is kiépíthetők.

A fényvezető szálas csatorna három szolgálati osztályt támogat. Az első osztály az egyszerű vonalkapcsolt szolgálat, amely garantálja a kézbesítést. Az adatcsatorna üzemmódok ezt a szolgálati osztályt használják. A második osztály a csomagkapcsolt, garantált kézbesítésű szolgálat. A harmadik osztály pedig a csomagkapcsolt, kézbesítési garancia nélküli szolgálat.



4.49. ábra. *A fényvezető szálas csatorna protokollrétegei*

A fényvezető szálas csatorna gondosan kidolgozott protokollstruktúrával rendelkezik, amelyet a 4.49. ábra szemléltet. Itt öt réteget láthatunk, amely a fizikai és adatkapcsolati rétegeket fedik le. A legalsó réteg a fizikai közeggel foglalkozik. Jelenleg 100, 200, 400 és 800 Mb/s-os adatsebességeket támogat. A második réteg a vonali kódolást kezeli. A használt rendszer hasonlít az FDDI kódolásához, csak nem 5 biten kódol 16 érvényes szimbólumot, hanem 10 biten 256-ot, így kismértékű redundanciát biztosít. Ez a két réteg együtt, funkcióját tekintve megfelel az OSI fizikai rétegének.

A középső réteg definiálja a keretek felépítését és a fejrészek formátumát. Az adatokat keretekben továbbítják, amelyek hasznos tartalma legfeljebb 2048 bájt lehet. A következő réteg lehetővé teszi a felső réteg számára nyújtott általános szolgáltatások jövőbeli igény szerinti megvalósítását. Végül, a legfelső szint nyújt felületet a különböző támogatott számítógépek és perifériák illesztéséhez.

Mellesleg érdekes, hogy a fényvezető szálas csatornát az Egyesült Államokban tervezték meg, de nevének betűzését mégis a szabvány szerkesztője határozta meg, aki pedig angol volt. A fényvezető szálas csatornával kapcsolatos bővebb információk (Tolmie, 1992) dolgozatában érhetők el. A rendszer HIPPI-vel és ATM-mel való összehasonlításáról (Tolmie, 1995) munkájában olvashatunk.

4.6. Műholdas hálózatok

Igaz ugyan, hogy a legtöbb többszörös elérést biztosító csatornát a LAN-okban találjuk, mégis létezik egy WAN típus, amely szintén ilyen csatornát használ: a távközlési műholdakon alapuló WAN-ok. A következőkben áttanulmányozunk néhány problémát, amelyek műhold alapú nagy kiterjedésű hálózatokban fordulnak elő. Megvizsgálunk néhány protokollt is, amelyeket a problémák kezelésére találtak ki.

A távközlési műholdak megközelítőleg egy tucat transzponderrel rendelkeznek. Minden egyes transzponderhez egy sugárnyaláb tartozik, amely az alatta fekvő földdarabkát fedi le, amely egy széles nyaláb esetén akár 10 000 km, míg egy irányított

nyaláb esetén csupán 250 km átmérőjű is lehet. A sugárnyaláb hatósugarába eső állomások a felirányú (uplink) frekvencián küldhetnek kereteket a műholdnak, amely azokat a leirányú (downlink) frekvencián szórja újra szét. Különböző leirányú és felirányú frekvenciákat használnak, hogy megakadályozzák a transzponder oszcillációba kerülését. Azokat a műholdakat, amelyek egyszerűen mindent megismételnek, amit hallanak (a legtöbb ilyen), **visszahajlított cső (bent pipe)** műholdaknak nevezik.

Minden antenna megcélozhat egy területet, sugározhat néhány keretet, majd megcélozhat egy újabb területet. A célzás elektronikusan történik, de így is igénybe vesz néhány mikroszekundumot. Az időtartamot, amíg a sugárnyaláb egy adott területre irányul, **egyhelyben-tartózkodási időnek (dwell time)** nevezik. A maximális kihasználtság elérése érdekében ez nem lehet túl rövid, hiszen ekkor túl sok időt pocsékolnánk el a nyaláb mozgatásával.

A LAN-okhoz hasonlóan, a tervezés egyik kulcskérdése az, hogyan osszuk ki a transzponder csatornáit. A LAN-októl eltérően azonban, itt nem lehet vivőjel-érzékelést megvalósítani a 270 ms-os jelterjedési idő miatt. Amikor egy állomás belehallgatna a leirányú csatornába, csak azt tudhatná meg, hogy mi történt 270 ms-mal korábban. A felirányú csatorna érzékelése pedig teljesen lehetetlen. Emiatt a CSMA/CD protokollok (amelyek feltételezik, hogy az adó állomások az ütközéseket az első néhány bit-idő elteltével már érzékelik, és visszalépnek, ha ilyesmi történik) műholdakkal együtt nem használhatóak. Így másik protokollra van szükség.

A többszörös hozzáférésű felirányú csatornán öt protokoll osztályt használnak: lekérdezés (polling), ALOHA, FDM, TDM és CDMA. Igaz ugyan, hogy ezekről már mind tanultunk korábban, de a műholdas felhasználás néha újabb fordulatokkal egészíti ki ezeket. A fő probléma a felirányú csatornával van, hiszen a leirányú csatornán csak egyetlen küldő fél (a műhold) van, így azon nincsenek lefoglalási gondok.

4.6.1. Lekérdezés (Polling)

Egyetlen csatorna, versengő felhasználók közötti kiosztásának egy hagyományos módja az, hogy valaki lekérdezi azokat. A 270 ms-os időköz, amit a kérdés/válasz sorozatok igényelnének, túlzottan költségessé tennék, ha a műhold egyenként lekérdezné az állomásokat, hogy van-e elküldendő keretük.

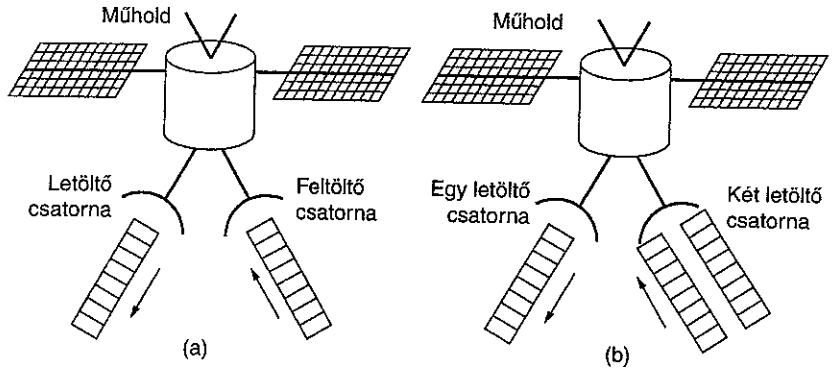
Mindazonáltal, ha a földi állomásokat összekötjük egy (tipikusan kis sávszélességű) csomagkapcsolt hálózattal, akkor az ötlet egy kis módosítással ugyan, de megvalósítható. Az alapötlet az, hogy rendezzük az állomásokat egy logikai gyűrűbe úgy, hogy mindegyikük ismerje a rákövetkezőt. E mentén a földi kör mentén keringjen egy vezérjel. A műhold sosem látja a vezérjelet. Egy állomás akkor adhat a felirányú csatornán, amikor nála van a vezérjel. Ha az állomások száma kicsi és állandó, a vezérjel átviteli ideje kicsi, és a felirányú csatorna löketei sokkal hosszabbak, a vezérjel körbejárási időnél az ilyen séma elég hatékony.

4.6.2. ALOHA

Az egyszerű ALOHA könnyen implementálható: minden állomás akkor ad, amikor csak akar. A baj csak az, hogy a csatorna kihasználtsága 18% körül várható. Általánosan igaz, hogy az ilyen alacsony kihasználási tényező megengedhetetlen több milliárd forint értékű műholdak esetén.

A réselt ALOHA használata megduplázza a hatékonyságot, de felveti azt a problémát, hogyan szinkronizáljuk össze az állomásokat, hogy azok tudják, mikor van az időrések kezdete. Szerencsére a műhold önmagában hordozza a megoldást, hiszen alapvetően adatszórási közeget képez. Egy földi állomás, a **referenciaállomás (reference station)**, periodikusan bocsát ki magából egy speciális jelzést, amelyet a visszaszórást követően a földi állomások az óráik indítójeleként kezelhetnek. Ha minden időrés hossza ΔT, akkor minden állomás tudja, hogy a k-adik időrés az óraindító jel után $k\Delta T$ idővel kezdődik. Mivel az órák kismértékben különböző sebességgel futnak, periodikus újraszinkronizálásra van szükség, hogy mindenki fázisban maradhasson. Pluszprobléma, hogy a műhold és a különböző földi állomások között a terjedési idő eltérő, de ez a jelenség korrigálható. A felirányú csatorna $1/e$-nél jobb kihasználása érdekében a 4.50.(a) ábrán látható, egy felirányú csatornás rendszer helyett használhatunk kettős felirányú csatornát is, ahogyan azt a 4.50.(b) ábra szemlélteti. Az elküldendő kerettel rendelkező állomások véletlenszerűen kiválasztják az egyik felirányú csatornát, és a következő időrésben azon adnak. Ilyen módon mind a két felirányú csatorna egymástól függetlenül működő réselt ALOHA csatornává válik.

Ha csak az egyik felirányú csatorna tartalmaz sikeresen továbbított keretet, akkor azt később, a megfelelő leirányú időrésben sugározzák vissza. Ha mindkét csatornán sikeres átvitel valósul meg, akkor a műhold pufferelheti az egyiket, hogy aztán később, egy üres időrésben azt is továbbítsa. A valószínűségeket kiszámítva láthatjuk, hogy végtelen pufferméretet feltételezve, elérhető a leirányú csatorna 0,736-os kihasználása mindössze azon az áron, hogy a szükséges sávszélességet másfélszeresére kell növelni.



4.50. ábra. *(a) Egy szokványos ALOHA rendszer. (b) Második felirányú csatorna hozzáadása*

4.6.3. FDM

A frekvenciaosztásos nyalábolás (FDM) a csatornakiosztás legrégebbi, és talán a legelterjedtebb módszere. Egy tipikusan 36 Mb/s-os transzponder statikusan körülbelül 500 darab 64 000 b/s-os PCM csatorna között osztható meg, amelyek közül mindegyik saját frekvencián üzemel, így nem befolyásolhatja a többit.

Annak ellenére, hogy az FDM egyszerű, van néhány hátránya is. Először is védősávokra van szükség a csatornák között, hogy az állomásokat szétválasszuk. Ilyen követelmény tényleg létezik, mivel lehetetlen olyan adókat készíteni, amelyek teljes energiájukat képesek kizárólag a fősávba korlátozni anélkül, hogy a szomszédos tartományokban nem sugároznak semmit sem. A védősávokra elpazarolt sávszélesség jelentős hányadát is kiteheti a teljes sávszélességnek.

Másodszor, az állomások teljesítményvezérlésének pontosnak kell lennie. Ha egy állomás túl nagy teljesítménnyel sugároz a fősávban, automatikusan túl nagy teljesítménnyel fog sugározni a környező sávokban is, ami így átszivároghat a szomszédos csatornákba és zavarhatja azokat. Végül, az FDM teljesen analóg technika, így szoftveresen nagyon nehezen implementálható.

Ha az állomások száma kicsi és állandó, akkor a frekvenciacsatornákat statikusan előre is kioszthatják. Ezzel szemben, ha az állomások száma vagy a rajtuk fellépő terhelés erőteljesen változó, akkor a frekvenciacsatornákat valamilyen dinamikus módszerrel kell kiosztani. Az egyik ilyen mechanizmus a **SPADE,** amelyet néhány korai Intelsat műholdas rendszerben használtak. Minden SPADE transzpondert 792 szimplex (64 kb/s-os) PCM hangcsatornára osztottak fel, amelyek mellett egy 128 kb/s-os általános jelzési csatorna is helyet kapott. A PCM csatornákat párokban használták, hogy így duplex szolgáltatást biztosítsanak. A transzponder teljes sávszélességéből egy 50 Mb/s-os részt a felirányú csatorna használ fel, a leirányú csatorna számára pedig másik 50 Mb/s áll rendelkezésre.

Az általános jelzési csatornát 50 ms-os egységekre osztották. Egy-egy egység 50 darab, egyenként 1 ms-os rést (128 bit) tartalmazott. Minden időrés a legfeljebb 50 állomás közül az egyik „birtokában" volt. Amikor egy földi állomás adatokat szeretne küldeni, véletlenszerűen kiválaszt egy használaton kívüli csatornát, és a számát beírja a legközelebbi 128 bites időrésbe. Ha a kiválasztott csatorna szabadon maradt addig, amíg a lefoglalási kérelem megjelent a leirányú csatornán, akkor a csatorna lefoglalásra került, és a többi állomás többet nem próbálhatta meg megszerezni. Ha két vagy több állomás is ugyanazt a csatornát próbálta meg lefoglalni, akkor ütközés jött létre, így később újra kellett próbálkozniuk. Amikor egy állomás befejezte a csatorna használatát, a közös csatornán elküldött egy felszabadító üzenetet a saját időrésében.

4.6.4. TDM

Az FDM-hez hasonlóan, a TDM is jól ismert és széles körben használt. Időszinkronizálást igényel az időrésekhez, de ez egy referenciaállomással biztosítható ugyanúgy, mint ahogy korábban, a réselt ALOHA-nál már részleteztük. Az FDM-hez hasonlóan, ha kis- és állandó számú állomás van, az időrések kiosztása fixen előre beállítható, de

az időréseket dinamikusan kell kiosztani, ha az állomások száma változó, vagy a fix számú állomás időben változó terhelést generál.

Az időréskiosztás központilag vagy decentralizált módon is megvalósítható. Példaként a centralizált időréskiosztásra nézzünk meg a kísérleti **ACTS-t (Advanced Communication Technology Satellite – fejlett kommunikáció-technológiai műhold)**, amelyet néhány tucat állomáshoz terveztek (Palmer és White, 1990). Az ACTM-et 1992-ben lőtték fel, és négy független, 110 Mb/s-os csatornával rendelkezett, kettő felirányú és kettő leirányú csatornával. Mindegyik csatorna 1 ms-os keretek sorozatából áll, és mindegyik keret 1728 időrést tartalmaz. Mindegyik időrés 64 bit adatot szállít, így lehetővé teszik, hogy mindegyik egy 64 kb/s-os hangcsatornát szolgáljon ki.

A sugarak átkapcsolhatók a különböző földrajzi területek között, de mivel a sugár mozgatása többrésnyi időbe is beletelik, azok a csatornák, amelyek forrás- és célállomása ugyanazon a földrajzi területen fekszik általában sorszámfolytonosan kapnak időréseket, hogy ezzel is megnöveljék az egyhelyben-tartózkodási időt (dwell time), és minimalizálják a sugár mozgatása által okozott időkiesést. Emiatt az időrések menedzsmentje az állomások földrajzi helyének pontos ismeretét igényli az elvesztegetett időrések számának minimalizálásához. Emiatt és más okok miatt is szükséges, hogy az időrés-menedzsmentet az egyik földi állomás, az **MCS (Master Control Station – mester vezérlőállomás)** oldja meg.

Az ACTS alapvető működése egy folytonos, háromlépéses művelet, amelyben minden lépés 1 ms-ot vesz igénybe. Az 1. lépésben a műhold vesz egy keretet, és eltárolja az 1728 bejegyzés méretű memóriájában. A 2. lépésben egy beépített számítógép átmásolja a bemeneti bejegyzéseket a megfelelő kimeneti bejegyzésekbe (lehet, hogy a másik antennára). A 3. lépésben a kimeneti keretet elküldi a leirányú csatornán.

