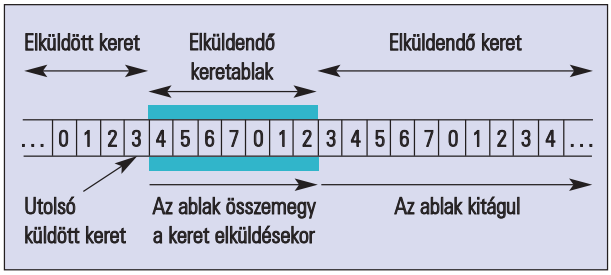
Számítógéphálózatok (9. rész)
Csúszóablakos protokollok, HDLC, SLIP, PPP

Amint azt az előző számban ígértük, ezúttal folytatjuk az adatkapcsolati protokollokkal való ismerkedést, de most már olyanokat is bemutatunk, amelyek a való életben is megállják a helyüket. Például az interneten nagy népszerűségnek örvendő SLIP és PPP.

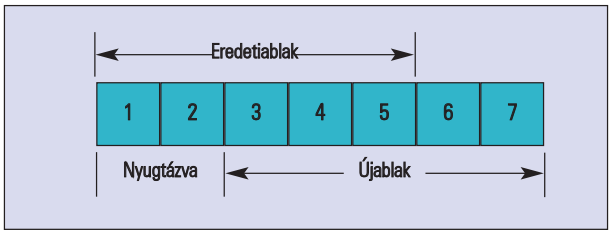
Előző írásunkban elemi adatkapcsolati protokollokkal foglalkoztunk. Sokféle keretátviteli módszert megvizsgáltunk, beleértve a ráültetéses ***(piggybacking)*** módszert, amelynek köszönhetően csökkenthettük a kimenő keretek számát.
Egy csatorna kihasználtságát azonban nem csak úgy növelhetjük, hogy kevesebb keretet küldünk. Az eddig ismeretett összes protokollban az volt a közös, hogy a csatornán egy időben mindig csak egy adatkeret, majd egy azt követő nyugtakeret volt.
Sokkal hatékonyabb lenne, ha a csatornán egyszerre több keret is haladhatna. Az úgynevezett csúszóablakos ***(sliding windows)*** protokollok ezt teszik lehetővé.

Csúszóablakos protokollok

A csúszóablakos protokoll a következő elven működik. Minden elküldött keret tartalmaz egy sorszámot, amely a 0 és valamilyen maximális érték közé esik. A forrás minden elküldött keret sorszámát beteszi egy halmazba. Ez a halmaz az úgynevezett adási ablak ***(sending window)***, amelyhez azok a keretek tartoznak, amelyeket ugyan már elküldtünk, de a vevő még nem nyugtázta őket. Amikor a hálózati rétegtől új csomag érkezik, akkor az adatkapcsolati réteg kiosztja neki a következő sorszámot, majd az ablak felső szélét feljebb csúsztatja.
Vessünk egy pillantást az *1. ábrára*. Itt a kioszthatő sorszámok 0 és 7 közé esnek. Az adási ablakban jelenleg a 4., 5., 6., 7., 0., 1. és 2. keretek vannak. Ezeket már útjukra bocsátottuk és csak a nyugtákra várunk. Amikor egy nyugta megérkezik, akkor az ablak alsó széle eggyel feljebb csúszik, így küldhetünk további kereteket.
A vevő is karbantart egy listát, amelyet vételi ablaknak ***(receiving window)*** nevezünk. Ebben azon keretek sorszámai szerepelnek, amelyeket elfogadhat. Ha egy olyan keret érkezik, amelynek a sorszáma nincs a vételi ablakban, azt azonnal eldobja. Ha az érkezett keret sorszáma mégis benne van az ablakban, a fogadó csak akkor küld nyugtát, ha egyrészt az adott keret még nem került nyugtázásra, másrészt ha minden olyan keretet vettünk, amelyek sorszámai az elsőnek várt és a most érkezett közé esett.



