Hálózatok (11. rész)

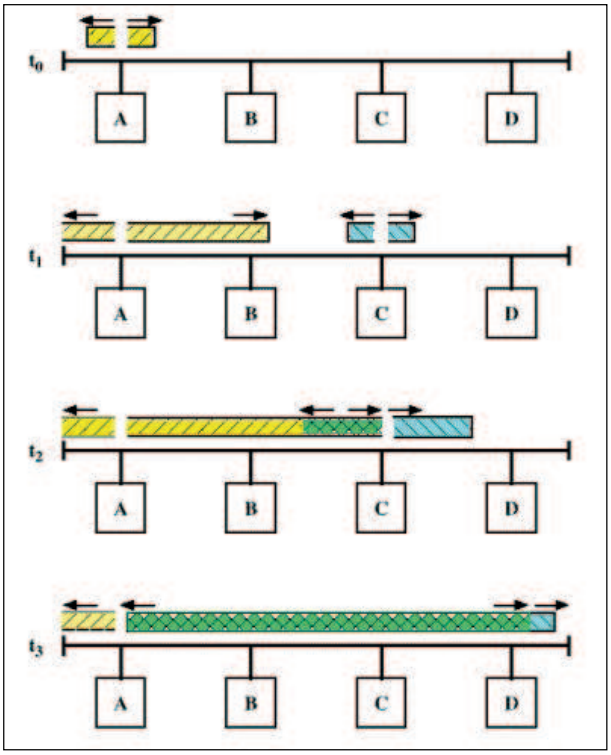
Kettes exponenciális visszalépéses algoritmus, kapcsolók, egyéb szabványok

Legutóbb az Ethernet szabvánnyal foglalkoztam, de adós maradtam néhány dologgal. Ebben a cikkben egyrészt ezeket pótolom, másrészt egy újabb, izgalmas területre vezetem be az olvasót: a kapcsolók világába.

Legutóbb az ***Ethernettel***, illetve az ***IEEE 802.3***-as elnevezésű szabvánnyal foglalkoztunk. Amikor egyszerre több állomás osztozik egy csatornán, akkor a legfontosabb megoldandó probléma a keretek ütközésének elkerülése. Meg kell akadályozni, hogy egyszerre két állomás próbáljon adni. Az előző két részben erre több megoldást is láthattunk, amelyek közül a gyakorlatban az egyik leghatékonyabbnak a ***CSMA/CD*** bizonyult. Segítségével az állomások nem csak a csatornán lévő forgalmat képesek érzékelni (azaz meg tudják állapítani, hogy az adott pillanatban éppen beszél-e valaki), hanem a keretütközést is (vagyis azt, ha egyszerre két állomás forgalmaz). Ha az utóbbi két eset közül valamelyik fennáll, akkor az állomás véletlenszerű ideig várakozik, majd ismét megpróbálkozik a keret elküldésével.

Adós maradtam azzal, hogy miként is érdemes meghatározni azt a véletlenszerű értéket, amennyit az állomásoknak várakozniuk kell. Vessünk egy pillantást az *1. ábrára*, amely már az előző részben is szerepelt. Ezen egy keretütközést örökítettünk meg, ahol az A állomás a ***t0***, a ***C*** pedig a ***t1*** időpillanatban kezdi meg egy keret adását. A csatornán a keretek a ***t3*** időpontban „találkoznak”. Könnyen beláthatjuk, hogy a keret forgalmazásának elkezdése és az ütközés pillanata között ***t3 – t1*** idő telt el. Az A csak akkor érzékeli az ütközést, mikor az ütközés helyétől visszaérkezik hozzá a jel, amely dupla ennyi időt vesz igénybe. Ha ezeket összeadjuk, megkapjuk, hogy az adás megkezdése és az ütközés érzékelése között ***2 * (t3 – t1)*** idő telt el. A ***t3 – t1*** értéke az ütközés helyétől függ, de mivel a ***802.3*** szabványban meg van határozva a hálózat maximális mérete, ezért kiszámolhatjuk, hogy mekkora lehet a ***t3 – t1*** legnagyobb lehetséges értéke. Ez pontosan ***51,2 µs***. Érdemes tehát az ütközés pillanata után az időt diszkrét egységekre felbontani, és ezt az értéket megválasztani időrésnek. A véletlenszám-generátor csak azt döntse el, hogy hány ilyen időrésnyit várakozzunk. Az igazi kérdés az, hogy mekkora legyen az az intervallum, amelyből véletlenszerűen kiválaszthatunk egy értéket.