Kezdetben minden állomáshoz legalább egy időrés van rendelve. Újabb csatornák lefoglalásához (újabb hanghívások esetén) az állomás rövid üzenetet küld az MCS-nek. Hasonlóan, az MCS-nek küldött üzenettel ereszthet el egy-egy csatornát az állomás. Ezek az üzenetek néhány nem-adat bitet használnak fel, és egy speciális vezérlőcsatornát hoznak létre az MCS felé, amelynek kapacitása állomásonként körülbelül 13 üzenet/s. A csatornák egyediek, nem kell értük versenyezni.

Dinamikus TDM réskiosztás szintén lehetséges. Az alábbiakban három sémát fogunk tárgyalni. Ezek mindegyikében a TDM kereteket időrésekre osztják, ahol minden időrésnek (időszakos) tulajdonosa van. Kizárólag a tulajdonosa használhatja az adott időrést.

Az első séma feltételezi, hogy több időrés áll rendelkezésre, mint ahány állomás van, így mindegyik állomásnak van egy saját „otthoni" időrése (Binder, 1975). Ha az állomások számánál több időrés van, akkor a fennmaradó rések senkihez sem tartoznak. Ha az aktuális csoportban egy állomás nem tart igényt időrésére, akkor az üresen marad. Az üres időrés egy jelzés a többi állomás felé, hogy a tulajdonosának nincs több küldenivalója. A következő keretben az időrés mindenki számára versengéses (ALOHA) alapon elérhetővé válik.

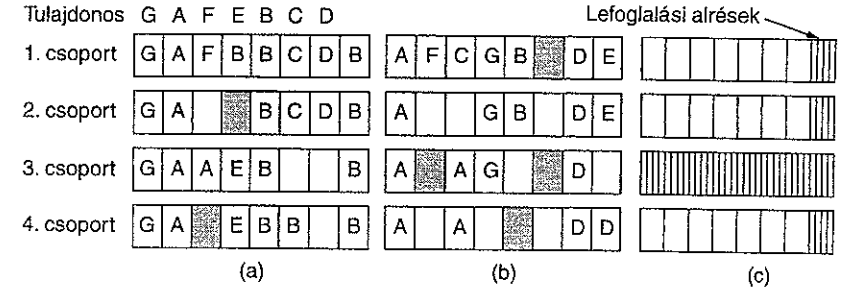
Ha a tulajdonos vissza akarja szerezni saját „otthoni" időrését, akkor továbbít egy keretet, így tehát ütközést hoz létre (ha volt egyéb forgalom). Ütközés után, a következő keretben a tulajdonost leszámítva egyik állomás sem használhatja az adott időrést. A tulajdonos tehát legrosszabb esetben is legfeljebb két keretidő alatt megkezdheti a

forgalmazást. Alacsony csatornakihasználás esetén a rendszer nem működik olyan hatékonyan, mint a réselt ALOHA, mivel minden ütközés után az ütköző feleknek várniuk kell egy keretet, hogy lássák, a tulajdonos akarja-e visszaszerezni az időrést. A 4.51.(a) ábra egy nyolc időréses keretet mutat be, amelyből hét sor a *G*, *A*, *F*, *E*, *B*, *C* és *D* birtokában vannak, míg a nyolcadik időrés senkihez sem tartozik, így küzdhetnek érte.

A második séma akkor használható, ha az állomások száma nem ismert és változó (Crowther és mások, 1973). Ebben a módszerben a réseknek nincs állandó tulajdonosa, mint a Binder esetén. Ezzel szemben az állomások réselt ALOHA használatával versengenek a résekért. Amikor egy átvitel sikeres, a következő keretben a sikeres átvitelt megvalósító állomás számára van fenntartva az adott időrés. Így addig, amíg egy állomásnak van elküldendő adata, addig korlátlanul folytathatja a küldést (persze néhány „ne légy önző" szabály betartása mellett). Lényegében az ajánlás a réselt ALOHA és a TDM egy dinamikus összekeverését teszi lehetővé azzal, hogy a két eljáráshoz rendelt rések száma igény szerint változik. A 4.51.(b) ábra egy keretet mutat be nyolc időréssel. Kezdetben *E* az utolsó időrést használja, de két keret után többé már nincsen rá szüksége. Egy keret erejéig üresen marad az időrés, majd *D* kapja el, és meg is tartja addig, amíg nem végez.

Egy harmadik séma, amely Roberts (1973) nevéhez fűződik, azt igényli az állomások részéről, hogy előre jelezzék igényüket, mielőtt adnának. Mindegyik keret tartalmaz valamennyi, mondjuk egy speciális időrést (a 4.51.(c) ábrán az utolsó), amelyet *V* kisebb alrésre osztanak, amelyeket a lefoglalásokhoz használnak. Amikor egy állomás adatokat akar küldeni, szétszór egy rövid lefoglaló keretet az egyik, véletlenszerűen kiválasztott alrésben. Ha a lefoglalás sikeres (vagyis nincs ütközés), akkor a következő normális időrést (időréseket) lefoglalta. Állandóan mindenkinek figyelnie kell, hogy éppen milyen hosszú a sor (a lefoglalt időrések száma), így ha egy állomás sikeresen foglal le magának egy időrést, tudni fogja, hogy mennyi adatidőrést kell várnia a küldés előtt. Az állomásoknak nem kell figyelniük, hogy *kik* állnak sorban, csak annyit kell tudniuk, hogy milyen hosszú a sor. Amikor a sor hossza nullára csökken, mindegyik időrés visszaáll lefoglalási alrésekké, hogy felgyorsítsák a lefoglalási eljárást.

Annak ellenére, hogy a TDM-et széles körben használják, akár lefoglalással, akár



4.51. ábra. *Lefoglalási sémák. (a) Binder. (b) Crowther. (c) Roberts. A besötétített rések ütközést jelölnek. Mindegyik sémához négy jellemző időréscsoportot mutatunk be*

anélkül, ennek is vannak hátulütői. Először is elvárja, hogy az összes állomás időben szinkronizált legyen, ami nem teljesen triviális probléma, mivel a műholdak a pályájukról időnként lesodródnak, ami megváltoztatja a földi állomások felé a terjedési időt. Az is szükséges, hogy az összes földi állomás képes legyen kiemelkedően magas löketsebességekre. Például, ha egy ACTS állomásnak mindössze egy 64 kb/s-os csatornája van, akkor képesnek kell lennie 64 bitet kiadni egy 578 ns hosszú időrés alatt. Másként fogalmazva, tulajdonképpen 110 Mb/s-on kell működnie. Ezzel szemben egy 64 kb/s-os FDM állomás tényleg 64 kb/s-on működik.

4.6.5. CDMA

Az utolsó séma a CDMA. A CDMA egyaránt megoldja az időszinkronizáció és a csatornafoglalás problémáját. Teljesen decentralizált és dinamikus.

Mindazonáltal van három hátránya. Először, a CDMA csatorna kapacitása zaj és nem-koordinált állomások jelenléte esetén tipikusan alacsonyabb, mint amit TDM-mel elérhetünk. Másodszor, 128 töredék/s (egy általános érték) mellett a töredéksebesség akkor is nagy lesz, ha a bitsebesség nem is magas, ami gyors (vagyis drága) adóvevőket tesz szükségessé. Harmadszor, csak néhány gyakorló mérnök ismeri ténylegesen a CDMA-t, ami nem növeli felhasználásának esélyét még akkor sem, ha egyes alkalmazások számára ez lenne a legjobb módszer. Mindezek ellenére a hadsereg már évtizedek óta használ CDMA-t, és egyre kedveltebbé válik a kereskedelmi alkalmazások terén is.

4.7. Összefoglalás

Néhány hálózat csak egyetlen csatornával rendelkezik, és minden kommunikáció ezt használja. Ezekben a hálózatokban a tervezés kulcskérdése az, hogy a csatornát hogyan osszuk ki a használatáért versengő állomások között. A fontosabb csatornakiosztási módszerek összefoglalója a 4.52. ábrán található.

A legegyszerűbb kiosztási séma az FDM és a TDM. Ezek akkor hatékonyak, ha az állomások száma kicsi, és a forgalom állandó. Ilyen körülmények között mindkettőt elterjedten alkalmazzák, például telefontrönkökként használt műholdas kapcsolatok sávszélességének kiosztásához.

Amikor az állomások száma nagy és változó, vagy amikor a forgalom lökésszerűen változó, akkor az FDM és a TDM használata nem szerencsés. Alternatívaként az ALOHA protokoll jelent meg réseléssel vagy anélkül, és vezérléssel vagy anélkül. Az ALOHA-t, annak változatait és leszármazottait, amelyeket széles körben alkalmaznak valós rendszerekben, részletesen tárgyaltuk és elemeztük.

Ha a csatorna állapota érzékelhető, akkor az állomások elkerülhetik, hogy más állomások adása közben kezdjenek adni. Ez a módszer, amelyet csatornafigyelésnek neveznek, egy egész sor LAN-ban és MAN-ban használatos protokoll kialakulását segítette elő.

Módszer	Leírás
FDM	Minden állomáshoz saját frekvenciasáv tartozik
TDM	Minden állomáshoz saját időrés tartozik
Tiszta ALOHA	Nem szinkronizált forgalmazás tetszőleges pillanatban
Réselt ALOHA	Véletlenszerű forgalmazás a jól meghatározott időrésekben
1-perzisztens CSMA	Szabályos csatornafigyeléses többszörös hozzáférés
Nemperzisztens CSMA	Véletlenszerű várakozás, ha a csatorna foglalt
P-perzisztens CSMA	CSMA, amely p valószínűséggel perzisztens
CSMA/CD	CSMA, amely leáll, ha ütközést érzékel
Bit-térkép	Ciklikus ütemezés bit-térkép használatával
Bináris visszaszámlálás	A legmagasabb sorszámú kész állomás adhat
Fabejárás	Csökkentett versenyhelyzet kialakítása szelektív engedélyezéssel
Hullámhossz osztás	Dinamikus FDM séma optikai szálakhoz
MAC, MACAW	Vezeték nélküli LAN protokollok
GSM	FDM plusz TDM cella-rádiózáshoz
CDPD	Csomagküldő rádió egy AMPS csatornán
CDMA	Mindenki egyszerre beszél, de különböző nyelven
Ethernet	CSMA/CD bináris exponenciális visszalépéssel
Vezérjeles sín	Logikai gyűrű fizikai sínen
Vezérjeles gyűrű	Kapd el a vezérjelet, hogy keretet küldhess
DQDB	Elosztott sorbaállás egy két-sínes LAN-on
FDDI	Optikai szálas vezérjeles gyűrű
HIPPI	Keresztrudas kapcsolás 50-100 csavart érpárral
Optikai csatorna	Keresztrudas kapcsolás optikai szálakkal
SPADE	FDM dinamikus csatornakiosztással
ACTS	TDM központi időrés-kiosztással
Binder	TDM, mellette ALOHA, ha a tulaj nem érdekelt
Crowther	ALOHA, de az időrés birtokosa megtarthatja a következő rést is
Roberts	A csatornaidő előre lefoglalása ALOHA versengéssel

4.52. ábra. *Csatornakiosztási módszerek és rendszerek közös csatornákhoz*

Ismert a protokolloknak egy olyan osztálya, amely kiküszöböli a versenyhelyzetet, vagy legalábbis jelentősen csökkenti. A bináris visszaszámlálás teljesen megszünteti a versengést. A fabejáró protokoll azzal csökkenti le a versengést, hogy az állomásokat dinamikusan két különálló csoportra osztja, az egyik csoport adhat, a másik nem. Ad-

dig végzi az osztásokat, amíg végül csak egyetlen küldésre kész állomás kapja csak meg az engedélyt.

A vezeték nélküli LAN-oknak megvannak a saját problémáik és megoldásaik. A legnagyobb problémát a rejtett állomások jelentik, így a CSMA nem működik. A megoldások egyik osztálya a MACA, amely megpróbál átvitelt létrehozni a vevő környékén, hogy a CSMA jobban működjön.

A hordozható számítógépek és telefonkészülékek számára a cellarádiózás jelenti a jövő technológiáját. A GSM, a CDPD és a CDMA széles körben használt.

Az IEEE 802 LAN-ok: CSMA/CD, vezérjeles sín, vezérjeles gyűrű. Mindegyiküknek megvannak az egyedi előnyei és hátrányai, és mindegyik megtalálta a saját felhasználói körét, amelyet várhatóan még évekig fog szolgálni. Egyetlen LAN szabvány létrehozása lehetetlen. Egy új tag a családban a DQDB, amelyet sok városban adtak el MAN hálózatként.

Egy szervezet, amelynek több LAN-ja is van, legtöbbször szeretné azokat hidakkal összekapcsolni. Amikor egy híd két vagy több LAN-t kapcsol össze, újabb problémák merülnek fel, amelyek egy része megoldhatatlan.

Amíg a 802 LAN-ok a mindennapok igás lovai, addig az FDDI, a gyors Ethernet, a HIPPI és a fényvezető szálas csatorna a versenylovak. Ezek mindegyike 100 Mb/s-os vagy nagyobb sávszélességet ajánl fel.

Végül, a műholdas hálózatok szintén többszörös hozzáférésű csatornákat használnak (felirányban). A legkülönbözőbb csatornakiosztási módszereket használják ezen a területen, beleértve az ALOHA-t, az FDM-et, a TDM-et és a CDMA-t.

Feladatok

1. Egy N állomásból álló csoport egyetlen, 56 kb/s-os, egyszerű ALOHA csatornán osztozik. Az állomások 1000 bites kereteiket 100 ms-onként küldik el még akkor is, ha az előző kereteiket sem tudták elküldeni (pl. az állomások pufferelnek). Mekkora lehet N maximális értéke?

2. Vegyük a késleltetést egyszerű és réselt ALOHA esetén, ha a terhelés kicsi! Melyik a kisebb? Adjunk magyarázatot!

3. Tízezer repülőjegy-foglaló állomás egyetlen réselt ALOHA csatorna használatáért verseng. Egy átlagos állomás 18 kérést ad ki óránként. Egy rés hossza 125 μs. Megközelítőleg mekkora a teljes csatornaterhelés?

4. Egy nagyszámú ALOHA felhasználókból álló populáció, az eredeti és az újraadásokat együtt számolva, 50 kérést bocsát ki másodpercenként. Az időrések 40 ms hosszúak.

   (a) Mi az első kísérlet sikerességének valószínűsége?

   (b) Mi a valószínűsége pontosan k ütközésnek, majd utána a sikernek?

   (c) Mi az adási kísérletek számának várható értéke?

5. Egy végtelen populációjú réselt ALOHA rendszer mérései azt mutatják, hogy a rések 10%-a tétlen.

   (a) Mekkora a G csatornaterhelés?

   (b) Mekkora az áteresztőképesség?

   (c) A csatorna kihasználatlan vagy túlterhelt?

6. Egy végtelen populációjú réselt ALOHA rendszerben egy állomásnak átlagosan 4 rést kell várnia egy ütközés és az azt követő újraküldés között. Rajzoljuk fel a rendszer késleltetés görbéjét az áteresztőképesség függvényében.

7. Egy LAN Mok- és Ward-féle bináris visszaszámlálást használ. Egy adott pillanatban a tíz állomás a következő virtuális állomásszámot viseli: 8, 2, 4, 5, 1, 7, 3, 6, 9 és 0. A következő három állomás, amelyik küld, sorban 4, 3 és 9. Mik lesznek az új virtuális állomásszámok, miután mindhárom állomás befejezte az átvitelt?

8. Az adaptív fabejáró protokoll alkalmazásával tizenhat állomás verseng egy csatorna használatáért. Ha az összes olyan állomás, amelynek prím száma van, egyszerre kerül adásra kész állapotba, akkor mennyi bit-résre van szükség a versengés feloldásához?

9. Egy megosztott kábel használati jogáért 2^n állomás egy csoportja verseng adaptív fabejárás protokoll használata mellett. Egy adott pillanatban kettő közülük adásra kész lesz. Minimálisan, maximálisan, illetve átlagosan hány időrés szükséges a fa bejárásához, ha $2^n \gg 1$?