1. ábra Adási ablak



2. ábra Vételi ablak

Szegezzük tekintetünket a *2. ábrára*! Az aktuális ablak első eleme a 3. Ha mondjuk beérkezik az 5. sorszámú keret, csak akkor küldünk róla nyugtát, ha már beérkezett a 3. és az 5. is. Ha pont egy olyan keret érkezik, amelynek sorszáma pont az ablak alsó szélével egyenlő, akkor arról azonnal küldhetünk nyugtát, és az ablakot eggyel elcsúsztathatjuk.
Vegyük észre, hogy a vevőnek nem kell minden keretet nyugtáznia. Visszatérve a *2. ábrára*, ha a forrás megkapja a 6-os kerethez tartozó nyugtát, akkor az egyben a 4-es és 5-ös keretek nyugtázását is jelenti.
Egy másik érdekes dolog, hogy míg az adási ablak mérete folyamatosan változik, addig a vételi ablak mindig ugyanakkora marad. Az adási ablak mérete azonban sohasem mehet a kioszthatő legnagyobb sorszám fölé (ami jelen esetben 7). Ha eléri a maximális méretét, akkor az adatkapcsolati rétegnek „le kell kapcsolnia” a hálózati réteget, azaz nem szabad engedni, hogy az további csomagokat adjon át továbbítás végett.

Mivel a keretek útközben elveszhetnek, a forrásnak az adási ablakba eső kereteket a memóriában kell tartani. Minden kerethez indítania kell egy időzítőt, amelynek lejártáig a nyugtának meg kell érkeznie. Ha ez mégsem történik meg, akkor a keretet ismét el kell küldeni.

Egybites csúszóablakos protokoll

Ez az összes ilyen jellegű protokoll között a legegyszerűbb. Itt minden ablak mérete maximum 1 lehet. A protokoll működése sokban hasonlít az előző részben bemutatott megáll-és-vár protokolléhoz.

Igaz, hogy itt az adatkeretek mindkét irányba mehetnek, de a következő keret csak akkor kerül elküldésre, ha az előzőt a vevő már nyugtázta. A nyugtát azonban hordozhatja egy ellenirányú adatkeret. Mivel egy keret indításának feltétele, hogy az előző nyugtázva legyen, ezért a sorszám értéke csak 0 vagy 1 lehet.

Nézzük, hogyan is működik ez a protokoll.

Legyen a kommunikációban részt vevő két gép neve A és B. Először az A kezdi meg az adást, így elküldi az első adatkeretet B-nek. Az adatkeretbe betesz egy nyugtát, ezzel elhitetve B-vel, hogy az előző keret küldése sikeres volt, így B is elküldi az első keretet (amely egyben A első keretének a nyugtája is). Ezután A veszi B keretét, majd küldi a következőt. De mi történik akkor, ha elveszik egy keret? Például az A elküldte a 0-ás sorszámú keretet B-nek, ami rendben meg is érkezik, a nyugta azonban elvész. Így előbb vagy útóbb lejár A időzítője, és ismét elküldi ugyanazt a keretet. B vételi ablakába azonban már az 1-es keret esik, így a duplikátumot szó nélkül eldobja.

Ezután a B küld A-nak egy keretet, amelynek sorszáma 1, és egyben a 0-s keret nyugtája. Amikor ez megérkezik A-hoz, A elküli a következő keretet. Láthatjuk tehát, hogy hiába veszik el bármelyik keret, a protokoll biztosan nem fog kihagyni egyet sem, és nem fog kettőzött keretet átadni a hálózati rétegnek.

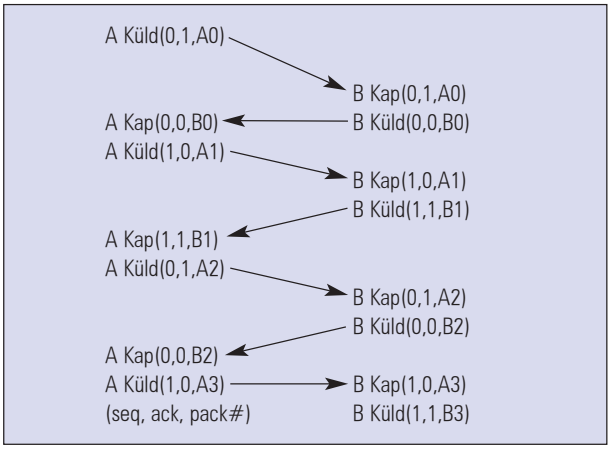
Duplikátumok küldéséhez vezethet azonban az a sajátos eset, amikor A és B az első keretet egyszerre küldik el. Az első keretek ugyanis 1-es nyugtát tartalmaznak a várt 0 helyett, ezért duplikátumok keletkeznek *(3. ábra)*.

