Budapesti Műszaki és Gazdaságtudományi Egyetem
Automatizálási és Alkalmazott Informatikai Tanszék

INFORMATIKA 2

SZÁMÍTÓGÉP HÁLÓZATOK

Iváncsy Szabolcs és Vajk István

2007 Október

Tartalomjegyzék

Ábrák jegyzéke

Táblázatok jegyzéke

1. fejezet

Számítógép hálózatok

1.1. Bevezetés

A számítógépek és a távközlés egybeolvadása alapvető befolyással volt a számítógépes rendszerek szervezésére. A hagyományos "számítóközpont" fogalom, mely alatt egy nagyteljesítményű számítógépet tartalmazó helyiséget kell értenünk, ahová az emberek programjukat feldolgoztatni jártak, elavultnak bizonyult, ugyanis minden feladatot egyetlen hatalmas számítógépnek kellett elvégeznie, és a felhasználóknak kellett a számítógéphez menniük. A régi modellt hamarosan teljesen felváltja, a munkát sok, egymástól függetlenül működő, de egymással összekötött számítógép, melyet számítógép-hálózatoknak nevezünk.

Két számítógépet összekapcsoltnak tekintünk, ha azok információcserére képesek. Az összekapcsolás rézhuzallal, lézersugárral, mikrohullámmal, és távközlési műholddal is történhet. Az egy vezérlőegységből és több szolgából álló rendszer nem hálózat! (mester/szolga viszony miatt).
Az elosztott rendszerekben a felhasználó számára az autonóm számítógépek létezése nem látható, tehát a felhasználónak nincs tudomása több processzor létezéséről, neki az egész rendszer egyetlen virtuális processzorként jelenik meg. A munkák processzorokhoz való kiosztása, az állományok lemezekhez való hozzárendelése, azoknak a tárolási és a rendeltetési helyük közötti mozgatása, és minden más rendszerfunkció automatikus. Egy hálózatban a felhasználónak explicit módon be kell jelentkeznie egy gépre, egy távoli programot explicit módon kell elindítania, az állományok továbbítását explicit módon kell meghatároznia, és egyáltalán

minden hálózati tevékenységet közvetlenül kell vezérelnie. Az elosztott rendszerek és a hálózatok közötti különbség nem a hardverben, hanem a szoftverben (operációs rendszerben) keresendő. Közös azonban, hogy mindkettőben szükség van állomány-továbbításra, viszont különbség, hogy az egyikben a rendszer, másikban a felhasználó kezdeményezi a tevékenységet.

1.2. A számítógép-hálózatok használata, a hálózatba kötés céljai

Sok szervezet rendelkezik nagyszámú, egymástól távol eső helyeken működő számítógéppel. Ilyen esetekben érdemes a számítógépeket összekötni annak érdekében, hogy megoszthassák az erőforrásokat. Az erőforrás-megosztás céljai:

- *Erőforrás-megosztás*, azaza hálózatban levő programok, adatok és eszközök - az erőforrások és a felhasználók fizikai helyétől függetlenül - bárki számára elérhetőek legyenek .
- Nagyobb megbízhatóság, amit alternatív erőforrások alkalmazásával érhetünk el.
- *Takarékosság.* A kis számítógépek sokkal jobb ár/teljesítmény aránnyal rendelkeznek, mint a nagyobbak.
- *Skálázhatóság*, tehát annak a biztosítása, hogy a rendszer teljesítményét a terhelés növekedésével oly módon lehessen fokozatosan növelni, hogy újabb processzorokat adunk hozzá.
- Egymástól nagy távolságra levő felhasználók a számítógép-hálózatok segítségével egy hatékony kommunikációs eszköz birtokába jussanak.
- *Nagy információs rendszer.* Az olyan információs rendszerekhez való hozzáférés, mint amilyen a világháló (World Wide Web).

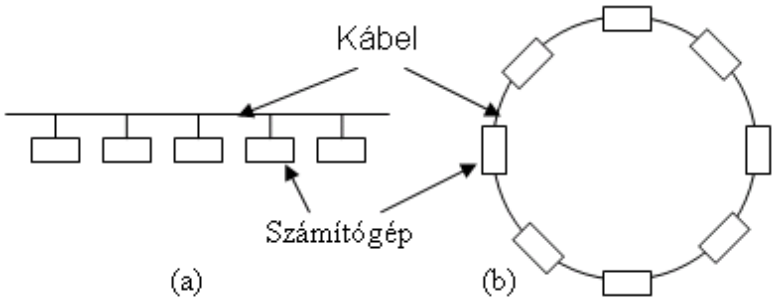
1.2.1. Hálózati alapfogalmak,hálózatok osztályozása

Első megközelítésben megkülönböztethetjük a lokális hálózatokat (Local Area Network, LAN), a nagyvárosi hálózatokat (Metropolitan Area Network, MAN) és a nagy kiterjedésű hálózatokat (Wide Area Network, WAN), ugyanakkor fontos, hogy ezek vezeték nélküliek, esetleg összekapcsolt hálózatok, ezért ezekről és érdemes szót ejteni.