10. A tanult vezeték nélküli LAN-ok CSMA/CD helyett inkább olyan protokollokat használtak, mint az MACA. Milyen körülmények mellett lehetne CSMA/CD-t használni?

11. Milyen tulajdonságok közösek a WDMA és GSM csatornakiosztási protokollokban?

12. Határozzuk meg, hogy a 4.14. ábra keretezési struktúráját használva, milyen gyakran küldhet egy adott felhasználó keretet!

13. Képzeljük el, hogy A, B és C egyidejűleg 0 biteket továbbít egy CDMA rendszerben a 4.16.(b) ábra töredéksorozatainak felhasználásával! Mi lesz az eredményül kapott töredéksorozat?

14. Amikor a CDMA töredéksorozatok ortogonalitásáról beszéltünk, kijelentettük, hogy ha $\mathbf{S} \bullet \mathbf{T} = 0$, akkor $\mathbf{S} \bullet \overline{\mathbf{T}}$ szintén 0. Bizonyítsuk be!

15. Nézzük más szemszögből a CDMA töredéksorozatainak ortogonális tulajdonságát! A sorozatokban a megfelelő bitek vagy megegyeznek, vagy nem. Fejezzük ki az ortogonális tulajdonságot az egyezések és különbségek felhasználásával!

16. Egy CDMA vevő a következő sorozatot fogja: (–1 +1 –3 +1 –1 –3 +1 +1). A 4.16.(b) ábra töredéksorozatait feltételezve melyik állomások adtak, és milyen biteket?

17. Egy hétemeletes irodaház minden szintjén 15 szomszédos iroda van. Minden iroda elülső oldalán van egy terminálaljzat, így azok a függőleges síkon rácsozatot alkotnak. Az aljzatok mind függőlegesen, mind vízszintesen 4 m távolságban vannak egymástól. Ha feltesszük, hogy bármelyik két aljzat között kihúzható egyenes vonalban kábel, akkor hány méter kábelre van szükség az összes aljzat bekötéséhez, ha a használt hálózat:

(a) Egy csillag összeállítás, egyetlen routerrel a közepén?

(b) Egy 802.3 LAN?

(c) Gyűrűhálózat, amely nem használ vonalközpontot?

18. Mekkora a baud-sebessége egy szabványos 10 Mb/s-os 802.3 LAN-nak?

19. Egy 1 km hosszú, 10 Mb/s-os CSMA/CD LAN-on (nem 802.3) a terjedési sebesség 200 m/μs. Az adatkeretek 256 bit hosszúak, amely magában foglalja a a fejrész, az ellenőrző összege és az egyéb, nem adat jellegű információk 32 bitjét is. Egy sikeres átvitelt követően az első rés a vevőnek van fenntartva azért, hogy egy 32 bites nyugtakeretet küldhessen a feladónak. Feltételezve, hogy nincsenek ütközések, mekkora a hasznos (fejléc nélküli) adatátviteli sebesség?

20. Két CSMA/CD állomás hosszú (többkeretes) állományok elküldésével próbálkozik. Miután egy keretet valamelyikük elküldött, a bináris exponenciális visszalépéses algoritmus használatával ismét versenyezni kezdenek a csatorna használatáért. Mi a valószínűsége annak, hogy a versenyhelyzet k kör után feloldódik, és mi a versengési periódusonként szükséges körök várható száma?

21. Képzeljünk el egy 1 km hosszú CSMA/CD hálózatot, amely 1 Gb/s sebességen működik, és nem tartalmaz ismétlőket! A jel sebessége a kábelben 200 000 km/s. Mekkora a minimális kerethossz?

22. Vázoljuk fel a 0001110101 bitsorozat Manchester-kódját!

23. Vázoljuk fel az előző feladat differenciális Manchester-kódját! Feltételezzük, hogy a vonal kezdetben alacsony állapotban van!

24. Egy vezérjeles sínt alkalmazó rendszer a következőképpen működik: amikor a vezérjel megérkezik egy állomáshoz, akkor az időzítő óra nullázódik; az állomás ezután a 6-os prioritású kereteket kezdi el küldeni egészen addig, amíg az óra értéke *T6* nem lesz; ekkor a 4-es prioritású keretekre vált át, és *T4* időpontig ezeket a kereteket küldi. Ez az algoritmus ismétlődik a 2-es és 0-s prioritásokra is. Ha minden állomásban a *T6*, ..., *T0* időzítési értékek rendre 40, 80, 90, 100 ms-nak felel meg, akkor a teljes sávszélesség hányad része jut az egyes prioritási osztályokra?

25. Mi történik egy vezérjeles sínen akkor, ha az állomás a vezérjel megszerzése után azonnal tönkremegy? Hogyan kezeli le az ismertetett protokoll ezt az eseményt?

26. Hány méter kábel okoz 5 Mb/s átviteli sebesség és 200 m/μs jelterjedési sebesség mellett ugyanakkora 1 bites késleltetést, mint egy vezérjeles gyűrű interfésze?

27. Egy vezérjeles gyűrűben az összes késleltetésnek elég nagynak kell lennie ahhoz, hogy a teljes vezérjelet magába foglalhassa. Ha a vezeték nem elég hosszú, akkor mesterséges késleltetéseket kell beszúrni. Magyarázzuk meg, hogy miért fontos ez az extra késleltetés egy 16 bitnyi késleltetést tartalmazó gyűrűben, ahol a vezérjel hossza 24 bit!

28. Egy nagyon erősen terhelt 1 km hosszú, 10 Mb/s-os vezérjeles gyűrű 200 m/μs-os terjedési sebességű. A gyűrű mentén, egymástól egyenlő távolságra 50 állomás helyezkedik el. Az adatkeretek a 32 bites nem adat jellegű információkkal együtt 256 bitesek. A nyugtát maga az adatkeret hordozza, így gyakorlatilag nem igényel plusz „overhead"-et. A vezérjel 8 bites. Ennek a gyűrűnek vagy egy 10 Mb/s-os CSMA/CD hálózatnak nagyobb a hasznos adatátviteli sebessége?

29. Egy vezérjeles gyűrűben az adó távolítja el a kereteket. Milyen módosításokat igényelne a rendszer ahhoz, hogy a vevő távolíthassa el a kereteket, és milyen következményekkel járna?

30. Egy 4 Mb/s-os vezérjeles gyűrűnek 10 ms-os vezérjeltartási ideje van. Mekkora a gyűrűn küldhető leghosszabb keret?

31. Van-e a vonalközpontoknak bármiféle hatása egy vezérjeles gyűrű működésére?

32. Egy fényvezető szálas vezérjeles gyűrű, amelyet MAN-ként használnak, 200 km hosszú és 100 Mb/s sebességen üzemel. Miután egy állomás elküldött egy keretet, addig nem rakja vissza a vezérjelet, amíg a keretet le nem csapolta a gyűrűről. A fényvezető szálban a jelterjedési sebesség 200 000 km/s, a leghosszabb keret pedig 1 kilobájt hosszú. Mekkora a gyűrű maximális hatékonysága (leszámítva minden egyéb „overhead"-et)?

33. A 4.32. ábrán a *D* állomás küldeni szeretne. Melyik állomásnak szánja a keretet?

34. A 4.32. ábrán látható rendszer üres állapotban van. Kicsivel később a *C*, *A* és *B* állomások, ebben a sorrendben válnak egymás után rövid időn belül küldésre készszé. Feltételezve, hogy az adatkeretek küldése csak a három kérelem elküldése után kezdődik csak meg, mutassuk meg *RC* és *CD* értékeit minden egyes kérés, illetve adatkeret után!

35. Néha az Ethernetre azt mondják, hogy nem megfelelő valós idejű számításokhoz, mivel az újraküldési intervallumnak a legrosszabb esetben nincs felső korlátja. Milyen körülmények között lehet ugyanezt a kifogást emelni vezérjeles gyűrű esetén? Milyen körülmények között rendelkezik a vezérjeles gyűrű ismert legrosszabb esettel? Feltesszük, hogy az állomások száma változatlan és ismert.

36. Ethernet hálózatok esetén a kereteknek legalább 64 bájt hosszúnak kell lennie, hogy az adó-vevő biztosan adjon akkor is, ha a vezeték távoli részein következik be ütközés. A gyors Ethernet minimális kerethossza szintén 64 bájt, pedig a biteket tízszer gyorsabban adja. Hogyan lehetséges, hogy ugyanazt a minimális kerethosszt használják?

37. Képzeljünk el két LAN hidat, amelyek mindegyike két 802.4 hálózatot köt össze. Az egyiknek másodpercenként 1000 512 bájtos keretet kell lekezelnie, míg a másiknak ugyanennyi idő alatt 200 4096 bájtos keretet. Melyik hídnak kell, hogy gyorsabb CPU-ja legyen? Indokoljunk!

38. Képzeljük el, hogy az előbbi feladat két hídja egy-egy 802.4 és 802.5 LAN-t köt össze. Befolyásolja-e ez a változtatás az előbbi feladatra adott választ?

39. Egy 802.3 és egy 802.4 LAN-t egy olyan híd köt össze, amelyik memóriájában időszakos hibák vannak. Okozhat ez olyan hibákat, amelyeket nem lehet érzékelni, vagy mind észrevehető az ellenőrző összegből?

40. Egy egyetem számítógép-tudományi tanszékén 3 Ethernet szegmens található, amelyeket két transzparens híddal lineáris hálózatba kötöttek. Egy nap a hálózat adminisztrátora kilép, és helyére sietve a számítóközpontból lép valaki. A számítóközpont egy IBM vezérjeles gyűrűt üzemeltet. Az új adminisztrátor észreveszi, hogy a hálózat két vége nincs összekötve, és gyorsan rendel még egy transzparens hidat, amellyel összeköti a szabadon maradt végeket. Mi történik ilyenkor?

41. Egy nagy FDDI gyűrűbe 100 állomás csatlakozik, a vezérjel körbejárási ideje pedig 40 ms. A vezérjeltartási idő 10 ms. Mi a gyűrűvel elérhető maximális hatékonyság?

42. Képzeljük el, hogy szuperszámítógépek csatlakoztatására építünk interfészt a HIPPI megközelítés, de modern technológia felhasználásával. Az adatútvonal most 64 bit széles, és egy-egy szó küldhető minden 10 ns-ban. Mekkora a csatorna sávszélessége?

43. A szövegben azt állítottuk, hogy két, réselt ALOHA-t használó felirányú csatorna mellett a műhold leirányú csatornájának a kihasználása 0,736 lehet, amennyiben végtelen puffert feltételezünk. Mutassuk meg, hogyan jött ki ez az eredmény!

5. A hálózati réteg

A hálózati réteg feladata, hogy a csomagokat a forrástól egészen a célig eljuttassa. Ehhez esetleg több routeren is keresztül kell a csomagnak haladnia. Ez a feladat láthatóan elkülönül az adatkapcsolati réteg feladatától, amely ennél szerényebb: azaz keretek továbbítása egy vonal egyik végétől a másikig. Ezért a hálózati réteg a legalacsonyabb réteg, amely két végpont közti átvitellel foglalkozik. További információért lásd (Huitema, 1995; Perlman, 1992) műveit.

E célok elérése érdekében a hálózati rétegnek ismernie kell a kommunikációs alhálózat (vagyis a routerek halmaza) topológiáját, és megfelelő útvonalakat kell találnia azon keresztül. Arra is ügyelnie kell, hogy úgy válassza ki a routereket, hogy elkerülje néhány kommunikációs vonal és router túlterhelését, míg mások tétlenül maradnak. Végül, amikor a forrás és a cél különböző hálózatokban vannak, a hálózati réteg dolga az, hogy foglalkozzon e különbözőségekkel, és megoldja az ezekből adódó problémákat. Ebben a fejezetben ezeket a kérdéseket fogjuk tanulmányozni és két működő példánkkal illusztrálni. Ezek az Internet és az ATM.

5.1. A hálózati réteg tervezési kérdései

A következő szakaszok bevezetnek néhány olyan kérdésbe, amelyekkel a hálózati réteg tervezőjének meg kell birkóznia. Ezek közt találjuk a szállítási rétegnek nyújtott szolgálatot és az alhálózat belső tervezését.

5.1.1. A szállítási rétegnek nyújtott szolgálatok

A hálózati réteg a hálózati réteg/szállítási réteg interfészen nyújtja szolgálatait a szállítási rétegnek. Ez az interfész gyakran még valami más ok miatt is különösen fontos: sokszor ez ad csatlakozási lehetőséget a szolgáltató és az ügyfél közt is, vagyis ez az alhálózat határa. A szolgáltató felügyeleti joga gyakran a hálózati rétegig terjedő protokollokra és interfészekre terjed ki, a feladata pedig az, hogy az ügyfelei által rendelkezésre bocsátott csomagokat továbbítsa. Ezen oknál fogva ezt az interfészt különösen jól kell definiálni.

A hálózati réteg szolgálatait a következő célok szem előtt tartásával tervezték:

1. A szolgálatoknak függetleneknek kell lenniük az alhálózat kialakításától.

2. A szállítási réteg elől el kell takarni a jelenlevő alhálózatok számát, típusát és topológiáját.

3. A szállítási réteg rendelkezésére bocsátott hálózati címeknek egységes számozási rendszert kell alkotniuk, még LAN-ok és WAN-ok esetén is.

Ezen célok figyelembevételével, a hálózati réteg tervezői nagy szabadsággal rendelkeznek a szállítási rétegnek nyújtandó szolgáltatások részletes specifikációinak elkészítése során. Ám ez a szabadság gyakran indulatos csatározássá válik két szembenálló csoport közt. A vita középpontjában az áll, hogy vajon a hálózati réteg összeköttetés alapú vagy összeköttetés nélküli szolgáltatást nyújtson-e.

Az egyik tábor (amelyet az Internet közössége képvisel) véleménye az, hogy az alhálózat dolga csak a bitek ide-oda mozgatása és semmi más. Az ő nézetük szerint (amelyet egy valódi, működő számítógép-hálózattal való 30 éves tapasztalat támaszt alá), az alhálózat eredendően megbízhatatlan, függetlenül annak tervezésétől. Ezért a hosztoknak a megbízhatatlanságot tényként kell elfogadniuk, és a hibavédelmet (vagyis a hibajelzést és -javítást) és a forgalomszabályozást maguknak kell elvégezniük.

Ez a nézet gyorsan ahhoz a következtetéshez vezet, hogy a hálózati szolgálatnak összeköttetés nélkülinek kell lennie, a SEND PACKET és RECEIVE PACKET primitíveken kívül alig kell valami más. Hangsúlyozottan nincs szükség a csomagok sorrendi kezelésére és forgalomszabályozásra, mert ezt a hosztok amúgy is mindenképpen megteszik, és valószínűleg kevés nyereség származik abból, ha ezeket kétszer hajtjuk végre. Továbbá, minden csomagnak hordoznia kell a teljes célcímet, mivel mindegyik elküldött csomag az előző csomagoktól függetlenül kerül továbbításra (ha egyáltalán voltak ilyenek).

A másik tábor (amelyet a telefontársaságok képviselnek) véleménye, hogy az alhálózatnak egy (viszonylag) megbízható, összeköttetés alapú szolgálatot kell nyújtania. Állításuk szerint a világméretű telefonhálózattal való 100 éves sikeres gyakorlat jó alapot ad ehhez. Ezen nézet szerint az összeköttetéseknek a következő tulajdonságokkal kell rendelkezniük:

1. Adatok elküldése előtt a küldő oldalon levő hálózati rétegbeli folyamatnak egy összeköttetést kell felépítenie a vevő oldali társfolyamattal. Ezt az összeköttetést, amelynek egy speciális azonosítót adnak, addig használják, amíg az összes adatot el nem küldték, majd a felek lebontják azt.

2. Amikor az összeköttetés felépül, a két folyamat tárgyalásokba bocsátkozhat a nyújtandó szolgálat paramétereiről, minőségéről és áráról.