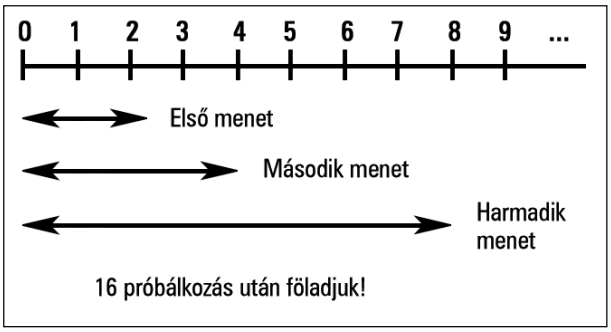
Ha ez az intervallum kicsi, akkor nagy valószínűséggel



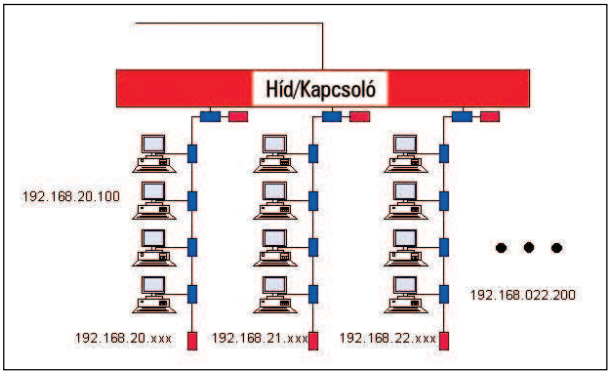
1. ábra Keretütközés

bekövetkezhet egy újabb ütközés (mivel nagyobb az esélye annak, hogy a konkurens állomás is ugyanazt az értéket választotta). Ha csak 0 és 1 lenne a lehetséges választás, akkor mondjuk 50 egyszerre forgalmazni kívánó állomás közül csak akkor adhatna valaki, ha 49-en egyszerre választották a 0-t, és csak ő egyedül az 1-et (vagy fordítva). Nem sokan szeretnének egy olyan hálózatban dolgozni, ahol akár hónapokat is várni kellene egy-egy keret elküldésére.

Az intervallum növelésével azonban az ütközések valószínűsége csökkenthető. Ha például az állomások 0 és 1023



2. ábra Az intervallum mérete minden ütközés alkalmával exponenciális ütemben változik



3. ábra A kapcsolók csak annyit követelnek meg, hogy szabványos keretek érkezzenek a bemeneteire. Így akár egy hub-ot is rákötheténk, és akár több LAN-t is összeköthetünk a segítségükkel. Ilyenkor a kapcsolót hídnak nevezzük.

közötti időrésnyi időt várakozhatnak, akkor a lottót is megszégyenítő eséllyel fordulhatna csak elő egy újboli keretütközés. Itt más probléma merül fel: túl nagy a késleltetés. Szerencsétlen esetben egy állomásnak 1023 időrésnyi időt kéne végigunatkoznia, mikor már 1 időrés elteltével vígan forgalmazhatna. Még átlagos esetben is több száz időrés a késleltetés, ami nagyon nagy hatékonyságbeli romlás. Kell tehát egy olyan algoritmus a várakozás időtartamának meghatározására, amely kis valószínűséggel okoz keretütközéseket, mégsem nő szakállunk, amíg a keret elküldésére várunk. Ilyen módszer például a ***kettes exponenciális visszalépés (binary exponential backoff)***. A működése a következő: az állomás először 0 vagy 1 időrésnyit vár, majd ismét megpróbálkozik az adással. Ha mindkét vetélkedő állomás ugyanazt az értéket választja (amelynek a valószínűsége 0,5), akkor ismét sorsolnak, de most már 0 és 3 közötti értékek közül. Ekkor az ütközés valószínűsége már csak 0,25. Ha azonban még sincs szerencsénk, akkor a választás 0 és 7 közül történik. Láthatjuk, hogy minden ütközés után a sorsolandó értékek száma exponenciálisan nő. Az ***i***-edik ütközés után 0 és 2^i - 1 közötti időrésnyi időt kell várakozni. Fontos azonban, hogy ez az érték nem nő 1023 fölé és a 16. ütközés után a hálózati kártya vezérlője általában hibajelzést küld, mivel valószínűsíthető, hogy valami probléma van a hálózattal *(2. ábra)*.