1.2.1.1. Lokális hálózatok (Local Area Network, LAN)

A helyi hálózatokat általában kisméretű helyeken, épületeken belül, illetve egymáshoz közel fekvő épületek közt használják. Előnyei első látásra adódnak: közös erőforrások használata, megosztása, üzenetküldés, file-átvitel, központi adatbázisrendszerek használata, stb.. Nevéből adódóan helyi hálózatokról van szó, így ezek

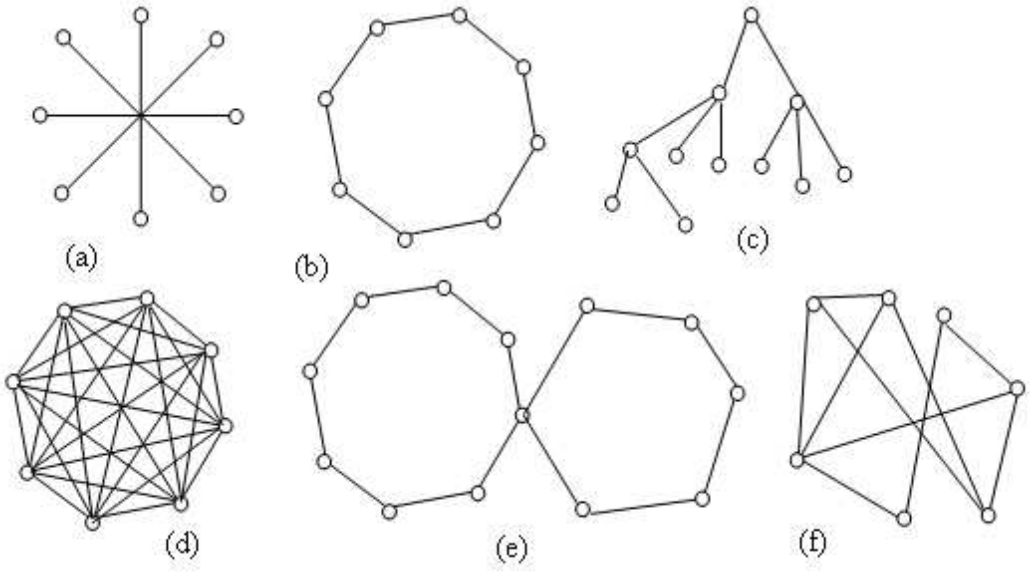
kiterjedése általában kicsi. A LAN-oknak két típusa ismert: a két pont közötti és az adatszóró hálózatok. Az adatszóró hálózatoknak többféle topológiája ismert, a két legelterjedtebb a 1.1. ábrán látható.



1.1. ábra. Két adatszóró hálózat. (a) Sín. (b) Gyűrű

Mindkét átviteli technika egyetlen kábelen alapul, melyre az összes gép rácsatlakozik. Az IEEE 802.3-as szabvány definiálja a legismertebb sín topológiájú hálózatot, az Ethernet-et. Az ilyen hálózatban bármelyik gép bármikor adhat, azonban ha 2 gép egyszerre ad, adatütközés jön létre, és ezeket az adatokat újra kell adni. Az újabb átvitel kezdetéig mindkét gépnek véletlen hosszú ideig kell várakoznia. Az Ethernet sebessége az összeköttetés módja szerint 10 Mb/s vagy 100 Mb/s lehet. Az adatszóró hálózatok másik típusa a gyűrű topológiájú LAN. Működése a következő: Az adó gép sorban egymás után berakja a gyűrűbe a küldeni kívánt biteket, melyek a közeg által meghatározott sebességgel körbe haladnak a gyűrűn, így minden potenciális vevő érzékelheti és feldolgozhatja az adatokat. Ilyen például az IEEE 802.5 által definiált 4 Mb/s és 16 Mb/s-os sebességű LAN; ez az IBM-féle vezérjeles gyűrű.

Pont-pont összeköttetésre a legelterjedtebb topológiák a 1.2. ábrán láthatók. Ezekben az esetekben a két kommunikációs végpontot, pl. egy kábellel kötik össze, és az üzenetek ezen a kábelen keresztül haladnak. Ha a vevő egy olyan csomagot kap, ami nem neki szól, akkor azt továbbadja egy következő pont-pont összeköttetésen keresztül a címzett felé.