3. Az összeköttetés kétirányú, és a csomagok sorrendhelyesen kerülnek kézbesítésre.

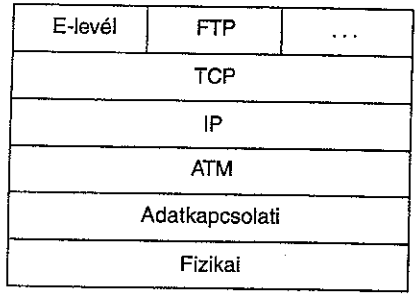
4. A forgalomszabályozás automatikusan biztosított, hogy egy gyors adó ne küldhesse gyorsabban csomagjait a csőbe, mint ahogy a vevő azokat ki tudná venni, azaz ne legyen túlcsordulás.

Más tulajdonságok, mint például a garantált kézbesítés, a kézbesítés visszaigazolása, és a magas prioritású csomagok, opcionálisak. Mint ahogy az 1. fejezetben szó volt róla, az összeköttetés nélküli szolgálat a postára, míg az összeköttetés alapú szolgálat a telefonrendszerre hasonlít.

Az összeköttetés alapú és összeköttetés nélküli szolgálatok közti vita hátterében az áll, hogy hova helyezzük az összetettséget. Az összeköttetés alapú szolgálatban ez a hálózati rétegben (az alhálózatban) van; az összeköttetés nélküli szolgálatban pedig a szállítási rétegben (a hosztokban). Az összeköttetés nélküli szolgálat támogatói szerint a felhasználók által elérhető számítási teljesítmény olcsó, tehát nincs ok nem ide tenni az összetettséget. Egy további érvük az, hogy az alhálózat egy nagy, évtizedekre szóló nemzeti (vagy nemzetközi) beruházás, tehát nem szabad olyan szolgáltatásokat belezsúfolni, amelyek gyorsan elavulnak, de az árakba még sok évig bele kell őket számolni. Ezenkívül néhány alkalmazásnak, mint például a digitalizált hangnak és a valós idejű adatgyűjtésnek, sokkal inkább fontos a *gyors,* mint a *pontos* kézbesítés.

Másfelől, az összeköttetés alapú szolgáltatás támogatói szerint a legtöbb felhasználónak nem érdeke összetett szállítási rétegbeli protokollokat futtatni a gépeiken. Amit ők akarnak, az egy megbízható, gondoktól mentes szolgálat, és ez a szolgálat legjobban a hálózati rétegbeli összeköttetésekkel biztosítható. Továbbá, néhány szolgáltatást, mint a valós idejű hang- és képátvitelt sokkal könnyebb egy összeköttetés alapú hálózati réteg felett megvalósítani, mint egy összeköttetés nélküli hálózati réteg felett.

Bár ezen vitákban ritkán esik erről szó, de két különálló poblémakörről van szó. Az első az, hogy a hálózat összeköttetés alapú (összeköttetés-felépítés szükséges) vagy összeköttetés nélküli (nincs szükség összeköttetés felépítésére). Másodszor, hogy megbízható (nincsenek elveszett, megkettőzött vagy összekevert csomagok), vagy megbízhatatlan (a csomagok elveszhetnek, megkettőződhetnek vagy összekeveredhetnek). Elméletben mind a négy kombináció létezik, de a két legfontosabb kombináció a



5.1. ábra. *TCP/IP futtatása ATM alhálózat felett*

megbízható összeköttetés alapú és a megbízhatatlan összeköttetés nélküli, a másik kettő lényegtelen.

Eme két tábort a két működő példánk képviseli. Az Internetnek összeköttetés nélküli hálózati rétege van, az ATM hálózatoknak összeköttetés alapú. Felvetődik egy nyilvánvaló kérdés: hogyan működik az Internet, amikor egy ATM alapú szolgáltató által biztosított alhálózaton fut. A válasz az, hogy a forráshoszt először egy ATM hálózati rétegbeli kapcsolatot hoz létre a célhoszttal, és ezután független IP csomagokat küld át rajta, ahogy az 5.1. ábrán látszik. Bár ez a megközelítés működik, de nem hatékony, mivel bizonyos tevékenységek mindkét rétegben szerepelnek. Például az ATM hálózati rétege biztosítja, hogy a csomagok mindig sorrendhelyesen kerüljenek kézbesítésre, ugyanakkor a TCP kód is tartalmazza a teljes eljárást a sorrenden kívül érkezett csomagok kezelésére és helyes sorrendbe állítására. Arról, hogy hogyan futtassunk IP-t ATM fölött, lásd: RFC 1577 és (Armitage és Adams, 1995).

5.1.2. A hálózati réteg belső szervezése

Miután láttuk, hogy a hálózati réteg kétfajta szolgálatot tud a felhasználóinak nyújtani, ideje, hogy megnézzük, hogyan működik belül. Alapvetően kétfajta filozófia létezik az alhálózat megszervezésére: az egyik, amelyik összeköttetéseket használ, és a másik, amelyik összeköttetések nélkül működik. Ha az alhálózat *belső* működéséről beszélünk, akkor az összeköttetéseket rendszerint **virtuális áramköröknek (virtual circuits)** nevezzük, a telefonrendszer által létrehozott fizikai áramkörök mintájára. Az összeköttetés nélküli szervezetek függetlenül elküldött csomagjait pedig **datagramoknak (datagrams)** nevezzük, a távirat (telegram) mintájára.

A virtuális áramköröket rendszerint olyan alhálózatokban használják, amelyek elsődleges szolgálata összeköttetés alapú, ezért ebben az összefüggésben tárgyaljuk őket. A virtuális áramkörök mögött az az ötlet húzódik meg, hogy elkerüljük azt, hogy minden elküldött csomagnak vagy cellának újra útvonalat kelljen választani. Ehelyett, amikor egy összeköttetés felépül, ennek a felépülésnek a részeként egy útvonalat is kiválasztanak a forrásgéptől a célgépig, amelyet megjegyeznek. Ezt az útvonalat használják a teljes, ezen az összeköttetésen keresztül folyó forgalom számára, amely pontosan ugyanúgy működik, ahogyan a telefonrendszer. Amikor a kapcsolatot lebontják, a virtuális áramkör is megszakad.

Ezzel ellentétben, egy datagram alapú alhálózatban nincsenek előre meghatározott útvonalak, még akkor sem, ha maga a szolgálat összeköttetés alapú. Minden egyes csomagot az előzőektől függetlenül irányítanak. Az egymás után következő csomagok más-más útvonalat követhetnek. Bár a datagram alapú alhálózatoknak több munkát kell végezniük, rendszerint robusztusabbak és könnyebben alkalmazkodnak a hibákhoz és torlódásokhoz, mint a virtuális áramkör alapúak. A két megközelítés előnyeiről és hátrányairól később lesz szó.

Ha egy adott virtuális áramkörön keresztül haladó csomagok mindig ugyanazt az utat követik az alhálózaton keresztül, minden routernek tudnia kell, merre továbbítsa a csomagokat, amelyek valamelyik rajta keresztülhaladó, pillanatnyilag nyitott virtuális áramkörhöz tartoznak. Minden routernek egy táblázatot kell fenntartania, amelyben

minden rajta keresztülhaladó nyitott virtuális áramkörhöz egy bejegyzés tartozik. Minden csomagnak, amely az alhálózaton keresztülhalad, tartalmaznia kell egy, a virtuális áramkör számának fenntartott mezőt a fejrészében, a sorszámok, ellenőrző összegek és ehhez hasonló dolgok mellett. Amikor egy csomag egy routerhez érkezik, a router tudja, melyik vonalon érkezett, és mi a virtuális áramkör száma. Csupán ezen információ alapján a csomagot a helyes kimenő vonalon kell továbbítani.

Amikor egy hálózati összeköttetést felépítenek, egy, az adott gépen még nem használt virtuális áramkörszámot választanak az összeköttetés azonosítójának. Mivel minden gép függetlenül választja meg a virtuális áramkörszámokat, ezek a számok csak helyi érvényességgel bírnak. Ha az egész hálózatra érvényesek lennének, valószínű, hogy két virtuális áramkör, amelyeknek azonos a virtuális áramkörszáma, keresztülhaladna ugyanazon közbenső routeren, ez pedig kétértelműséghez vezethet.

Mivel a virtuális áramkörök bármelyik végpontból kezdeményezhetők, probléma akkor lép fel, amikor a hívásfelépítések mindkét irányban egyszerre terjednek egy router-láncon. Egy ponton szomszédos routerekhez fognak érkezni, ahol mindegyik routernek egy virtuális áramkörszámot kell választania a létrehozni kívánt (duplex) áramkörhöz. Ha úgy programozták be őket, hogy az adott vonalon használaton kívüli legalacsonyabb számot válasszák, akkor ugyanazt a számot fogják adni, így két független virtuális áramkörnek lesz ugyanaz a száma ugyanazon a fizikai vonalon. Ha később egy adatcsomag érkezik, az ezt vevő router nem tudja eldönteni, vajon ez egy előre haladó csomag az egyik áramkörön, vagy egy visszafele haladó csomag a másikon. Ha az áramkörök szimplexek, nincs ilyen kétértelműség.

Megjegyezzük, hogy minden folyamattól megkívánják, hogy jelezze, amikor befejezte egy virtuális áramkör használatát, annak érdekében, hogy a virtuális áramköröket kitörölhessék a routerek táblázataiból, és felszabadítsák a lefoglalt területet. A nyilvános hálózatokban ez inkább fenyegetés, mint jutalmazás hatására működik: a felhasználók mindig az összeköttetési időért és a szállított adatok mennyiségéért fizetnek. Továbbá, valamiféle óvintézkedést kell tenni, ha olyan géppel van dolgunk, amelyek a virtuális áramköreiket összeomlás útján szakítják meg, ahelyett, hogy dolguk végeztével udvariasan lebontanák azokat.

Ennyit az alhálózaton belüli virtuális áramkörök használatáról. A másik lehetőség az, hogy belül datagramokat használunk. Ez esetben a routerek táblázataiban levő bejegyzések nem a rajtuk keresztülhaladó nyitott virtuális áramkörök azonosítói. Ehelyett, egy ilyen táblázat azt mutatja meg, hogy minden egyes lehetséges célrouterhez melyik kimeneti vonalat kell használni. Ezekre a táblázatokra akkor is szükség van, amikor belül virtuális áramköröket használunk a kapcsolatfelépítő csomag útvonalának meghatározására.

Minden datagramnak tartalmaznia kell a teljes célcímet. Nagy hálózatokban ezek a címek elég hosszúak lehetnek (egy vagy akár több tucat bájt). Amikor egy csomag megérkezik, a router kikeresi a használandó kimeneti vonalat, és útjára bocsátja a csomagot. A hálózati vagy szállítási rétegbeli összeköttetések létrehozása és lebontása szintén nem igényelnek a routerek részéről semmilyen külön munkát.

5.1.3. A virtuális áramkör és a datagram alapú alhálózatok összehasonlítása

Mind a virtuális áramköröknek, mind a datagramoknak megvannak a maguk támogatói és ellenzői. Mi most megkíséreljük mindkét oldal érveit összefoglalni. A főbb témák megtalálhatók az 5.2. ábrán, bár a szőrszálhasogatók bizonyára az ábra minden részére tudnának ellenpéldát találni.

Az alhálózaton belül sokfajta kompromisszum lehetséges a virtuális áramkörök és a datagramok közt. Egy kompromisszum a routerek memóriaterülete és a sávszélesség közt áll fenn. Virtuális áramkörök esetén a csomagoknak csak áramkörszámokat kell tartalmazniuk teljes célcímek helyett. Ha a csomagok általában elég rövidek, a minden csomagban jelenlevő teljes célcím jelentős mértékű többletet jelent, és ezáltal sávszélesség veszhet el. A virtuális áramkörök használatáért fizetendő ár a routereken belüli táblázat helyigénye. A kommunikációs áramkörök és a router memóriájának egymáshoz viszonyított költségétől függően egyik vagy másik lehet olcsóbb.

Egy másik kompromisszum az összeköttetés-felépítési idő és a címfeldolgozási idő közt kereshető. A virtuális áramkörök használata megkövetel egy összeköttetés-felépítési fázist, amely időbe kerül és erőforrásokat igényel. A virtuális áramkör alapú alhálózatban azonban könnyű eldönteni, mi legyen a csomagokkal: a router az áramkör számát indexként használja a táblázatokhoz, a kimenő vonal meghatározásához. Egy datagram alapú alhálózatban bonyolultabb eljárás szükséges ahhoz, hogy eldöntsük, merre menjen a csomag.

A virtuális áramköröknek van némi előnyük az alhálózaton belüli torlódás elkerülésénél, mert az erőforrásokat előre le tudjuk foglalni, amikor a kapcsolat felépül. Amikor érkezni kezdenek a csomagok, a szükséges sávszélesség és router-kapacitás

Kérdés	DG alhálózatokban	VÁ alhálózatokban
Áramkörfelépítés	Nem szükséges	Megkövetelt
Címzés	Minden csomag tartalmazza a teljes forrás- és célcímet	Minden csomag egy rövid VÁ számot tartalmaz
Állapotinformáció	Az alhálózat nem tartalmaz állapotinformációkat	Minden VÁ táblázat helyet követel az alhálózatban
Forgalomirányítás	Minden csomagot függetlenül irányítanak	Az útvonalat akkor választják ki, amikor a VÁ felépül; minden csomag ezt az útvonalat követi
A forgalomirányítók meghibásodásainak hatása	Semmi, eltekintve az összeomlás során elveszett csomagoktól	Minden VÁ megszakad, amely a csődöt mondott forgalomirányítón keresztülhaladt
Torlódásvédelem	Bonyolult	Könnyű, ha elég puffert lehet előre lefoglalni mindegyik VÁ számára

5.2. ábra. *A datagram és a virtuális áramkör alapú alhálózatok összehasonlítása*

rendelkezésre fog állni. Datagram alapú alhálózatoknál a torlódás elkerülése bonyolultabb.

A tranzakció-feldolgozó rendszereknél (mint például, amikor az üzletek hitelkártyákkal történt vásárlásokat ellenőriznek) egy virtuális áramkör felállításához és kitörléséhez szükséges többlet könnyen felülmúlhatja az áramkör valódi használatát. Ha a forgalom nagy része várhatóan ilyen típusú lesz, az alhálózaton belüli kapcsolt virtuális áramkörök használatának nincs sok értelme. Másfelől viszont, az állandó virtuális áramkörök, amelyeket kézzel hoznak létre, és hónapokig vagy évekig fennállnak, itt hasznosak lehetnek.

A virtuális áramkörök azonban sebezhetőek is. Ha egy router összeomlik, és elveszti memóriája tartalmát, még ha egy másodperccel később újra is indul, az összes rajta keresztülhaladó virtuális áramkört meg kell szakítani. Ezzel ellentétben, ha egy datagram alapú router megy tönkre, csak azok a felhasználók szenvednek kárt, akiknek a csomagjai éppen ebben a routerben álltak sorban, sőt talán nem is mindegyikük, attól függően, hogy nyugtázták-e már azokat. Egy kommunikációs vonal elvesztése végzetes a rajta keresztülhaladó virtuális áramkörök számára, de könnyen ellensúlyozható datagramok használata esetén. A datagramok arra is lehetőséget nyújtanak a routereknek, hogy kiegyenlítsék a terhelést az alhálózaton, mivel az útvonalak félúton is megváltoztathatók.

Érdemes külön rámutatni, hogy a felkínált szolgálat (összeköttetés alapú vagy összeköttetés nélküli) az alhálózat szerkezetétől független kérdés (virtuális áramkör vagy datagramok). Elméletben mind a négy kombináció lehetséges. Nyilván az összeköttetés alapú szolgálat virtuális áramkör alapú megvalósítása és az összeköttetés nélküli szolgálat datagram alapú megvalósítása értelmesek. Az összeköttetések datagramokkal való megvalósításának szintén lehet értelme, amikor az alhálózat egy igen robusztus szolgálatot igyekszik biztosítani.