Az n-el visszalépő és a szelektív ismétlő protokoll

Az egybites csúszóablakos protokoll csak abban az esetben lehet hatékony, ha a keret és a visszaküldött nyugta célbaérésének együttes ideje elhanyagolható. Ez azonban nem mindig van így. Műholdas átvitel esetén például a keretek átviteli ideje olyan nagy, hogy a fent bemutatott módszerrel a sávszélességnek csupán 4%-át használhatjuk ki (ha 50 kb/sec-os műholdas csatornát használunk).

Ezen úgy tudnánk javítani, ha nem követelnénk meg az adótól, hogy csak akkor küldhet el egy keretet, ha az előzőt a vevő már nyugtázta. Engedjük meg inkább azt, hogy az adó egyszerre több, legfeljebb m darab keretet bocsáthasson a csatornára, majd csak ezután blokkoljon addig, amíg a nyugták meg nem érkeznek.

Ha ezt az m-et ügyesen választjuk meg, akkor az első keret körülfordulási ideje alatt folyamatosan tudunk további kereteket továbbítani anélkül, hogy betelne az adási ablak. (Egy keret körülfordulási idején a keret elküldésétől a nyugta beérkezéséig eltelt időt értjük).



3. ábra Példa az 1 bites csúszóablakos protokoll működésére

Maradjunk az előző példánál, melyben egy 50 kb/sec-es műholdas csatornán szeretnénk kereteket továbbítani. Tegyük fel, hogy egy teljes keret elküldése 20 ms-ig tart, és egy keret körülfordulási ideje 500 ms. Nézzük meg, mi történik ha az előző protokollt használjuk. Ha a kezdő pillanatban elküldjük az első keretet, akkor a 270-edik ms-ban érkezik meg a vevőhöz.

Ha a vevő valamiféle mesebeli masina, amely várakozás nélkül képes a nyugta visszaküldésére, és a nyugta is meglehetősen rövid (legalább annyira, hogy nem kerül időbe a csatornára ráhelyezni), akkor pontosan 520 ms múlva érkezik a forráshoz a nyugta.

Ekkor küldhetjük csak a második keretet, azaz körülbelül félmásodpercenként küldhetünk egyet. (Feltéve ha nem veszik el útközben).

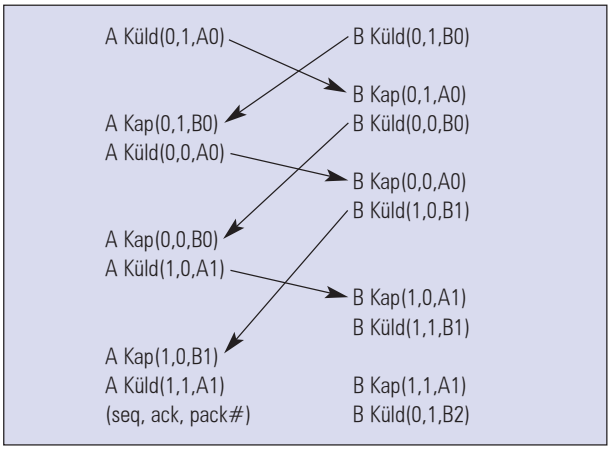
Ha megengedjük, hogy az adó egyszerre több keretet is küldhessen, akkor sokkal jobb eredményt érhetünk el. Ha továbbra is 20 ms-ig tart egy keret elküldése, akkor 520 ms alatt 26 keretet küldhetünk. Mikor a 26. keretet elküldtük, akkor meg is érkezik az első keret nyugtája. Így ha az m értékét 26-nak választjuk, mindig akkor kapjuk meg az engedélyt a következő keret elküldésére, amikor kell.

Mivel mindig 25, illetve 26 nyugtázatlan keret van kint, ha az adási ablak maximális méretét 26-nak szabjuk meg, az sohasem fog betelni (az adó nem fog blokkolni). Így 25-ször nagyobb hatékonyságot értünk el, mint az egybites csúszóablakos protokoll használatával. Ez a techinka csővezetékezés ***(pipelining)*** néven híresült el.

Ez így nagyon meggyőző, de érdemes belegondolni, mi történik akkor, ha a csatorna zajos, és megsérülnek, esetleg el is vesznek egyes keretek.

Az adó csak azután értesül a sérült keretről, miután sok másikat továbbított. Egyértelmű, hogy a vevő egy sérült kerettel nem tud mit kezdeni, de mit csináljon azokkal, amelyek a sérült keret után érkeztek? Ez azért gond, mert az adatkapcsolati réteg a kereteket csak érkezésük sorrendjében adhatja át a hálózati rétegnek.