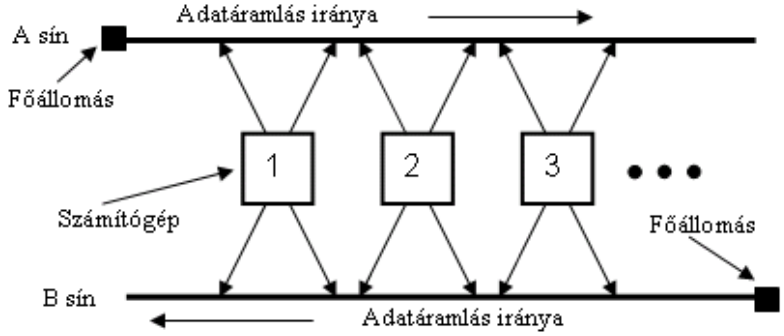


1.2. ábra. Néhány lehetséges két pont közötti alhálózati topológia. a) Csillag. (b) Gyűrű. (c) Fa. (d) Teljesen összekötött. (e) Egymást metsző gyűrűk. (f) Szabálytalan.

1.2.1.2. Nagyvárosi hálózatok (Metropolitan Area Network, MAN)

Általában ugyanazokat a jellemzőket mondhatjuk el róluk, mint a LAN-okról, tekintve, hogy a WAN-ok egy egész várost, vagy több irodaházat, stb. is összeköthetnek, kiterjedésük tehát nagyobb. Azért érdemes róluk beszélni, mert kidolgoztak számukra egy szabványt, melyet mostanában implementálnak. Ez az IEEE 802.6, másnéven DQDB (Distributed Queue Dual Bus). Lényege, hogy 2db egyirányú kábel köti össze a gépeket, és mindkét sín (kábel) rendelkezik egy főállomással, amely az átviteli tevékenységeket kezdeményezi. A küldőtől jobbra eső gépeknek szánt üzenetek a felső sínt, az attól balra lévőknek

szánt üzenetek pedig az alsó sínt használják, ahogy ezt az alábbi ábra mutatja.



1.2.1.3. Nagy kiterjedésű hálózatok (Wide Area Network, WAN)

Már egészen nagy területeket, általában egy országot, vagy egy földrészt fed le. A végrendszereket (end systems, host) egy vagy több kommunikációs alhálózat köti össze, amelyek feladata a hosztok közötti üzenettovábbítás. A legtöbb esetben az alhálózat két különböző komponensből áll: az átviteli vonalakból és a kapcsolóelemekből. Az átviteli vonalak a biteket szállítják a számítógépek között, ezeket más néven trönköknek is hívjuk. A kapcsolóelemek pedig több átviteli vonal összekapcsolására szolgálnak (tipikusan az egyik bejövő vonalon érkező adatokat irányítja át valamelyik kimenő csatornára), ezeket a kapcsolóként működő számítógépeket a továbbiakban routernek illetve, ún. csomóponti gépeknek (Interface Message Processors, IMP) nevezzük.

A nagyobb megbízhatóság érdekében mindegyik gépnek legalább két másik csomóponti géphez kell csatlakoznia, így néhány vezeték, vagy csomóponti gép meghibásodása esetén az üzeneteket automatikusan más útvonalon lehet küldeni. Amikor egy üzenet (csomag) az egyik IMP-től egy másig IMP-ig közbenső IMP-ken keresztül haladva ér el, az érintett IMP-k az üzeneteket teljes egészében megkapják, majd tárolják, amíg a kívánt kimeneti vonal fel nem szabadul, és csak ezután továbbítják azokat. Ezeket az alhálózatokat két pont közötti, tároló és továbbító, vagy csomagkapcsolt alhálózatoknak nevezzük. Megjegyzés: Ezért használatos még a tárol és továbbít (store-and-forward) elnevezés is. A nagy kiterjedésű hálózatok másik jelentős csoportja a műholdas és a földi rádiós rendszerek. Ezekben a rendszerekben minden routernek antennája van, amelyen keresztül adni és venni tud.

1.2.1.4. Vezeték nélküli hálózatok

A notebookok, PDA-k és egyéb hordozható gépek robbanásszerű elterjedésével együtt nő a vezeték nélküli hálózatok száma. Egy vezeték nélküli LAN kiépítése egyszerű, azonban számos korlátba ütközünk a használata során. Sebességük általában 1-2 Mb/s, ami messze elmarad a vezetékes LAN-okétól. A hibaarány szintén magasabb vezetékes társaikénál, ugyanakkor nem szabad elfelejtkezni az egyes gépek egymást zavaró hatásaikról sem. Másrészt bizonyos helyzetekben szóba sem jöhet más összeköttetési mód.