A negyedik lehetőség, az összeköttetésmentes szolgálat egy virtuális áramkör alapú alhálózat tetején különösnek látszik, de előfordul. A nyilvánvaló példa az IP futtatása egy ATM alhálózat fölött. Itt kívánatos egy meglevő összeköttetés nélküli protokollt futtatni egy új, összeköttetés alapú hálózati réteg fölött. Amint már korábban említettük, ez inkább egy alkalmi, mint egy jól megtervezett megoldás a problémára. Egy új rendszerben, amit arra terveztek, hogy ATM alapú alhálózaton fusson, rendszerint

Felsőbb réteg	Alhálózat típusa	
	Datagram	Virtuális áramkör
Összeköttetés nélküli	UDP alatt IP	UDP alatt IP alatt ATM
Összeköttetés alapú	TCP alatt IP	ATM AAL1 alatt ATM

5.3. ábra. *Példák a szolgálatok és az alhálózati szerkezetek különféle kombinációira*

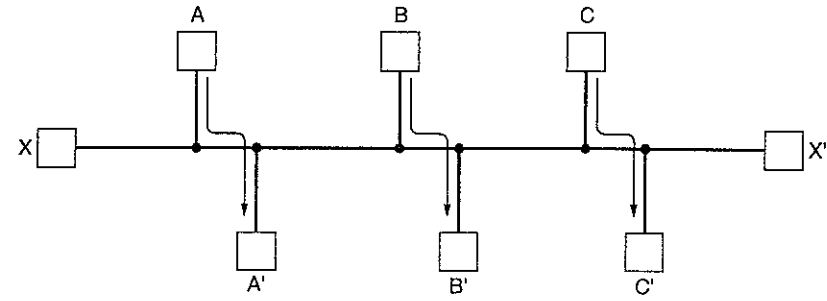
nem tennénk egy IP-hez hasonló összeköttetés nélküli protokollt egy, az ATM-hez hasonló összeköttetés alapú hálózati réteg fölé, és helyeznénk egy összeköttetés alapú szállítási protokollt az összeköttetés nélküli protokollra. Az 5.3. ábrán mind a négy esetre találhatók példák.

5.2. Forgalomirányító algoritmusok

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

A **forgalomirányító algoritmus (routing algorithm)** a hálózati réteg szoftverének azon része, amely azért a döntésért felelős, hogy egy bejövő csomag melyik kimeneti vonalon kerüljön továbbításra. Ha az alhálózat belül datagramokat használ, ezt a döntést újra meg újra meg kell hozni minden beérkező adatcsomagra, mivel a legjobb útvonal lehet, hogy változott a legutóbbi meghatározás óta. Ha az alhálózat belül virtuális áramköröket használ, a forgalomirányító döntéseket csak akkor hozzák meg, ha új virtuális áramkör épül fel. Ezután az adatcsomagok már csak az előzőleg kijelölt útvonalat követik. Ezt az utóbbi esetet néha **viszony-forgalomirányításnak (session routing)** is nevezik, mivel az útvonal érvényben marad a teljes felhasználói viszony (mint pl. egy bejelentkezési viszony egy terminálról, vagy egy fájl-átvitel) alatt.

Függetlenül attól, hogy az útvonalakat minden egyes csomagra egymástól függetlenül választják ki, vagy csak egy új összeköttetés létrejöttekor, vannak bizonyos tulajdonságok, amelyek kívánatosak egy forgalomirányító algoritmus esetében. Ezek a következők: helyesség, egyszerűség, robusztusság, stabilitás, igazságosság és optimalitás. A helyesség és az egyszerűség magától értetődő, a robusztusság azonban elsőre talán kevésbé nyilvánvaló. Amikor egy nagyobb hálózatot üzembe helyeznek, elvár-



5.4. ábra. *Konfliktus az igazságosság és az optimalitás közt*

ják, hogy az évekig rendszerszintű hibáktól mentesen működjön. Ez alatt az idő alatt mindenféle hardver- és szoftverhiba lesz. Hosztok, routerek és vonalak többször is tönkremennek és megjavulnak, és a topológia is sokszor fog változni. A forgalomirányító algoritmusnak képesnek kell lennie arra, hogy megbirkózzon a topológiában és a forgalomban bekövetkező változásokkal anélkül, hogy az összes hoszton futó összes munkát meg kelljen szakítani, és a hálózatot újraindítani, ahányszor csak valamelyik router összeomlik.

A stabilitás szintén fontos célja a forgalomirányító algoritmusnak. Léteznek algoritmusok, amelyek soha nem közelítik meg az egyensúlyi állapotot, bármilyen hosszan fussanak is. Az igazságosság és az optimalitás szintén nyilvánvalónak tűnnek – bizonyára senkinek nem lenne kifogása ellenük – de amint kiderül, gyakran egymásnak ellentmondó célok. Erre a konfliktusra egy egyszerű példa látható az 5.4. ábrán. Tegyük fel, hogy elegendő forgalom van A és A', B és B', és C és C' között ahhoz, hogy telítse a vízszintes összeköttetéseket. Hogy a folyamok összegét maximalizáljuk, az X és X' közti forgalmat teljesen el kellene zárni. Sajnos X és X' esetleg nem így látja a dolgot. Nyilvánvaló, hogy valami kompromisszumra van szükség a globális hatékonyság és az egyes állomások azonos elbírálása közt.

Mielőtt akárcsak megpróbálnánk kompromisszumot keresni az igazságosság és az optimalitás közt, el kell határoznunk, mit akarunk optimalizálni. Nyilvánvaló jelölt erre az átlagos csomagkésleltetés minimalizálása, de ugyanígy a teljes hálózati átbocsátóképesség maximalizálása is. Ez a két cél is ütközik, azonban ha bármely sorbanállási rendszert a teljesítőképessége határán működtetünk, az hosszú sorbanállási késleltetést von maga után. Kompromisszumként sok hálózatban a csomagok által megteendő ugrások számát kísérlik meg minimalizálni, mivel az ugrások számának csökkentése a késleltetést javítja és a felhasznált sávszélességet is csökkenti, ezáltal az átbocsátóképesség is javul.

A forgalomirányító algoritmusok két nagy osztályba sorolhatók: adaptív és nem adaptív algoritmusok. A **nem adaptív algoritmusok (nonadaptive algorithms)** nem támaszkodnak döntéseikben mérésekre vagy becslésekre az aktuális forgalomról és topológiáról. Ehelyett az I-től J-ig használandó útvonalat (minden I-re és J-re) előre, off-line módon számolják ki, és a hálózat indításakor letöltik a routerekbe. Ezt az eljárást néha **statikus forgalomirányításnak (static routing)** is szokták nevezni.

Ezzel ellentétben az **adaptív algoritmusok (adaptive algorithms)** úgy változtatják forgalomirányítási döntéseiket, hogy tükrözzék a topológiában és rendszerint a forgalomban is történt változásokat. Az adaptív algoritmusok különbözhetnek abban, hogy honnan kapják az információikat (helyileg, a szomszédos routerektől vagy minden routertől), mikor változtatják az útvonalakat (minden ΔT másodpercben, amikor a terhelés változik, vagy amikor a topológia változik), és milyen mértéket használnak az optimalizáláshoz (távolságot, ugrások számát vagy becsült áthaladási időt). A következő szakaszokban sokféle algoritmust fogunk tárgyalni, ezek között lesznek statikusak és dinamikusak is.

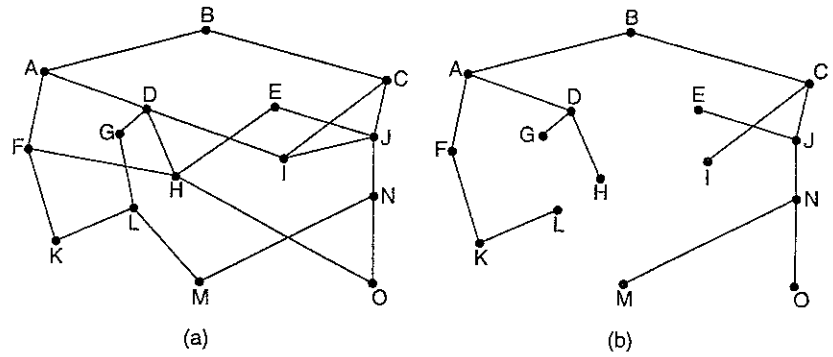
5.2.1. Az optimalitási elv

Mielőtt beleásnánk magukat az egyes algoritmusokba, segítségünkre lehet, hogy az optimális útvonalakról egy általános megállapítás tehető, a hálózat topológiájától és a forgalomtól függetlenül. Ez a megállapítás az **optimalitási elv (optimality principle)** néven ismeretes. Az állítás szerint, ha a J router az I routertől K router felé vezető optimális útvonalon helyezkedik el, akkor a J-től K-ig vezető útvonal ugyanerre esik. Hogy ezt belássuk, nevezzük az útvonal I-től J-ig tartó részét r_1-nek és az útvonal másik részét r_2-nek. Ha létezne egy r_2-nél jobb, J-től K-ig tartó útvonal, ezt összefűzhetnénk r_1-gyel, hogy az I-től K-ig tartó útvonalon is javítsunk. Ez viszont ellentmond állításunknak, miszerint r_1r_2 optimális.

Az optimalitási elv egyenes következményeként beláthatjuk, hogy az összes forrásból egy célba tartó optimális útvonalak egy fát alkotnak, amelynek gyökere a cél. Az ilyen fákat **nyelőfának (sink tree)** nevezzük. Egy ilyen látható az 5.5. ábrán, ahol a távolság mértékegysége az ugrások száma. Vegyük észre, hogy a nyelőfa nem feltétlenül egyedi; más fák is létezhetnek ugyanolyan úthosszakkal. Minden forgalomirányító algoritmus célja a nyelőfák felderítése és használata az összes router számára.

Mivel a nyelőfa valóban fa, nem tartalmaz hurkot, ezért minden csomag véges és korlátos számú ugráson belül kézbesítésre kerül. A gyakorlatban az élet nem ilyen könnyű. Kapcsolatok és routerek tönkremehetnek és megjavulhatnak működés közben, ezért különböző routereknek különböző elképzeléseik lehetnek a pillanatnyi topológiáról. Szintén csendben elmentünk azon kérdés mellett, hogy a routereknek egyénileg kell-e beszerezni az információt, amelyre a nyelőfaszámítás épül, vagy ezt az információt valami más módszerrel gyűjtik össze. Ezekre a kérdésekre hamarosan visszatérünk. Mindazonáltal az optimalitási elv és a nyelőfa egy mérőszámot adnak, amelyhez más forgalomirányító algoritmusokat viszonyíthatunk.

A következő három szakaszban három különböző statikus forgalomirányító algoritmust veszünk szemügyre, majd ezek után áttérünk az adaptív algoritmusokra.



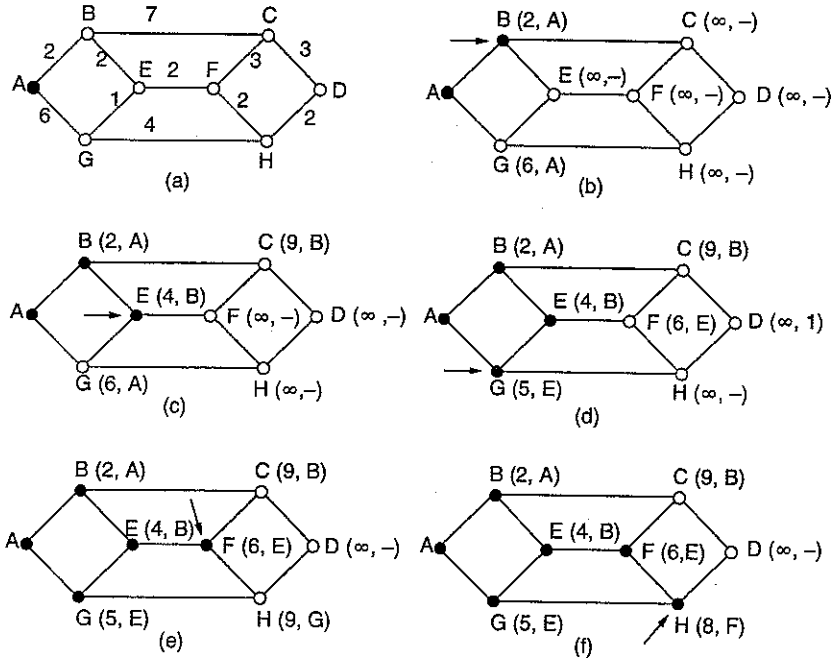
5.5. ábra. *(a) Egy alhálózat. (b) Egy nyelőfa a B routerhez*

5.2.2. Legrövidebb útvonal alapú forgalomirányítás

Kezdjük a forgalomirányító algoritmusok tanulmányozását egy olyan módszerrel, amelyet sok formában, széles körben használnak, mivel egyszerű és könnyű megérteni. Az ötlet az, hogy építsük fel az alhálózat egy gráfját, ahol minden router egy csomópontnak, és minden él egy kommunikációs vonalnak (amit gyakran kapcsolatnak [link] is hívnak) felel meg. Két adott router közötti útvonal kiválasztásához az algoritmus egyszerűen a köztük levő legrövidebb utat keresi meg a gráfban.

A **legrövidebb út (shortest path)** fogalma némi magyarázatra szorul. Egy út hosszát mérhetjük például a megtett ugrások számával. Ezzel a mértékegységgel az 5.6. ábrán szereplő *ABC* és *ABE* út ugyanolyan hosszú. Egy másik mértékegység lehet a földrajzi távolság kilométerekben, amikor is *ABC* nyilvánvalóan sokkal hosszabb, mint *ABE* (feltéve, hogy az ábra léptékhelyes).

Ám az ugrások száma és a földrajzi távolság mellett sok egyéb mértékegység lehetséges. Például mindegyik él súlya lehetne egy szabványos tesztcsomagra vonatkozó átlagos sorbanállási és átviteli késleltetés, amelyet óránkénti próbafuttatásokkal mérnénk. Ezzel az élsúlyozással a legrövidebb út a leggyorsabb út lesz, a legkevesebb kilométerű vagy legkevesebb élből álló helyett.



5.6. ábra. *Az első öt lépés az A-tól D felé vezető legrövidebb út kiszámításában. A nyilak a munkacsomópontot jelzik*

A legáltalánosabb esetben az élsúlyok a távolság, a sávszélesség, az átlagos forgalom, a kommunikációs költség, az átlagos sorhosszúság, a mért késleltetés és más értékek függvényeként számíthatók ki. A súlyozó függvény változtatásakor az algoritmus a „legrövidebb" utat, a számos feltétel közül valamelyik vagy ezek kombinációja szerint számolná ki.

Számos algoritmus ismert egy gráf két csomópontja közti legrövidebb út kiszámítására. Az alábbi Dijkstrától (1959) származik. Minden csomópontot felcímkézünk (zárójelben) a forráscsomóponttól való, legrövidebb ismert út mentén mért távolságával. Kezdetben nem ismertek ilyen utak, ezért minden csomópont címkéje végtelen. Ahogy az algoritmus halad előre és találja meg az utakat, a címkék módosulhatnak, jelezve az egyre jobb utakat. Egy címke lehet ideiglenes vagy állandó. Kezdetben minden címke ideiglenes. Amikor kiderül, hogy a címke a legrövidebb lehetséges utat jelzi, állandóvá tesszük, és ezek után már nem módosítjuk.