Ennek az algoritmusnak az az előnye, hogy tekintettel van a konkurens állomások számára. Ezt azzal érjük el, hogy az ütközések hatására az intervallum méretét exponenciálisan növeljük. Ezzel biztosítjuk, hogy egyrészt nagyszámú vetélkedő állomás esetén se kelljen sokáig várnunk a forgalmazásra, másrészt fordított helyzetben (kevés állomás esetén) sem kell számolnunk nagy késleltetéssel.

Kapcsolt LAN-ok

Ahogy nő a hálózatba kapcsolt állomások száma, a csatorna telítődni fog. Ebben az esetben meg kell fontolnunk, hogy beszerzünk egy úgynevezett ***kapcsoló*** vagy ***switch*** nevű mágikus dobozt *(4. ábra)*.

Míg a hub-ok csak arra voltak képesek, hogy a beérkező jelet az összes kimenetre elküldjék, addig a kapcsolók már képesek betekinteni a keretek tartalmába. Ennek köszönhetően az egyik porton beérkező keretet nem küldik tovább az összes többire, csak arra, amelyikre a célpont is csatlakozik.

A kapcsoló lelke egy belső áramkör, amelyre úgynevezett vonali kártyák csatlakoznak, amelynek száma 4 – 32 között változhat. Egy-egy kártyán 1-8 csatlakozó található. Ezektől a csatlakozóktól rendszerint csavart érpár vezet az állomásokhoz.

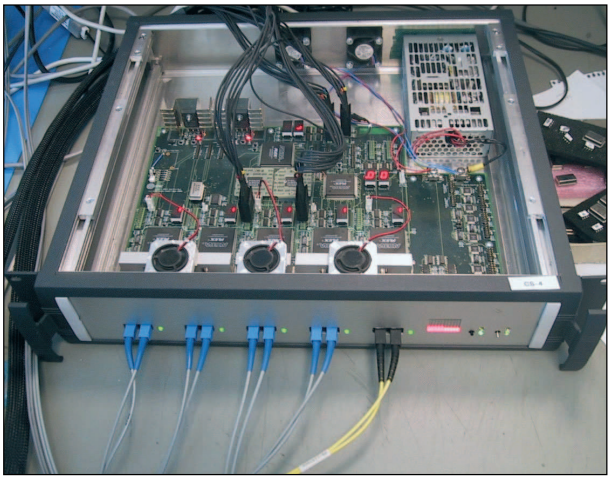
Nézzük, hogyan is zajlik a kommunikáció a kapcsoló segítségével. Tegyük fel, hogy beérkezik egy keret az egyik portra. Ekkor a vonali kártya belenéz a keretbe, hogy vajon ki a címzett. Szerencsés esetben a címzett is ugyanarra a vonali kártyára van kapcsolva, így egyszerűen a keretet csak a megfelelő portra kell másolni. Ha a címzett egy másik kártyán van, akkor előtte át kell oda vinni. Ezt a kapcsoló hátlapján lévő átviteli panel végzi, amely körülbelül 1 Gb/s-os sebességgel dolgozik, és egy belső protokollt használ. A hálózat állomásainak nem kell foglalkozniuk azzal, hogy melyik állomás melyik vonali kártyára kapcsolódik.