1.2.1.5. Összekapcsolt hálózatok

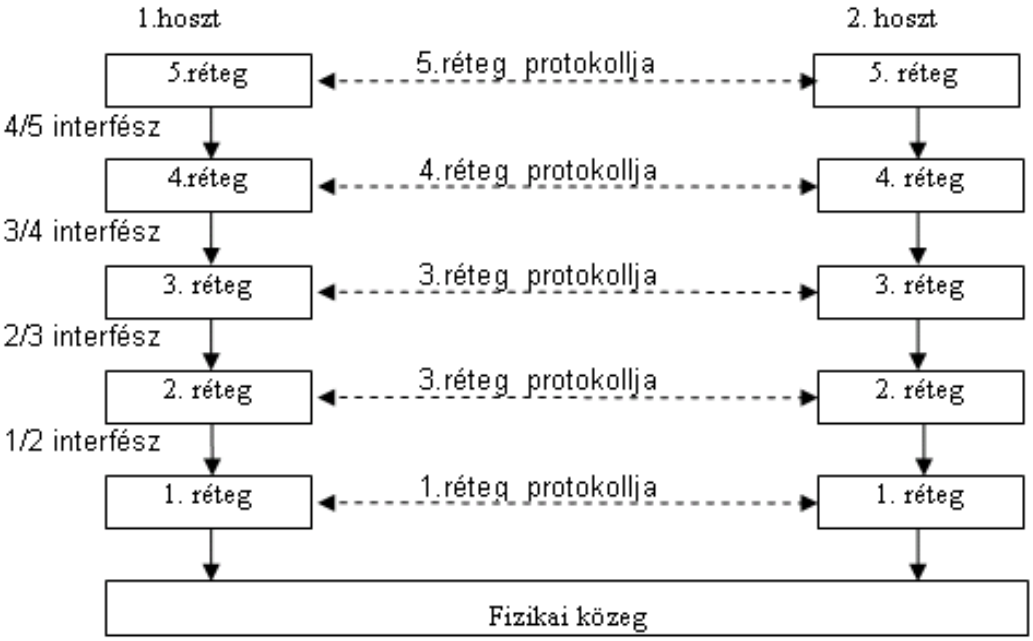
Az egyes hálózatok összekapcsolása egy idő után létfontosságúvá vált. Az adott hálózatok szoftvere, hardvere azonban sok esetben eltér egymástól. A különböző - egymással nem kompatibilis - hálózatok összekapcsolását általában egy átjáró (gateway) oldja meg.

1.2.2. Hálózati architektúrák

A számítógép-hálózatok tervezésének összetettségének csökkentése érdekében az elvégzendő, megoldadndó feladatokat rétegekbe (layer), vagy szintekbe (level) szervezik, amelyek mindegyike az azt megelőzőre épül. Az egyes rétegek célja minden hálózatban az, hogy jól definiált szolgáltatásokat biztosítva, a felsőbb rétegek elől eltakarják a nyújtott szolgáltatások megvalósításának részleteit. Az egyik gépen levő N-edik réteg logikailag a másik gép N-edik rétegével kommunikál. A kommunikáció során használt szabályok és konvenciók összességét protokollnak nevezzük. Az egymással szomszédos rétegek között interfész (interface) található, amely a két réteg közötti elemi műveleteket és szolgálatokat határozza meg.
Fontos, hogy minden réteg egymástól jól elválasztható feladatkörrel rendelkezzen. A kommunikáció során az adó gép N-edik rétegéből az adatok nem közvetlenül jutnak át a vevő gép N-edik rétegébe, hanem mindegyik réteg közvetlenül az alatta lévőnek továbbítja az adatokat egészen addig, míg azok a legalsó rétegig el nem jutnak. Az első réteg alatt a fizikai közeg (physical medium) található, amelyen a valódi, kommunikáció zajlik. A vevő oldalán pedig az alsó rétegek a felettük lé-

vőknek adják át az adatokat. A 1.3. ábrán a virtuális kommunikációt szaggatott vonalakkal, a fizikai kommunikációt pedig folytonos vonalakkal jelöltük.

A rétegek és protokollok halmazát hálózati architektúrának (network architecture) nevezzük. Az architektúra kialakítása során néhány tervezési szempontot figyelembe kell venni. Először is dönteni kell az adatátvitel szabályiról: szimplex (egyirányú), fél-duplex (váltakozóan kétirányú) vagy duplex (egyszerre kétirányú) legyen.