A címkéző algoritmus bemutatásához nézzük az 5.6.(a) ábrán szereplő, irányítatlan, súlyozott gráfot, ahol a súlyok például távolságot jeleznek. Az *A*-tól *D*-ig vezető legrövidebb utat akarjuk megtalálni. Azzal kezdjük, hogy az *A* csomópontot állandónak jelöljük meg, amit a kör besötétítésével jelzünk. Ezután sorban végigvizsgáljuk az *A*-val (a munkacsomóponttal) szomszédos csomópontokat, és újracímkézzük őket az *A*-tól való távolságukkal. Amikor ezeket újracímkézzük, akkor azzal a csomóponttal is felcímkézzük őket, amelyiktől a vizsgálatot végeztük, hogy később rekonstruálhassuk a végső utat. Miután *A* minden szomszédját megvizsgáltuk, az egész gráfban levő ideiglenesen felcímkézett csomópontok közül a legkisebb címkével rendelkezőt állandóvá tesszük, amint az az 5.6.(b) ábrán látszik. Ez lesz az új munkacsomópont.

Most *B*-től kezdjük, és az összes szomszédját megvizsgáljuk. Ha *B* címkéjének és a *B* és a vizsgálandó csomópont közti él súlyának összege kisebb, mint a vizsgálandó csomópont címkéje, akkor rövidebb utat találtunk, és a csomópontot újracímkézzük.

Miután a munkacsomópont összes szomszédját megvizsgáltuk, és ha lehetett, az ideiglenes címkéket újra cseréltük, az egész gráfból kikeressük a legkisebb értékű ideiglenes címkével rendelkező csomópontot. Ezt állandóvá tesszük, és ez lesz a következő körben a munkacsomópont. Az 5.6. ábrán az algoritmus első öt lépése látható.

Hogy lássuk, miért működik az algoritmus, nézzük az 5.6.(c) ábrát. Ezen a ponton éppen *E*-t tettük állandóvá. Tegyük fel, hogy volna *ABE*-nél rövidebb út, mondjuk *AXYZE*. Két lehetőség van: vagy a *Z* csomópontot már állandóvá tettük, vagy még nem. Ha igen, akkor az *E*-t már megvizsgáltuk (abban a körben, ami előtt *Z*-t állandóvá tettük), tehát az *AXYZE* út nem kerülte el a figyelmünket.

Most nézzük azt az esetet, amikor *Z* még mindig ideiglenesen van felcímkézve. *Z* címkéje vagy nagyobb, vagy egyenlő *E*-vel, amikor is *AXYZE* nem lehet *ABE*-nél rövidebb út, vagy *E*-énél kisebb, amikor is *Z* és nem *E* lesz előbb állandó, lehetővé téve, hogy *E*-t *Z*-ből vizsgálhassuk.

Ezt az algoritmust az 5.7. ábrán adjuk meg. A program és a fentebb leírt algoritmus között az egyetlen különbség, hogy az 5.7. ábrán a legrövidebb utat a végcsomóponttól, *t*-től számítjuk, a forráscsomópont, *s* helyett. Mivel egy irányítatlan gráfban a *t*-től *s*-ig vezető legrövidebb út ugyanaz, mint az *s*-től *t*-ig vezető, nem számít, melyik végén kezdjük (hacsak nincs több legrövidebb út, amikor is a keresés megfordítása egy

```
#define MAX_NODES 1024              /* a csomópontok számának maximuma */
#define INFINITY 1000000000         /* egy szám, amely nagyobb, mint bármely
                                       leghosszabb út */
n, dist[MAX_NODES][MAX_NODES]  /* dist[i][j] a távolság i-től j-ig */

void shortest_path(int s, int n, int path[])
{ struct state {                    /* a munka tárgyát képező út */
    int predecessor;                /* előző csomópont */
    int length;                     /* a forrástól eddig számolt hossz */
    enum {permanent, tentative} label; /* a címke állapota */
} state[MAX_NODES];

int i, k, min;
struct state *
            p;
for (p = &state[0]; p < &state[n]; p++) { /* állapotinicializálás */
    p->predecessor = -1;
    p->length = INFINITY;
    p->label = tentative;
}
state[t].length = 0; state[t].label = permanent;
k = t;                              /* k a kezdeti munkacsomópont */
do {                                /* Van jobb út k felől? */
    for (i = 0; i < n; i++)         /* ennek a gráfnak n csomópontja van */
      if (dist[k][i] != 0 && state[i].label == tentative) {
        if (state[k].length + dist[k][i] < state[i].length) {
          state[i].predecessor = k;
          state[i].length = state[k].length + dist[k][i];
        }
      }

    /* Keressük meg a legkisebb címkével rendelkező ideiglenesen felcímkézett
    csomópontot! */
    k = 0; min = INFINITY;
    for (i = 0; i <n; i++)
      if (state[i].label == tentative && state[i].lenth < min) {
        min = state[i].length;
        k = i;
      }
    state[k].label = permanent;
  } while (k != s);

  /* Másoljuk az utat a kimeneti tömbbe! */
  i = 0; k = s;
  do {path[i++] = k; k = state[k].predecessor; while (k >= 0);
```

5.7. ábra. *Dijkstra algoritmusa egy gráfon keresztüli legrövidebb út kiszámítására*

másikat derítene fel). A visszafelé keresésnek az az oka, hogy minden csomópontot a megelőző, és nem a következő csomóponttal címkéztünk fel. Amikor a végső utat a kimeneti változóba, a *path*-ba másoljuk, emiatt megfordul az útvonal. A keresés megfordításával a két hatás kioltja egymást, és a választ a helyes sorrendben szolgáltatjuk.

5.2.3. Elárasztás

Egy másik statikus algoritmus az **elárasztás (flooding),** amelyben minden bejövő csomagot minden kimenő vonalon kiküldünk, kivéve azon, amelyiken beérkezett. Az elárasztás nyilván nagyszámú kettőzött csomagot eredményez, valójában végtelen számút, hacsak nem teszünk valamilyen intézkedéseket a folyamat elfojtására. Egy ilyen intézkedés, hogy legyen minden csomag fejrészében egy ugrásszámláló, amely minden ugráskor csökken eggyel, és ha eléri a nullát, a csomagot eldobjuk. Ideális esetben az ugrásszámláló kezdeti értékének a forrástól a célig vezető út hosszát kell beállítani. Ha a küldő nem tudja, milyen hosszú az út, beállíthatja a legrosszabb esetre, nevezetesen az alhálózat teljes átmérőjére.

Az elárasztás gátak közötti tartására egy másik módszer az, hogy tartsuk nyilván, mely csomagokkal végeztük már el az elárasztást, hogy elkerüljük másodszori kiküldésüket. E cél elérésének egyik módja az, hogy a forrásrouter helyezzen el egy sorszámot minden csomagba, amelyet a hosztjaitól kap. Ezek után minden routernek szüksége van forrásrouterenként egy listára, amely megmondja, mely sorszámokat látott már ebből a forrásból. Ha a beérkező csomag rajta van a listán, a router nem küldi tovább szomszédainak.

Hogy meggátolja a lista korlát nélküli növekedését, minden listát ki kell egészíteni egy számlálóval, *k*-val, melynek jelentése: *k*-ig minden sorszám előfordult már. Amikor egy csomag megérkezik, egyszerűen ellenőrizni kell, hogy az másodpéldány-e. Ha igen, el kell dobni. Ráadásul az egész *k* alatti listára nincs szükség, mivel *k* hatékonyan összefoglalja azt.

Az elárasztás egy változata, amely valamivel ésszerűbb, a **szelektív elárasztás (selective flooding).** Ebben az algoritmusban a routerek nem küldenek ki minden bejövő csomagot minden vonalon, csak azokon, amelyek megközelítőleg jó irányba mutatnak. Rendszerint nincs sok értelme egy nyugatra tartó csomagot egy kelet felé vezető vonalon kiküldeni, hacsak nem egészen különleges a topológia.

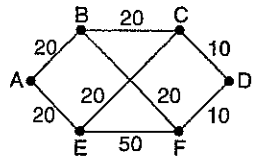
Az elárasztás nem ésszerű a legtöbb alkalmazásban, de van néhány eset, ahol hasznos. Például katonai alkalmazásokban, ahol minden pillanatban nagyszámú router robbanhat darabokra, kifejezetten kívánatos az elárasztás egészséges robusztussága. Elosztott adatbázis-alkalmazásokban néha szükséges az összes adatbázis egyszerre történő frissítése, amikor is az elárasztás hasznos lehet. Az elárasztás harmadik lehetséges felhasználása, hogy más forgalomirányító algoritmusokat hasonlítunk hozzá mint mértékegységhez. Az elárasztás mindig a legrövidebb utat választja, mert az összes lehetséges utat egyszerre választja. Ebből következően nincs olyan algoritmus, amely rövidebb késleltetést tudna produkálni (ha az elárasztási folyamatból adódó többletidőt elhanyagoljuk).

5.2.4. Folyamalapú forgalomirányítás

Az eddig tanulmányozott algoritmusok csak a topológiát veszik figyelembe, a terhelést nem. Ha például mindig nagy mennyiségű forgalom van az 5.6. ábrán *A*-tól *B* felé, lehet, hogy jobb lenne az *A*-tól *C* felé tartó forgalmat az *AGEFC* útvonalra irányítani, annak ellenére, hogy ez az út sokkal hosszabb, mint az *ABC*. Ebben a részben egy statikus algoritmust fogunk tanulmányozni, amely a forgalomirányításhoz a topológiát és a terhelést is felhasználja. Ennek a neve: **folyamalapú forgalomirányítás (flow-based routing).**

Néhány hálózatban a csomópont párok közti átlagos adatfolyam viszonylag állandó és megjósolható. Például egy kiskereskedelmi üzletlánc vállalati hálózatában minden üzlet rendeléseket, eladási jelentéseket, leltárfrissítéseket és más jól meghatározott üzeneteket küldhet ismert helyekre előre meghatározott rendszer szerint, ezért a teljes forgalom nagysága naponta csak keveset ingadozik. Azon feltételek mellett, hogy az *i*-től *j* felé tartó átlagos forgalom előre ismert és reális közelítéssel időben állandó, lehetséges a folyamok matematikai elemzése a forgalomirányítás optimalizálása céljából.

Az elemzés alapötlete az, hogy ha egy adott vonalnak ismerjük a kapacitását és átlagos adatfolyamát, akkor ki lehet számítani az átlagos csomagkésleltetést ezen a vonalon a sorbanállás elmélet segítségével. A minden vonalra ismert átlagos késleltetésből már egyszerű kiszámítani ezek adatfolyamokkal súlyozott átlagát, hogy megkapjuk az egész alhálózat átlagos csomagkésleltetését. A forgalomirányítási probléma ekkor lecsökken arra, hogy az alhálózaton minimális átlagos késleltetést produkáló algoritmust válasszuk.



(a)

Cél

Forrás	A	B	C	D	E	F
A		9 AB	4 ABC	1 ABFD	7 AE	4 AEF
B	9 BA		8 BC	3 BFD	2 BFE	4 BF
C	4 CBA	8 CB		3 CD	3 CE	2 CEF
D	1 DFBA	3 DFB	3 DC		3 DCE	4 DF
E	7 EA	2 EFB	3 EC	3 ECD		5 EF
F	4 FEA	4 FB	2 FEC	4 FD	5 FE	

(b)

5.8. ábra. *(a) Egy alhálózat és a vonalak kapacitásai kb/s-ban. (b) A forgalom csomag/s-ban és a forgalomirányító mátrix*

Hogy ezt a módszert használhassuk, bizonyos információkat előre kell tudnunk. Először is, ismernünk kell az alhálózat topológiáját. Másodszor, az F_{ij} forgalommátrixot meg kell adni. Harmadszor, a C_{ij} vonalkapacitás-mátrixnak, amely minden vonalnak meghatározza a kapacitását b/s-ban, elérhetőnek kell lennie. Végül egy (valószínűleg ideiglenes) forgalomirányító algoritmust kell választani.

Példaként erre az eljárásra, vegyük szemügyre az 5.8.(a) ábrán látható duplex alhálózatot. Az élek súlyai adják meg a C_{ij} kapacitásokat mindkét irányban, kb/s-ban mérve. Az 5.8.(b) ábrán látható táblázatban minden forrás-cél párnak megfelel egy bejegyzés. Az *i* forráshoz és *j* célhoz tartozó bejegyzés megadja az *i–j* forgalom által használandó útvonalat, és az *i* forrásból a *j* cél felé küldendő csomagok számát másodpercenként. Például, *B*-től *D* felé 3 csomag megy másodpercenként, és a *BFD* útvonalat használják. Vegyük észre, hogy már valamilyen forgalomirányítási algoritmust kellett használjunk, hogy a táblázatban szereplő útvonalakat előállítsuk.

Ezen információk birtokában egyszerű kiszámítani a λ_i összeget az *i*. vonalra. Például a *B–D* forgalom 3 csomag/s-mal terheli a *BF* vonalat és szintén 3 csomag/s-mal az *FD* vonalat. Hasonlóan az *A–D* forgalom mindhárom vonalat 1 csomag/s-mal terheli. Mindegyik keletre tartó vonal összforgalmát az 5.9. ábra λ_i oszlopa mutatja. Ebben a példában a forgalom szimmetrikus, vagyis minden *X*-re és *Y*-ra azonos az *XY* és az *YX* forgalom. Valódi hálózatokban ez a feltétel nem mindig teljesül. Az ábra mindegyik vonalra megadja az átlagos csomag/s értéket, μC_i-t is, feltételezve egy $1/\mu = 800$ bites átlagos csomagméretet.

Az 5.9. ábra utolsó előtti oszlopa minden vonalra megadja az átlagos késleltetést, az alábbi sorbanállás elméleti formulából levezetve:

$$T = \frac{1}{\mu C - \lambda}$$

ahol $1/\mu$ az átlagos csomagméret bitekben, C a kapacitás b/s-ban, és λ az átlagos adatfolyam csomag/s-ban. Például $\mu C = 25$ csomag/s-os kapacitással és $\lambda = 14$ csomag/s

i	Vonal	λ_i (csomag/sec)	C_i (kb/s)	μC_i (csomag/sec)	T_i (msec)	Súly
1	AB	14	20	25	91	0,171
2	BC	12	20	25	77	0,146
3	CD	6	10	12,5	154	0,073
4	AE	11	20	25	71	0,134
5	EF	13	50	62,5	20	0,159
6	FD	8	10	12,5	222	0,098
7	BF	10	20	25	67	0,122
8	EC	8	20	25	59	0,098

5.9. ábra. *Az 5.8. ábra alhálózatának analízise 800 bites átlagos csomagméretet használva. A visszairányú forgalom (BA, CB stb.) ugyanaz, mint az előreirányú*

tényleges adatfolyammal az átlagos késleltetés 91 ms. Vegyük észre, hogy $\lambda = 0$-nál az átlagos késleltetés még mindig 40 ms, mivel a kapacitás 25 csomag/s. Más szavakkal, a „késleltetés" magába foglalja a sorbanállási és a kiszolgálási időt is.

Hogy az egész hálózatra kiszámítsuk az átlagos késleltetési időt, vegyük mind a nyolc vonal súlyozott összegét, ahol a súly a teljes forgalom adott vonalra eső része. Ebben a példában az átlag 86 ms-nak adódik.

Hogy egy másik forgalomirányítási algoritmust is kiértékelhessünk, megismételhetjük a teljes folyamatot más adatfolyamokkal, hogy új átlagos késleltetést kapjunk. Ha az egyutas forgalomirányítási algoritmusokra korlátozzuk magunkat, mint ahogy eddig is, akkor csak véges számú lehetőség van a csomagok forrástól célig tartó útvonalának kijelölésére. Mindig lehetséges olyan programot írni, amely egyszerűen mindet egymás után kipróbálja, és megkeresi, hogy melyiknek van a legkisebb átlagos késleltetése. Mivel ez a számítás előre, off-line módon elvégezhető, az időigényessége nem feltétlenül jelent komoly problémát. Ez a legjobb forgalomirányítási algoritmus. Bertsekas és Gallager (1992) részletesen tárgyalják a folyamalapú forgalomirányítást.