Érdekes kérdés, hogy mi történik akkor, amikor két vagy több, ugyanarra a vonali kártyára kapcsolódó állomás egyszerre kezd forgalmazni. A megoldás ilyenkor a vonali kártyák kialakításától függ. Az első megközelítés az, hogy a kártyán kialakított portok egymással össze vannak kötve, így egy közös csatorna alakul ki az ugyanahhoz a kártyához csatlakozó állomások között. Ez azt jelenti, hogy egy adott kártyához kapcsolódó állomások közül mindig csak az egyik adhat, különben ütközés lép fel, ugyanakkor a kapcsolóban lévő többi vonali kártya egymástól függetlenül működhet. Ezt úgy szokás mondani, hogy minden kártya saját ***ütköztetési tartománnyal*** rendelkezik.

A másik (valamivel drágább) megközelítés, hogy nem csak a kártyák, hanem a portok is függetlenek egymástól. Minden port rendelkezik egy saját pufferrel, ahova a beérkező keretek kerülnek. Ezután a kapcsoló belső programja ***(mert ilyen is van neki)*** dönti el, hogy melyik keretet mikor és hova továbbítja. Láthatjuk, hogy itt már a portok is külön ütköztetési tartománnyal rendelkeznek. Ha minden porthoz csak egy állomás csatlakozik, akkor a hálózaton ismeretlen lesz a keretütközés fogalma, és ezáltal nagyságrendekkel gyorsabb átvitelt érhetünk el, mint például egy koaxiális kábelt használó ***10Base2*** hálózaton.

Fontos megjegyeznünk, hogy a portokra nem csak egy állomást csatlakoztathatunk, hanem például egy hubot is, amely több állomást köt össze. Ilyenkor azonban már a kap-