Erre a problémára két megoldás is létezik. Az első a visszalépés n-el ***(go back n)***, amikor is a vevő eldob minden olyan keretet, amely a sérült keret után érkezett. Mivel nyugtát sem küld vissza róluk, az adó ezeket a kereteket is meg fogja ismételni.



4. ábra 1 bites csúszóablakos protokoll, ha mindkét fél egyszerre kezd el adni. Ilyenkor duplikátumok lépnek fel.

Ha belegondolunk, ez a megoldás nem más, mint egy 1 hosszúságú vételi ablak használata, vagyis a vevő a kereteket kizárólag sorrendben fogadja el. Addig nem foglalkozik a továbbiakkal, amíg az éppen soron következő épségben meg nem érkezik.

Nagyobb átviteli ablak

A másik módszer értelemszerűen nagyobb méretű átviteli ablak használata. Ezt szelektív ismétlésnek ***(selective repeat)*** nevezzük.
Ilyenkor a vevő az összes olyan keretet tárolja a memóriában, amelyek a hibás keret után érkeztek. Az adónak így csak az elveszett vagy megsérült keretet kell megismételnie. Ha az újra elküldött keret már rendben megérkezik, akkor a vevőnek sok-sok hibátlan, ám még nem nyugtázott kerete lesz. Ezeket érkezésük sorrendjében át tudja adni a hálózati rétegnek, majd nyugtázza a legnagyobb sorszámút. Így az adó tudni fogja, hogy az ennél a sorszámnál korábbi keretek hibátlanul célba értek.
Mindkét módszernek megvan a maga hátránya. Az n-el történő visszalépéses stratégiánál ez nyilvánvaló: ha nagyon zajos a csatorna (sok kerethiba lép fel), akkor az a hatékonyságból jelentősen visszavesz.
Az utóbbi módszernél a gond az, hogy minden beérkezett keretet tárolni kell a memóriában, egészen addig, amíg az adott keretnél korábban fogadottak át nem kerülnek a hálózati réteghez.
Hogy a két megoldás közül melyiket célszerű használni, az attól függ, hogy mi a fontosabb számunkra: a sávszélességet a lehető legjobban kihasználni, vagy inkább a memóriával spórolni.

Magas szintű adatkapcsolat-vezérlés

Most bemutatunk egy kicsit öreg, ám ma is elterjedt protokollt, a HDLC-t ***(High-level Data Link Control – magas szintű adatkapcsolat vezérlés)***. Ez egy bit alapú protokoll, amelynek a keretezési eljárását sok más protokoll, például a PPP is átvette (igaz, néhány kisebb különbséggel).
A HDLC-t azokban az időkben fejlesztették ki, amikor a kommunikációban résztvevő két fél általában egy terminál és egy „okos” számítógép volt.
A HDLC (és a hozzá hasonló bitalapú protokollok) az *5. ábrán* látható keretformátumot használják. Minden keretet egy speciális bitsorozat határol, amely a 01111110 (hexadecimálisan 7E). A cím mező olyan kapcsolatoknál érdekes, ahol terminál is csatlakozik a vonalra, és valahogy meg kell határoznunk, hogy az üzenet melyik terminálnak szól.
A vezérlés nevű mezőt sorszámozásra, nyugtázásra és egyéb hasznos dolgokra használjuk, erről majd egy kicsit később. Az adat mező értelemszerűen az átküldendő információt tartalmazza. Ennek nincs meghatározott mérete, csupán egy maximuma. Az ellenőrző összeg a hibakeresésre, illetve a hibajavításra szolgál.
A HDLC olyan csúszóablakos protokoll, amely 3 bites sorszámozást használ. Ez azt jelenti, hogy egyszerre 7 nyugtázatlan keret lehet „kint”.
Háromféle keret létezik: információs, felügyelő és számozatlan. Az 5. ábrán láthatjuk e háromféle keret vezérlés mezejének tartamát. Nézzük először az elsőt! A sorszám mező a keret sorszáma, az utolsó mező pedig a keretre ültetett nyugta. A lekérdezés/utolsó mezőt a HDLC-re épülő protokollok más és más célra használják. Néhol ezzel lehet a másik gépet kényszeríteni arra, hogy azonnal küldje a nyugtát, és ne várjon addig, amíg azt nem tudja ráültetni egy visszafelé menő keretre.
A felügyelőkereteket egymástól a felügyelő kód mező alapján lehet megkülönböztetni. Négyféle felügyelő keret létezik. Az első típus egy olyan nyugta, amelyet nem lehetett ráültetni egy ellenkező irányba menő információs keretre.
A második típus az úgynevezett negatív nyugta, amely arról árulkodik, hogy érkezett ugyan keret, de átviteli hiba miatt sérült volt.
Ilyenkor az adónak újra el kell küldeni az összes keretet a „következő” mezőben lévő sorszámtól kezdődően.
A harmadik típusú felügyelőkeret a „vételre nem kész”. Ez ugyan nyugtázza az összes idáig elküldött keretet, de utasítja az adót, hogy többet ne küldjön. Ez akkor hasznos, ha a vevőnek valamiféle átmeneti problémával kell megküzdenie, például elfogyott a memóriája a további keretek tárolásához.
Az utolsó típus a szelektív elutasítás, amely az adótól csak egy bizonyos keret újraküldését kéri.
Nem szóltunk még a keretek harmadik csoportjától, a számozatlan keretekről. Ezek hordozhatnak adatot és vezérlési információt is. A különbség az előző kettővel szemben az, hogy ezeket a kereteket nem kell nyugtázni.