5.2.5. Távolságvektor alapú forgalomirányítás

A modern számítógép-hálózatok általában a fentebb leírt statikus algoritmusok helyett dinamikus algoritmusokat használnak. Két algoritmus is igen népszerű, a távolságvektor alapú és a kapcsolatállapot alapú forgalomirányítás. Ebben a szakaszban az előzőt vesszük szemügyre, a következőben az utóbbit fogjuk tanulmányozni.

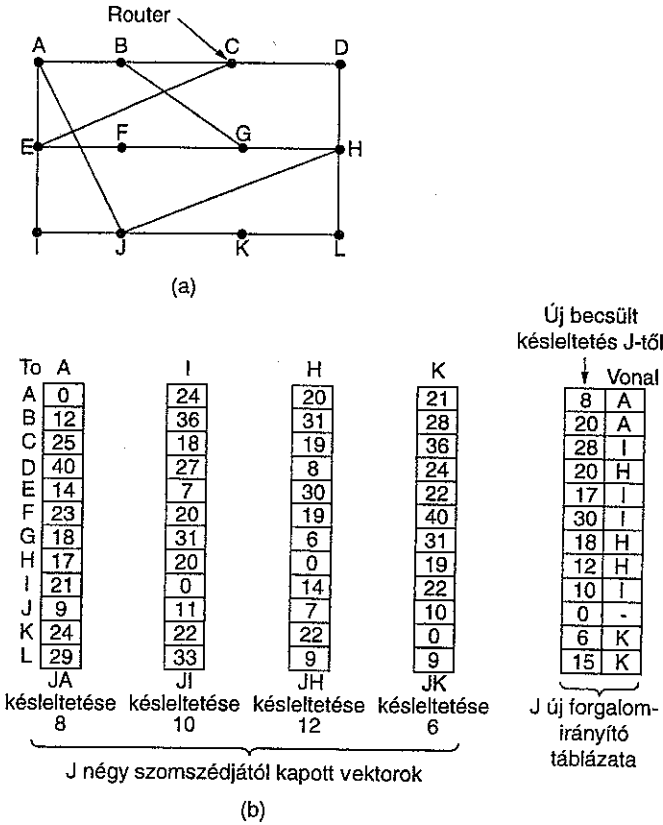
A **távolságvektor alapú forgalomirányítás (distance vector routing)** alapja, hogy minden routernek egy táblázatot (vagyis egy vektort) kell karbantartania, amelyben minden célhoz szerepel a legrövidebb ismert távolság, és annak a vonalnak az azonosítója, amelyiken a célhoz lehet eljutni. A táblázatokat a szomszédokkal való információcsere útján frissítik.

A távolságvektor alapú forgalomirányító algoritmust néha máshogy is nevezik, mint például elosztott **Bellman–Ford** forgalomirányítási algoritmus vagy – az őt kifejlesztő kutatók után (Bellman,1957; Ford és Fulkerson,1962) – **Ford–Fulkerson**-algoritmus, Ez volt az ARPANET eredeti forgalomirányító algoritmusa és ezt használták az Interneten is RIP néven, a DECnet korai verzióiban is és a Novell IPX-ében is. Az AppleTalk és Cisco routerek ugyancsak továbbfejlesztett távolságvektor protokollokat használnak.

A távolságvektor alapú forgalomirányító algoritmusban minden router karbantart egy táblázatot, amelyet az alhálózatban levő összes router szerint indexelnek, és amely mindegyikhez egy bejegyzést tartalmaz. Ez a bejegyzés két részből áll: az adott célhoz előnyben részesített kimeneti vonalból és a becsült időből vagy távolságból. A mértékegység lehet az ugrások száma, időbeni késleltetés milliszekundumokban, az út mentén sorban álló összes csomag száma, vagy valami hasonló.

Feltételezzük, hogy a router tudja a szomszédaitól való „távolságát". Ha a mértékegység az átugrások száma, akkor a távolság csak egy ugrás. Ha a mértékegység a sorhossz, akkor a router egyszerűen megvizsgál minden sort. Ha a mértékegység a késleltetés, a router ezt közvetlenül mérheti különleges *ECHO* csomagokkal, amelyeket a vevő csak időbélyeggel lát el és visszaküld, amilyen gyorsan csak tudja.

Példának okáért tételezzük fel, hogy a mértékegység a késleltetés és a router ismeri az összes szomszédja felé a késleltetést. T ms-onként minden router minden szomszédjának egy listát küld az összes cél felé becsült késleltetéseiről. Minden szomszédjától egy hasonló listát kap. Képzeljük el, hogy épp most érkezett egy táblázat az X szomszédtól, amely tartalmazza X_i-t, vagyis X becslését arról, hogy mennyi időbe telik az i routerig eljutni. Ha a router tudja, hogy a késleltetés X felé m ms, akkor azt is tudja, hogy az i routert X-en keresztül $X_i + m$ ms idő alatt tudja elérni. Ezt a számolást minden szomszédra elvégezve, egy router megtudhatja, melyik becslés tűnik a legjobbnak, és a következőkben ezt a becslést és a hozzá tartozó vonalat használja az új forgalomirányító táblázatban. Vegyük észre, hogy a régi forgalomirányító táblázatot nem használják a számításhoz.



5.10. ábra. *(a) Egy alhálózat. (b) J-hez érkező késleltetési vektorok A, I, H, K felől, és J új forgalomirányító táblázata*

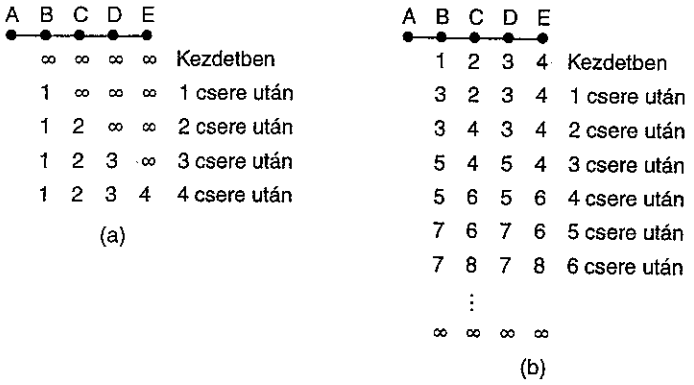
Ezt a frissítési folyamatot illusztrálja az 5.10. ábra. Az (a) rész egy alhálózatot ábrázol. A (b) rész első négy oszlopa a *J* router szomszédjaitól kapott késleltetési vektorokat mutatja. *A* állítása szerint *B* felé 12, *C* felé 25, *D* felé 40 ms-os késleltetéssel rendelkezik.Tegyük fel, hogy *J* a szomszédai, *A*, *I*, *H* és *K* felé a késleltetéseket sorrendben 8, 10, 12 és 6 ms-ra méri vagy becsüli.

Gondoljuk el, hogyan számítja ki *J* a *G* felé vezető új útvonalat. Tudja, hogy el tud jutni *A*-ba 8 ms alatt, és *A* állítása szerint képes 18 ms alatt eljutni *G*-be, tehát *J* tudja, hogy 26 ms-os késleltetéssel kell számolnia, ha a *G* felé tartó csomagokat *A*-nak továbbítja. Hasonlóan, a *G* felé *I*-n, *H*-n és *K*-n keresztül tapasztalható késleltetés sorrendben 41-nek (31 + 10), 18-nak (6 + 12), és 37-nek (31 + 6) adódik. Ezen értékek közül a 18 a legjobb, így bejegyzi a forgalomirányító táblázatába, hogy *G*-ig a késleltetés 18 ms, és a használandó útvonal *H*-n keresztül vezet. Ugyanezt a számolást kell elvégezni az összes többi célra is. Az új forgalomirányító táblázat az ábra utolsó oszlopában látható.

A végtelenig számolás problémája

A távolságvektor alapú forgalomirányítás elméletben működik, de a gyakorlatban van egy komoly hátránya: bár konvergál a helyes válaszhoz, de ezt esetleg lassan teszi. Pontosabban, gyorsan reagál a jó hírekre, de ráérősen a rosszakra. Vegyünk egy routert, amelynek az *X* célig vezető legjobb útja nagy késleltetésű. Ha a legközelebbi cserekor az *A* szomszéd hirtelen egy rövid várakozású utat jelent az *X*-ig, a router egyszerűen átkapcsolja az *X* felé menő forgalmat az *A* felé vezető vonalra. Egy vektorcsere folyamán a jó hír feldolgozásra került.

Hogy lássuk, milyen gyorsan terjed a jó hír, vegyük az 5.11. ábra öt csomópontból álló (lineáris) alhálózatát, ahol a késleltetés mértékegysége az ugrások száma. Tegyük fel, hogy *A* kezdetben nem működik, és ezt az összes többi router tudja. Más szavakkal, mind végtelent írtak be az *A* felé érvényes késleltetéshez.



5.11. ábra. *A végtelenig számolás problémája*

Amikor *A* megjavul, a többi router ezt a vektorcserékből tudja meg. Az egyszerűség kedvéért feltételezzük, hogy létezik egy hatalmas gong valahol, amelynek periodikus ütéseire a routerek egyszerre cserélik ki vektoraikat. Az első csere idején *B* megtudja, hogy bal oldali szomszédjának nulla késleltetése van *A*-ig. *B* most bejegyzi a forgalomirányító táblázatába, hogy *A* egy ugrásnyira van balra. Az összes többi router még mindig azt gondolja, hogy *A* nem működik. Az ekkor érvényes forgalomirányítási táblázatok *A*-ra vonatkozó bejegyzéseit láthatjuk az 5.11.(a) ábra második sorában. A következő cserekor *C* megtudja, hogy *B*-nek van egy 1 hosszú útja *A* felé, tehát frissíti a forgalomirányító táblázatát, hogy az egy 2 hosszú utat jelezzen, de *D* és *E* csak később tudja meg a jó hírt. Világos, hogy a jó hír egy ugrás/csere sebességgel terjed. Egy olyan alhálózatban, ahol a leghosszabb út *N* ugrás hosszú, *N* csere múlva mindenki tudni fog az újjáéledt vonalakról és routerekről.

Most vegyük az 5.11.(b) ábrán látható szituációt, ahol kezdetben minden vonal és router működik. A *B*, *C*, *D* és *E* routerek távolsága *A*-tól sorrendben 1, 2, 3 és 4. Hirtelen *A* elromlik, vagy más esetben, az *A* és *B* közti vonal megszakad, ami *B* felől nézve gyakorlatilag ugyanaz.

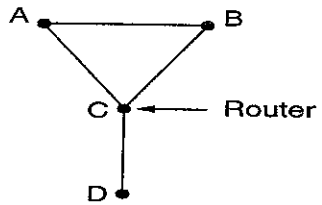
Az első vektorcserénél *B* nem hall semmit *A* felől. Szerencsére *C* azt mondja: „Ne aggódj. Van egy *A*-hoz vezető, 2 hosszú utam." *B* nem tudhatja, hogy *C* útja magán *B*-n vezet keresztül. Amennyit *B* tud, aszerint *C*-nek akár tíz kimenő vonala is lehet független *A*-hoz vezető utakkal, amelyek mind 2 hosszúak. Ennek eredményeként *B* most azt gondolja, hogy elérheti *A*-t *C*-n keresztül, egy 3 hosszú úttal. *D* és *E* nem frissítik az *A*-ra vonatkozó bejegyzéseiket az első cserekor.

A második cserekor *C* észreveszi, hogy mindkét szomszédja azt állítja, hogy az *A*-hoz vezető út 3 hosszú. Kiválaszt közülük egyet véletlenszerűen, és az *A*-hoz tartozó új távolságot 4-re állítja, mint ahogy az 5.11.(b) ábra harmadik sorában látszik. A sorozatos cserék idézik elő az 5.11.(b) ábra maradék részén mutatott történést.

Ebből az ábrából nyilvánvaló, miért terjed lassan a rossz hír: soha nincs egyik routernek sem több mint eggyel magasabb értéke, mint a szomszédainak a minimuma. Fokozatosan mindegyik router felküzdi magát a végtelenig, de az ehhez szükséges cserék száma a végtelen ábrázolásához használt számtól függ. Ennél az oknál fogva okos gondolat a végtelent a leghosszabb út hossza + 1-re állítani. Ha a mértékegység az időbeni késleltetés, nincs ilyen jól definiált felső korlát, tehát egy nagy értékre lesz szükség, hogy a hosszú késleltetésű utakat ne vegyük meghibásodottnak. Nem meglepő módon, ez a probléma a **végtelenig számolás (count-to-infinity) problémája** néven ismert.

A megosztott láthatár hozzátoldása

A végtelenig számolás problémájára sok alkalmi megoldási javaslat található az irodalomban, mindegyik bonyolultabb és kevésbé hasznos, mint az azt megelőző. Ezek közül itt csak egyet mutatunk be, és magyarázzuk majd el, miért vall kudarcot ez is. A **megosztott láthatár (split horizon)** algoritmus ugyanúgy működik, mint a távolságvektor alapú forgalomirányítás, kivéve, hogy az *X*-től való távolságot nem jelentjük azon a vonalon, amelyen az *X*-nek szóló csomagokat elküldjük (pontosabban, végte-



5.12. ábra. *Egy példa, ahol a megosztott láthatár csődöt mond*

lennek jelentjük). Az 5.11.(b) ábra kezdeti állapotában például a *C* a *D*-nek az igazat mondja az *A*-tól való távolságról, de a *B*-nek azt mondja, hogy az *A*-tól való távolsága végtelen. Hasonlóan a *D* igazat mond az *E*-nek, de hazudik a *C*-nek.

Most nézzük meg, mi történik, ha az *A* tönkremegy. Az első cserekor a *B* felfedezi, hogy a közvetlen vonal nem használható, és a *C* is végtelen távolságot jelent az *A*-ig. Mivel egyik szomszédja sem tud eljutni az *A*-hoz, a *B* is végtelenre állítja be a távolságot. A következő cserekor a *C* meghallja, hogy az *A* mindkét szomszédjából elérhetetlen, tehát ő is elérhetetlennek jelöli. A megosztott láthatárt használva, a rossz hír egy ugrás/csere sebességgel terjed. Ez sokkal jobb, mint megosztott láthatár nélkül.

Ám az igazi rossz hír az, hogy a megosztott láthatár, bár elterjedten használják, néha hibázik. Vegyük például az 5.12. ábra négy csomópontból álló alhálózatát. Kezdetben az *A* és a *B* is 2 távolságra van a *D*-től, és a *C* egyre.

Most tegyük fel, hogy a *CD* vonal meghibásodik. A megosztott láthatárt használva, az *A* és a *B* is azt mondja a *C*-nek, hogy nem tud a *D*-be eljutni. Ezért a *C* azonnal levonja a következtetést, hogy a *D* elérhetetlen, és ezt az *A*-nak és a *B*-nek is jelenti. Sajnos az *A* meghallja, hogy a *B*-nek van egy 2 hosszú útja a *D*-be, tehát feltételezi, hogy el tud jutni a *D*-be a *B*-n keresztül 3 ugrás alatt. Hasonlóan, az *A* arra a következtetésre jut, hogy el tud jutni a *D*-be az *A*-n keresztül 3 ugrás alatt. A következő cserekor mindkettő a *D*-től való távolságát 4-re állítja. Mindkettejük fokozatosan a végtelenig számol, és ez pontosan az a viselkedés, amelyet próbáltunk elkerülni.

5.2.6. Kapcsolatállapot alapú forgalomirányítás

Az ARPANET távolságvektor alapú forgalomirányítást használt 1979-ig, amikor is ezt felváltotta a kapcsolatállapot alapú forgalomirányítás. Két fő probléma okozta a bukását. Először is, mivel a távolságmérték a várakozási sorok hossza volt, az útvonal kiválasztásakor nem vette figyelembe a vonalak sávszélességét. Kezdetben az összes vonal 56 kb/s-os volt, így a vonalak sávszélessége nem volt téma, de miután néhány vonalat felfejlesztettek 230 kb/s-ra, másokat pedig 1,544 Mb/s-ra, jelentős probléma lett a sávszélesség figyelembe nem vétele. Természetesen lehetséges lett volna a késleltetés alapú mértéket úgy megváltoztatni, hogy a sávszélességet is beszámítsa, de volt egy második probléma is, nevezetesen, hogy az algoritmus gyakran túl lassan konvergált, még a megosztott láthatárhoz hasonló trükkökkel is. Ezen okok miatt felváltotta egy teljesen új algoritmus, amelyet mostanában **kapcsolatállapot alapú forgalomirányításnak (link state routing)** neveznek. Manapság a kapcsolatállapot alapú forgalomirányítás különböző változatait széles körben használják.