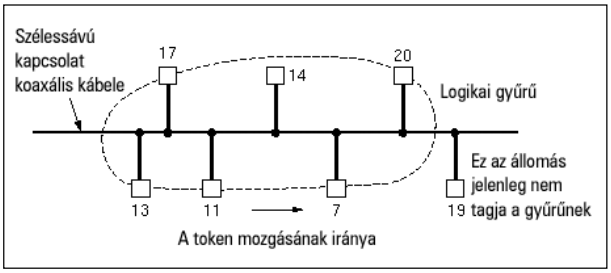
csolót két ugyanolyan szabványú ***(802.3)*** hálózatot összekötő hídnak is nevezhetjük *(3. ábra)*. A hidakkal még ebben a részben részletesen is foglalkozunk.
Az olvasó most valószínűleg arra gondol, hogy a kapcsoló nem csak valami felfejlesztett hub, hanem valójában egy számítógép. És valóban! A kapcsolónak van processzora, memóriája, perifériája ***(például a portok)***, sőt még operációs rendszere is. Alaptalan tehát az a népszerű feltételezés, hogy a kapcsoló valójában egy hub, csak sokkal gyorsabb. Igazából semmi közük egymáshoz. Míg a hub csak a beérkező jeleket ismétli minden irányban, addig a kapcsoló a beérkező kereteket, és csak arrafelé, amerre a keret címzettje van.
Hogy egy kapcsoló valóban hatékonyan működjön, szükség van egy fejlett belső szoftverre. Hogy ez a szoftver miként is végezze a feladatát ***(a keretek továbbítását)***, az nem egy egyszerű kérdés. Vannak kapcsolók, amelyek például a keretnek csak az első 16 bájtját olvassák be, amelyből már kiderül, hogy ki is a címzett. Amint kinyerték ezt az információt, egyből továbbítják is a megfelelő portra. Másfajta megközelítés a ***tárolás és továbbküldés (store and forward)*** módszer, amely az egész keretet beolvassa, és ellenőrzi annak sértetlenségét. Ez ugyan erőforrás igényes feladat, viszont előre kirostálhatóak a hibás keretek, így ezzel is csökken a hálózat terheltsége. Ilyenkor azonban a kapcsolónak meg kell kérnie az adót, hogy ismételje meg az eldobott keretet.
A kapcsolók a hálózatok gerincét képezik, ezért a szoftvernek illik másról is gondoskodnia. Például arról, hogy gonosz emberek ne terhelhessék túl ***(flood)*** a hálózatot. A legegyszerűbb ilyen eljárás az ***üzenetszórási vihar (broadcast storm).*** Ilyenkor a támadó rövid idő alatt rengeteg üzenetszórási (broadcast) csomagot indít útnak. Mivel ezek mindenkihez szólnak, a kapcsoló egy ilyen kerettel szemben úgy viselkedik, mintha egy közönséges hub lenne, vagyis minden portján továbbítja azt. Ezért a jobb kapcsolókban megtalálható a ***Broadcast Storm Control*** nevű szolgáltatás, amely minden porton egy időegység alatt csak meghatározott százalékban enged át ilyen jellegű üzeneteket. (Sajnos ez a védelem is kijátszható úgy, hogy broadcast üzenetek helyett olyan rendes kereteket küldünk, amelyek egy nem létező géphez szólnak és ezt a cél címet folyamatosan változtatjuk).
Persze a kapcsolónak nemcsak a hálózatot, hanem saját magát is védenie kell. Mivel ez is egy számítógép, ezért betelhet a memóriája, túlterhelődhet a processzora, stb., tehát a támadónak maga a kapcsoló is ideális célpontja lehet.
A processzort leterhelni például sok keret rövid idő alatt történő elküldésével lehet. A jobbfajta szoftverek ilyenkor szólnak a forgalmazó hardvernek, hogy lassabban küldjön. Ez megfelelő védelem, mivel csak nagyon régi hardverekkel lehetne kikerülni, amelyek még régebbi szabvány szerint készültek, ahol nem volt lehetőség a forgalom szabályozására. Persze vannak olyan kapcsolók is, amelyek egyszerűen csak lefagynak, esetleg eldobnak minden feldolgozhatatlan csomagot, vagy feladják, és úgy kezdenek működni, mint egy közönséges hub.
Ha már szóba került a kapcsolt hálózatok biztonsága, érdemes eloszlatnunk még egy félreértést, mely szerint a kapcsoló teljes védelmet nyújt a hálózaton folyó kommunikáció lehallgatása ***(sniffelés)*** ellen. Később látni fogjuk, hogy ez mennyire nem igaz. Léteznek olyan technikák, amellyel átejthetjük a kapcsolót, és elhitethetjük vele, hogy mi nem azok vagyunk, akik. Egy kis ügyeskedéssel rávehetjük, hogy a másnak szóló üzeneteket nekünk továbbítsa. Ezek a technikák az úgynevezett ***ARP Cache Poisoning*** nevű eljáráson alapulnak, amely nem jelent mást, mint hogy kilétünkről össze-vissza hazudunk mindenkinek. Annyit elöljáróban elárulunk, hogy az ***ARP*** egy olyan protokoll, amely segít az adatkapcsolati rétegnek megtudni, hogy melyik MAC cím felel meg a kérdéses IP címnek (mivel az adatkapcsolati réteg nem tudja az IP cím segítségével azonosítani az állomásokat). Mivel az ***ARP*** nem követel meg semmiféle hitelesítést, ezért nem lehet ellenőrizni, hogy mindenki becsületesen válaszolt-e. Ezt használják ki a ***Men in the Middle*** elnevezésű támadásoknál is. Később, az IP protokoll ismertetése után egy teljes részen keresztül foglalkozunk majd a támadásokkal, és azok ellenszereivel.



4. ábra Egy kapcsoló belülről...