Az Internet adatkapcsolati rétege

Az Internet önnálló gépek sokaságából áll, amelyeket durván két részre oszthatunk: hostokra és útválasztókra. Ezenkívül az Internethez tartozik még az az infrastruktúra, amely ezeket a gépeket összeköti. Kis távolságok esetén (például egy épület belsejében) általában LAN-okat alkalmaznak. Egymástól messze lévő útválasztók azonban úgynevezett kétpontos összeköttetésben vannak. Ilyen kétpontos összeköttetést létesíthetünk például egy bérelt vonal segítségével.
Kétpontos összeköttetést nem csak útválasztók összekapcsolására használunk. A másik legjellemzőbb felhasználási terület az, amikor az egyéni felhasználók otthoni számítógépüket modem segítségével az internetszolgáltatójuk

(ISP – Internet Service Provider) útválasztójához kapcsolják. Ebben az esetben a felhasználó számítógépe egy „teljes értékű” host lesz, persze csak addig, amíg a kapcsolat él. Legyen szó akár két útválasztóról, amely egy bérelt vonallal van összekapcsolva, vagy egy otthoni PC-ről, amely telefonvonalon keresztül összekapcsolódik az ISP útválasztójával, minden esetben szükség van valamilyen adatkapcsolati protokollra, amely a keretezést végzi. A továbbiakban két az internet világában népszerű adatkapcsolati protokollról lesz szó: a SLIP-ről és a PPP-ről.

SLIP (Serial Line IP – Soros Vonali IP)

Ez egy elég régi protokoll. 1984-ben fejlesztették ki azért, hogy munkaállomásokat tudjanak csatlakoztatni az ARPANET-hez modem segítségével. A SLIP nem is tud semmi egyebet. Nagyon egyszerűen működik. A hálózati rétegtől kapott IP csomagokat szépen sorban átküldi, a keretek végét pedig egy speciális jellel zárja.

Ha az a speciális jel esetleg előfordul a keretben lévő IP csomagban, akkor a régebben bemutatott karakterbeszúrás egyik speciális módszerét alkalmazza: a speciális jelzőbájt helyett egy másik két bájtos sorozatot küld. Egyes SLIP megvalósításokban a keretek is egy speciális bájttal kezdődnek. Az idő előrehaladtával megjelent a SLIP-nek olyan változata is, amely megpróbálja a TCP és az IP csomagok fejlécét tömöríteni. Mivel az ilyen csomagok fejléceinek sok mezője megegyezik (erről majd később lesz részletesen szó), megoldható, hogy a fejlécnek csak egy részét küldjük át.

A SLIP rendkívül népszerű még ma is, sok alkalmazás támogatja, ám sosem volt az Internet szabványos protokollja. Ezért van az, hogy sokféle, egymással nem teljesen kompatíbilis változata létezik. Vannak azonban más súlyosabb problémák, amelyek miatt a SLIP használata erősen behatárolt.