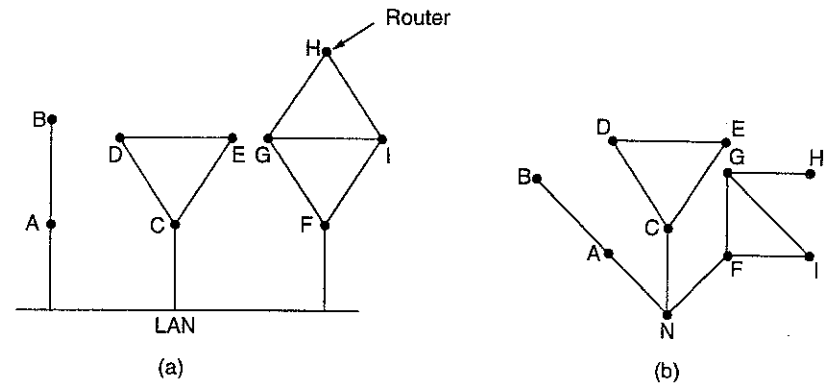
A kapcsolatállapot alapú forgalomirányítás mögötti ötlet egyszerű, és öt részből tehető össze. Minden routernek a következőket kell tennie:

1. Felkutatni a szomszédait és megtudni a hálózati címeiket.
2. Megmérni a késleltetést vagy költséget minden szomszédjáig.
3. Összeállítani egy csomagot, amely a most megtudottakat tartalmazza.
4. Elküldeni ezt a csomagot az összes többi routernek.
5. Kiszámítani az összes többi routerhez vezető legrövidebb utat.

Gyakorlatilag a teljes topológiát és az összes késleltetést kísérletileg megmérik és eljuttatják az összes routernek. Ezután a Dijkstra-algoritmust használhatják, hogy megtalálják a legrövidebb utat az összes többi routerhez. Az alábbiakban mind az öt lépést részletesebben is megvizsgáljuk.

A szomszédok megismerése

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



5.13. ábra. *(a) Kilenc router és egy LAN. (b) Az (a) gráfmodellje*

Amikor két vagy több routert LAN köt össze, a helyzet kicsivel bonyolultabb. Az 5.13.(a) ábra egy LAN-t mutat, amihez három router, *A*, *C* és *F* kapcsolódik közvetlenül. Ezen routerekhez egy vagy több további router kapcsolódik, amint az ábra mutatja.

A LAN modellezésének egy módja, hogy egy csomópontnak fogjuk fel, amint az 5.13.(b) ábra mutatja. Itt egy új, mesterséges csomópontot vezettünk be, *N*-et, amihez *A*, *C* és *F* kapcsolódik. A tényt, hogy lehetséges *A*-tól *C*-ig eljutni a LAN-on, itt az *ANC* út mutatja.

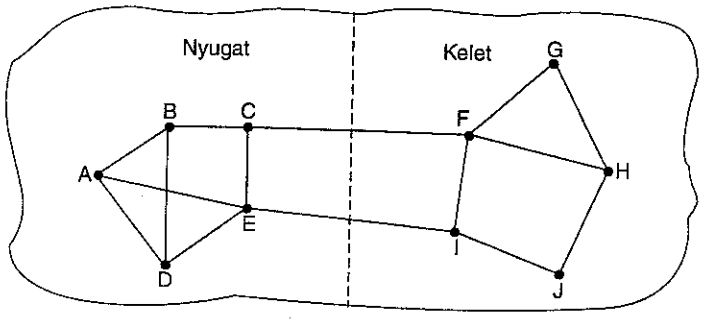
A vonal költségének mérése

A kapcsolatállapot alapú forgalomirányítás megköveteli, hogy minden router tudja, vagy legalábbis reálisan tudja becsülni a szomszédai felé fennálló késleltetést. A késleltetés meghatározásának legközvetlenebb módja egy speciális ECHO csomag kiküldése a vonalon, amelyet a másik oldalnak azonnal vissza kell küldenie. A körbejárási késleltetést megmérve és kettővel elosztva, a küldő router egy reális becslést kaphat a késleltetésről. Még jobb eredményekhez a kísérletet számos alkalommal el lehet végezni, és az átlagot lehet használni.

Egy érdekes kérdés, hogy a terhelést a késleltetés mérésekor figyelembe vegyük-e vagy sem. Ha beleszámoljuk a terhelést, akkor a körbejárási időzítőt akkor kell indítani, amikor az ECHO csomagot a sorba betesszük. Ha figyelmen kívül hagyjuk a terhelést, akkor az időzítőt akkor kell indítani, amikor az ECHO csomag a sor elejére ér.

Mindkét esetet meg lehet indokolni. Ha a forgalom által okozott késleltetéseket is belevesszük a mérésekbe, ez azt jelenti, hogy amikor egy router két azonos sávszélességű vonal közül választhat, és az egyik folyton terhelt, a másik pedig nem, akkor a terheletlen vonalon keresztül vezető utat rövidebbnek fogja fel. Ez a választás jobb teljesítményt fog eredményezni.

Sajnos van egy érv a terhelésnek a késleltetésbe való beszámítása ellen is. Vegyük az 5.14. ábra alhálózatát, amely két részre oszlik, keletire és nyugatira, amelyeket két vonal köt össze, *CF* és *EI*. Tegyük fel, hogy a kelet–nyugat közti forgalom legnagyobb része a *CF* vonalat használja, és ezért a vonal erősen terhelt, és hosszú késleltetéssel bír. Ha bevesszük a sorban állási késleltetést a legrövidebb út számításába, akkor *EI* vonzóbbnak fog tűnni. Miután használatba vesszük az új forgalomirányító táblázatokat, a kelet–nyugat forgalom legnagyobb része *EI*-n keresztül fog haladni, és ezt a vonalat túl fogja terhelni. Következésképpen a következő frissítéskor *CF* fog a legrövidebb útnak tűnni. Ennek eredményeként a forgalomirányító táblázatok vadul ingadozhatnak, amely szeszélyes forgalomirányításhoz és számos lehetséges problémához vezet. Ha a terhelést figyelmen kívül hagyjuk, és csak a sávszélességet nézzük, a probléma nem lép fel. Alternatívaként a terhelést szétoszthatjuk mindkét vonalra, de ez a megoldás nem használja ki teljesen a legrövidebb utat.

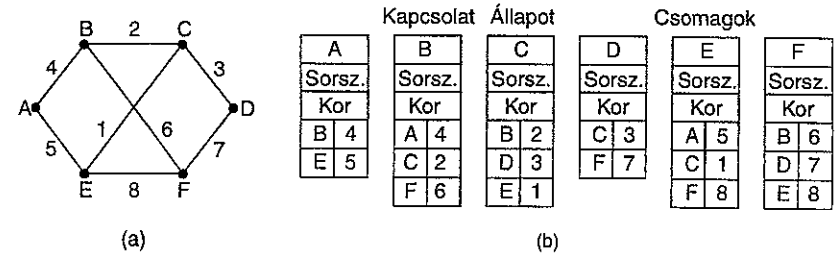


5.14. ábra. *Egy alhálózat, amelyben a keleti és nyugati részeket két vonal köti össze*

A kapcsolatállapot csomagok összeállítása

Ha egyszer már összegyűjtöttük a kicseréléshez szükséges információkat, a következő lépés, hogy minden router összeállítson egy csomagot, amely az összes adatot tartalmazza. A csomag a küldő azonosításával kezdődik, amit egy sorszám és egy korérték (amit később írunk le) követ, majd egy lista a szomszédokról. Minden szomszédhoz a fennálló késleltetést is megadják. Az 5.15.(a) ábrán egy példa alhálózat látható, a vonali késleltetések feltüntetésével. Az 5.15.(b) ábrán mind a hat routerhez láthatóak a megfelelő kapcsolatállapot csomagok.

A kapcsolatállapot csomagok összeállítása egyszerű. Nehéz azt eldönteni, hogy mikor állítsuk őket össze. Egy lehetőség, hogy periodikusan állítsuk össze őket, vagyis szabályos időközönként. Egy másik lehetőség, hogy amikor valami jelentős esemény történt, mint például egy vonal vagy szomszéd meghibásodott, megjavult, vagy megváltoztatta a tulajdonságait.



5.15. ábra. *(a) Egy alhálózat. (b) Ezen alhálózat kapcsolatállapot csomagjai*

A kapcsolatállapot csomagok szétosztása

Az algoritmus legkényesebb része a kapcsolatállapot csomagok megbízható szétosztása. Amint a csomagokat szétosztják és használni kezdik, az elsőket megkapó routerek meg fogják változtatni az útvonalaikat. Következésképpen, különböző routerek eset-

leg a topológia más-más változatait használják, ami inkonzisztenciákhoz, hurkokhoz, elérhetetlen gépekhez és más problémákhoz vezethet.

Először leírjuk az alapvető szétosztó algoritmust, majd később pár finomítást is adunk. Az alapötlet, hogy használjuk az elárasztást a kapcsolatállapot csomagok szétosztására. Hogy az áradást kézben tartsák, minden csomag tartalmaz egy sorszámot, amely minden egyes csomagküldésnél eggyel növekedik. A routerek számon tartanak minden (forrásrouter, sorszám) párt, amit csak látnak. Amikor egy új kapcsolatállapot csomag érkezik, összevetik a már látott csomagok listájával. Ha új, eddig nem látott csomag, akkor továbbítják minden vonalon, kivéve azon, amelyiken érkezett. Ha másodpéldány, eldobják. Ha egy csomag olyan sorszámmal érkezik, amely alacsonyabb, mint a legmagasabb már látott sorszám, akkor mint elavult csomag, nem kerül továbbításra.

Ennek az algoritmusnak van néhány problémája, de ezek kezelhetők. Először is, ha a sorszámok átfordulnak, zűrzavar fog eluralkodni. Itt a megoldás az, hogy 32 bites sorszámokat kell használni. Egy kapcsolatállapot csomagot feltételezve másodpercenként, az átfordulás 137 év múlva következne be, így ezt a lehetőséget figyelmen kívül hagyhatjuk.

Másodszor, ha egy router valamikor összeomlik, elfelejti, melyik sorszámnál tartott. Ha újra 0-ról kezdi, a következő csomagot másodpéldányként felmenti és nem továbbítja.

Harmadszor, ha egy sorszám megsérül, és 65 540 érkezik meg 4 helyett (egy 1 bites hiba), az 5-től 65 540-ig tartó csomagokat elavultként visszautasítja, mivel a jelenlegi sorszámot 65 540-nek hiszi.

Mindezekre a problémákra az a megoldás, hogy minden csomagba a sorszáma után a csomag korát is bele kell tenni, és ezt másodpercenként eggyel csökkenteni kell. Amikor a kor eléri a nullát, az attól a routertől származó információt eldobják. Rendszerint mondjuk 10 percenként érkezik egy új csomag, tehát a router információja csak akkor válik használhatatlanná, ha egy router meghibásodott (vagy ha hat egymás után következő csomag elveszett, de ez valószínűtlen). A kor mezőértékét minden router is csökkenti a kezdeti elárasztás során, így egy csomag sem veszhet el és élhet közelebbről meg nem határozott ideig (egy nulla korú csomag eldobásra kerül).

			Küldési jelzők			ACK jelzők			
Forrás	Sorsz.	Kor	A	C	F	A	C	F	Adat
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

5.16. ábra. *Az 5.15. ábra B routerének csomagpufferje*

A robusztusabbá tételhez néhány finomítás kell. Amikor egy kapcsolatállapot csomag beérkezik egy routerhez elárasztásra, nem kerül rögtön bele az adásra várakozó sorba. Ehelyett egy tárolóterületre kerül, hogy ott egy keveset várakozzon. Ha egy másik kapcsolatállapot csomag is beérkezik ugyanattól a forrástól, az elküldés előtt összehasonlítják a sorszámukat. Ha egyenlőek, a másodpéldányt eldobják. Ha különböznek, a régebbit dobják el. Hogy védekezzenek a router-router vonalakon bekövetkező hibák ellen, minden kapcsolatállapot csomagot nyugtáznak. Amikor egy vonal tétlen lesz, körforgásos alapon végignézik a tárolóterületet, hogy kiválasszanak egy elküldendő csomagot vagy nyugtát.

Az 5.16. ábrán látható a *B* router által az 5.15.(a) ábra alhálózatához használt adatstruktúra. Minden sor megfelel egy nemrég érkezett, de még nem teljesen feldolgozott kapcsolatállapot csomagnak. A táblázat rögzíti, honnan indult a csomag, a sorszámát, a korát, és az adatokat. Ezen kívül *B* mindhárom vonalához (az *A*, *C*, illetve *F* felé vezetőkhöz) vannak küldési és nyugtázási jelzők. A küldési jelzők azt jelentik, hogy a csomagot a jelzett vonalon tovább kell küldeni. A nyugtázási jelzők azt jelentik, hogy ott a csomagot nyugtázni kell.

Az 5.16. ábrán *A* kapcsolatállapot csomagja közvetlenül érkezett, így el kell küldeni *C*-nek és *F*-nek, és nyugtázni *A*-nak, ahogy a jelzőbitek mutatják. Hasonlóan az *F*-től érkezett csomagot továbbítani kell *A*-nak és *C*-nek, és nyugtázni *F*-nek.

A harmadik, *E*-től érkezett csomaggal azonban más a helyzet. Ez kétszer is megjött, egyszer *EAB*-n és egyszer *EFB*-n keresztül. Következésképpen, csak *C* felé kell elküldeni, de *A* és *F* felé is nyugtázni kell, ahogy a bitek mutatják.

Ha egy másodpéldány érkezik, amíg az eredeti még mindig a pufferben van, a biteket át kell állítani. Például, ha egy másolat érkezik *F* felől *C* állapotáról, mielőtt a táblázat negyedik bejegyzését továbbították volna, a hat bitet 100011-re kell változtatni, hogy jelezzük, hogy a csomagot nyugtázni kell *F* felé, de elküldeni nem.

Az új útvonalak számítása

Amint egy router a kapcsolatállapot csomagok egy teljes készletét összegyűjtötte, megszerkesztheti az alhálózat teljes gráfját, mivel minden kapcsolat képviselve van. Valójában minden kapcsolat kétszer szerepel, egyszer-egyszer mindkét irányba. A két érték átlagolható vagy külön használható.

Most helyileg lefuttathatjuk a Dijkstra-algoritmust, hogy megszerkesszük a legrövidebb utat minden lehetséges célhoz. Ennek az algoritmusnak az eredményeit beírhatjuk a forgalomirányító táblázatba, és visszatérhetünk a normális működéshez.

Egy olyan alhálózatban, amely n routerből áll és mindnek k szomszédja van, a bejövő adat tárolásához szükséges memória mérete kn-nel arányos. Nagy alhálózatoknál ez gond lehet. Szintén kérdéses lehet a számítási idő. Mindezek ellenére, sok gyakorlati helyzetben a kapcsolatállapot alapú forgalomirányítás jól működik.

De hardver vagy szoftver problémák nagy pusztítást okozhatnak ezzel az algoritmussal (és másokkal is). Például, ha egy router azt állítja, hogy van olyan vonala, amilyenje valójában nincs, vagy elfelejt egy olyan vonalat, amilyenje viszont van, az alhálózat gráfja helytelen lesz. Ha egy router nem továbbít csomagokat, vagy továbbítás