Más szabványos LAN-ok

Eddig a ***802.3***, illetve az ***Ethernet*** nevű szabvánnyal foglalkoztunk. Ez a világ legelterjedtebben használt hálózattípusa. Méltán tett szert erre a népszerűségre, ugyanis itt az állomásokat kedvünkre csatlakoztathatjuk, illetve távolíthatunk el anélkül, hogy le kéne állítanunk a hálózatot. Másik szimpatikus tulajdonsága, hogy ha nincs leterhelve a hálózat, akkor a keretek küldésének késleltetése nem számottevő.
Az éremnek azonban két oldallal van, így az ***Ethernet*** is rendelkezik hátrányokkal. Az első, hogy az alapvető komponensek analógok. Fel kell fedeznünk a keretütközéseket, ezért nem küldhetünk 64 bájtnál kisebb keretet, hiába akarunk csak 1-1 bájtot átküldeni ***(például terminálok esetén)***.
A legnagyobb hiányosság mégis az, hogy az ***Ethernet*** hálózatok nem determinisztikusak. Ez azt jelenti, hogy gyakran csak a szerencsén múlik, ki mikor forgalmazhat. A dolgok például úgy is alakulhatnak, hogy egy állomás egész hosszú ideig nem képes a keretét útnak indítani. Ilyen viszonyok között szinte lehetetlen bármiféle valós idejű feladatot elvégezni.
Speciális esetekben – például egy gyártósor automatizálásakor – az ***Ethernet*** hálózat tehát nem jelenthet megoldást.



5. ábra A vezérlőjeles sín hálózat

Ezért született meg egy másik, az ***Ethernettel*** megegyező technológiát használó, ám azzal egyáltalán nem kompatibilis szabvány, a 802.4, vagy más néven a ***vezérlőjeles sín (bus token)***.

A legfontosabb eltérés a ***802.3*** és a ***802.4*** között, hogy az utóbbiban kiszámolható egy legrosszabb esetbeli viselkedés. Az ***Ethernet*** esetében nincs felső korlátja annak, hogy egy állomás mikor küldheti el a keretét. Elvileg az is megtörténhet, hogy soha sem kerül erre sor. Egy gyűrű topológia esetében már kiszámolható egy legrosszabb érték. Ha ***N*** darab állomás van, és egy keret elküldéséhez ***T*** idő szükséges, akkor minden állomás legfeljebb ***N * T*** idő múlva útnak indíthatja keretét.

A gyűrű alakú topológiával azonban rengeteg baj van. Egyrészt, ha valahol megsérül, az kihat az egész hálózatra. Másrészt, amikor egy gyártósort elképzelünk, valahogy nem a kör alak jut az embernek eszébe. Ésszerű ötlet volt tehát, hogy a ***802.4*** fizikailag egy ***Ethernet*** hálózatra emlékeztessen, viszont logikailag egy gyűrűt alkosson. Így megmaradt a legrosszabb eset, azaz, hogy N * T idő alatt minden állomás biztosan el tudja küldeni az üzenetét.

A könnyebb megértés végett vessünk egy pillantást az *5. ábrára*. A vezérjeles sín egy koaxiális kábel, amelyre az ***Ethernet*** hálózatok jól ismert módján kapcsolódnak az állomások. Ez a hálózat fizikai elrendezése. Logikailag azonban egy gyűrűt alkotnak: minden gép ismeri a közvetlen jobb, illetve baloldali szomszédjának MAC címét. Amikor a kommunikáció elindul, akkor először a gyűrűben lévő első gép küldhet. Miután elküldte a keretet, egy vezérlőjelet, úgynevezett tokent küld a szomszédjának. A hálózatban mindig csak az küldhet, akinél éppen a vezérlőjel van. Ennek köszönhetően nem léphet fel keretütközés. A vezérlőjel körbejár az egész gyűrű mentén.

Fontos, hogy a gépek gyűrűbeli sorrendje teljesen független a fizikai elhelyezkedéstől. Itt is adatszórásos közeget használunk a kommunikációra, tehát az üzenet mindenkihez el fog jutni. A vezérlőjel pedig nem más, mint egy olyan keret, amelyet az adott állomás a gyűrűbeli közvetlen szomszédjának címez. Erre értelemszerűen csak a címzett fog figyelni. Az állomásokat ezért nem érdekli, hogy fizikailag ki hol helyezkedik el. Sőt, lehet olyan állomásunk is, amely nem részese a logikai gyűrűnek. Ilyenek például a hálózatra frissen felkapcsolt állomások. Ezek nem kerülnek bele automatikusan a gyűrűbe. A logikai gyűrű karbantartása (például állomások felvétele, kitörlése, illetve a sorrend meghatározása) a hálózat közegelérési ***(MAC)*** protokolljának feladata. Ez egy rendkívül összetett protokoll, részletes ismertetése meghaladná e sorozat kereteit.