A SLIP Problémái

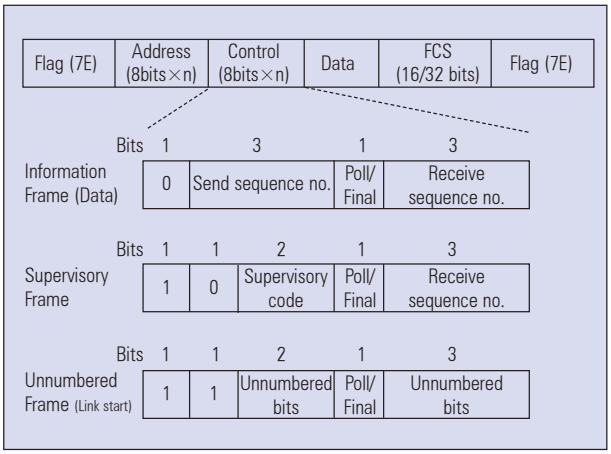
Az első ilyen az, hogy nem tud hibakeresést, illetve hibajavítást végezni. Ezért a felső rétegekre hárul az a hálátlan feladat, hogy ellenőrizzék és kijavítsák a hibákat.

Ezen kívül a SLIP feltételezi, hogy a kommunikációban résztvevő két fél tudja egymás IP címét. Magyarán nincs mód a kapcsolat létrejöttekor az IP cím dinamikus módon történő meghatározására.

Ez nem jó hír az otthonról modemmel csatlakozóknak, hiszen nem kaphat minden ügyfél a szolgáltatójától egyedi IP címet. Ráadásul még a hitelesítésre sincs mód, tehát az egyik fél sosem lehet teljesen biztos abban, hogy valójában kivel is beszél. Az utóbbi két dolog persze két útválasztó bérelt vonalas összeköttetésekor nem jelent gondot. Az viszont már kellemetlen, hogy a SLIP csak az IP alapú hálózatokkal hajlandó együttműködni. Pedig ma már sokféle „színű és szagú” hálózat tartozik az Internethez, és ezek közül nem mind épül az IP-re (például a Novell hálózatok is ilyenek voltak, még az 5-ös változat előtt).

PPP (Point-to-Point Protocol – Pontól – pontig protokoll)

Mivel a SLIP-pel ennyi probléma volt, sokkal kézenfekvőbbnek tűnt egy olyan új protokoll létrehozása, amely kiküszöböli ezeket a hátrányokat, és „méltó” arra, hogy internet-szabvánnyá válhasson. Így született hát a PPP, amely nemcsak képes a hibajelzésre, de többféle hálózatot is támogat, és lehetőséget biztosít a dinamikus IP cím hozzárendelésre, illetve a hitelesítésre is.



5. ábra HDLC protokoll kereteinek felépítése

A PPP legfontosabb feladata a keretezés, vagyis az, hogy a kereteket egymástól egyértelműen elkülönítse és kijavítsa az átviteli hibákból származó sérüléseket. A PPP ezen kívül tartalmaz két alprotokollt is.

Az első az LCP ***(Link Control Protocol – adatkapcsolat vezérlő protokoll)***, amellyel magát a kapcsolatot tudjuk irányítani. Ez alatt olyan dolgokat kell érteni, mint például a kapcsolat bontása, tesztelése, paraméterek beállítása, stb. A másik alprotokoll az NCP ***(Network Control Protocol)***, amellyel a hálózati réteg protokolljának bizonyos beállításait változtathatjuk meg. Az IP esetében ilyen lehet a dinamikus IP-cím hozzárendelése.

PPP a gyakorlatban

Hogy a dolog érthetővé váljon, nézzük meg, mi történik, amikor otthon modemmel csatlakozunk szolgáltatónk útválasztójához. A dolog ott kezd érdekes lenni, amikor a felhasználó és a szolgáltató modeme „összesípolt”, és létrejön a kapcsolat.

Ilyenkor kezdi meg a munkáját a PPP. Először LCP protokoll-csomagok indulnak útnak, amelyek a PPP keretek adatmezejében foglalnak helyet. Az LCP segítségével a kommunikációban résztvevő két fél megállapodik az alkalmazandó PPP paraméterekben.

A második lépés a PC-nk hálózati rétegének beállítása. Több mint valószínű, hogy TCP/IP protokollkészletet szeretnénk majd használni, ezért szükségünk lesz egy IP címre. Mivel a szolgáltatóknak általában kevesebb IP címük van, mint ügyfelük, ezért bejelentkezéskor dinamikusan osztanak ki egyet a felhasználónak. Az aktuális címet az NCP protokoll segítségével fogjuk megkapni.

Nem minden hálózat használ kétpontos összeköttetést. A LAN-ok például adatszóró csatornára épülnek.

A következő részben az ilyen csatornák protokolljaival foglalkozunk.

Garzó András
garzo@interware.hu