Meg kell említenünk, hogy természetesen olyan hálózatok is léteznek, amelyek fizikailag is gyűrűhálózatok. Ezek közül a legjelentősebb az 1970-es években, az IBM által kifejlesztett ***vezérlőjeles gyűrű (token ring)***. Valójában ez sokkal szélesebb körben elterjedt LAN, mint az előző, az ***Ethernet*** után a második legnépszerűbb. Az ***IEEE*** is felvette saját szabványai közé, és a ***802.5*** „kódnevet" ragasztotta rá.

Mit is mondhatunk el általánosan ezekről a hálózatokról? Az állomások értelemszerűen egy gyűrűre vannak felfűzve, egymással kétpontos csatornával vannak összekötve. A hálózat szemszögéből a csatorna típusa lényegtelen, lehet akár optikai vagy csavart érpár is. Amikor egy állomás keretet küld, az az egész gyűrűn körbehalad. Mivel a jelek gyorsan terjednek, ezért a keret eleje hamarabb visszaérkezik a feladóhoz, mint ahogy ő befejezné annak küldését. A célállomás a keret utolsó két bitjét változtatja meg: az egyiket akkor, amikor szembesült azzal, hogy a kérdéses keretet neki küldték, a másodikat pedig akkor, amikor sikeresen vette is azt. Amint a keret vége visszaért a feladóhoz, az értesülhet arról, hogy üzenete sikeresen célba ért-e. Ha igen, akkor leveheti a keretet a gyűrűről.

Az állomások itt is csak a megfelelő sorrendben adhatnak, és erről is a vezérlőjel gondoskodik, amely egy speciális keret. Minden állomás csak egy meghatározott ideig birtokolhatja a tokent, utána kötelezően tovább kell adnia. Persze ha éppen nincs mit küldenie, akkor a vezérjelet előbb is átadhatja. Lehetőség van arra is, hogy az állomások között egyfajta prioritási sorrendet állítsunk fel. Ennek a legegyszerűbb módja az, hogy bizonyos állomásoknak megengedjük a token hosszabb ideig történő megtartását.

Amikor nincs forgalom, akkor elég unalmas az élet a gyűrűhálózatokban: csak a vezérlőjel szaladgál körbe-körbe. Ha azonban nagy a terhelés, minden állomásnál már sorban állnak az elküldésre váró keretek, akkor amint elengedi valaki a vezérlő jelet, a következő állomás máris lecsap rá. Ily formán a hálózat kihasználtsága közel van a 100%-hoz.

Persze az élet nem ilyen egyszerű. A legnagyobb problémát egy-egy állomás meghibásodása jelentheti, amely kihathat az egész hálózat működésére. A problémát megoldhatjuk az úgynevezett huzalközpont segítségével. Ilyenkor a fizikai gyűrű megvalósítás helyett az állomásokat két csavart érpár segítségével a huzalközponthoz kapcsoljuk. Az egyik kábelen az állomásoktól érkező adatok haladnak, a másikon pedig azok, amiket mi küldünk feléjük. Amikor meghibásodás történik, a központ rövidre zárja a két kábelt, így a hálózat továbbra is működőképes marad.

Ezt általában úgy szokás megoldani, hogy a reléket, amelyek a rövidre zárásért felelnek, az állomások árammal látják el. Amikor egy állomás meghibásodik, illetve a gyűrű megszakad, akkor a relé nem kap több áramot, és ennek következtében zár. Lehetőség kínálkozik arra is, hogy ezeket a reléket szoftveresen vezéreljük.

A következő részben a hidakkal folytatjuk, amelyek segítségével különböző típusú LAN-okat kapcsolhatunk össze.

Garzó András
garzo@interware.hu