1.3. ábra. Rétegek, protokollok és interfészek

- Szimplex átvitel során az adatok csak egy irányban haladnak, az adó és vevő szerepe fix. Erre példa a TV adás.
- Fél duplex (half-duplex) átvitel esetén az adatok egyszerre szintén csak egy irányba haladhatnak, az adó és vevő szerepe azonban felcserélhető. Ilyen átvitel valósul meg pl. a CB rádiók használata során.
- Duplex átvitel esetén egyidejűleg kétirányú átvitel valósul meg. Pl..: telefon

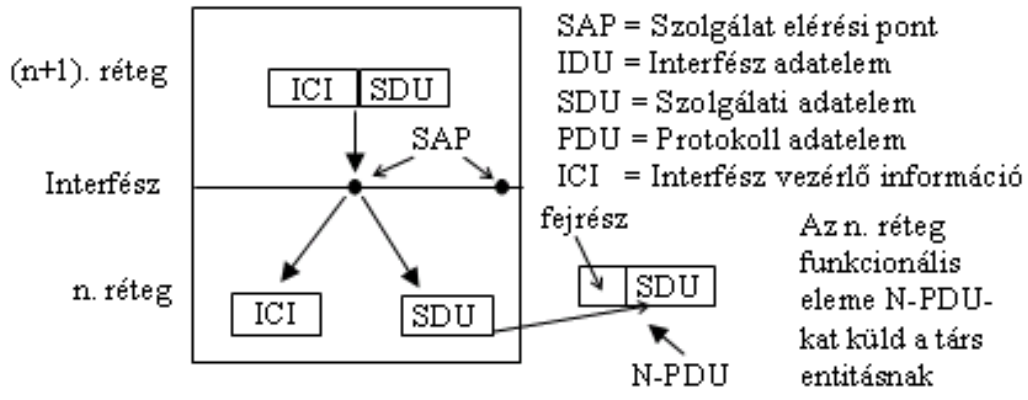
A tervezés további szintjein figyelembe kell venni, hogy

- minden réteg rendelkezzen egy, a kapcsolat felépítését és lebontását biztosító eljárással;
- egy üzenet küldőjének és vevőjének azonosítására szolgáló mechanizmussal;
- legyen benne hibavédelem, hibajelzés;

- a gyors adók és lassú vevők működésének összehangolására szolgáló vezérlési mechanizmus;
- az esetleges maximalizált üzenethossz miatt szétdarabolt üzenetek későbbi helyes összerakását biztosító eljárás;
- több lehetséges útvonal közül az optimális kiválasztására szolgáló algoritmus.

1.2.2.1. Interfészek és szolgálatok

Az egyes rétegekben lévő aktív, működő elemeket ún. entitásoknak (entities), vagy funkcionális elemeknek nevezzük. Ezek lehetnek szoftver-entitások (mint például folyamatok), vagy hardver-entitások (például chipek). Társentitásoknak (peer entities) nevezzük az azonos rétegben, de különböző gépeken található entitásokat. Az N-edik réteg általában az (N+1)-edik rétegnek nyújt szolgáltatást, ekkor az N-edik réteget szolgáltatónak (service provider), míg az (N+1)-edik réteget szolgálat-felhasználónak (service user) nevezzük.
A szolgálatok igénybevételére szolgálnak a szolgálat-elérési pontok (Service Access Points, SAP). Minden szolgálat-elérési pontot egy cím azonosít. Legjobb példa erre a telefonhálózat, ahol a szolgálat-elérési pontok a fali csatlakozók. Ha ismerjük a csatlakozó címét (telefonszámát), és egy készülék is van csatlakoztatva a fali aljzathoz, akkor létrejöhet a kommunikáció. Egy interfészen, a rétegek közti kapcsolatot mutatja a 1.4. ábra. Az interfész adategység (Interface Data Unit,



1.4. ábra. A rétegek közötti kapcsolat egy interfészen

IDU) két részből, a vezérlőinformációból (Interface Control Information, ICI) és

az adatelemből (Service Data Unit, SDU) áll. Az ICI csak az interfész megfelelő működéséhez szükséges, a tényleges információt az SDU hordozza. Az SDU olyan információ, ami a hálózaton keresztül előbb a társentitáshoz jut, majd az (N+1)-edik réteghez kerül. Az SDU-t az N-edik rétegbeli entitás még szétdarabolhatja és fejrésszel ellátva egyenként küldheti tovább. Az így kialakított adategységeket protokoll adategységeknek (Protocol Data Unit, PDU) nevezzük. Ilyen protokoll alapegység például a csomag (packet).

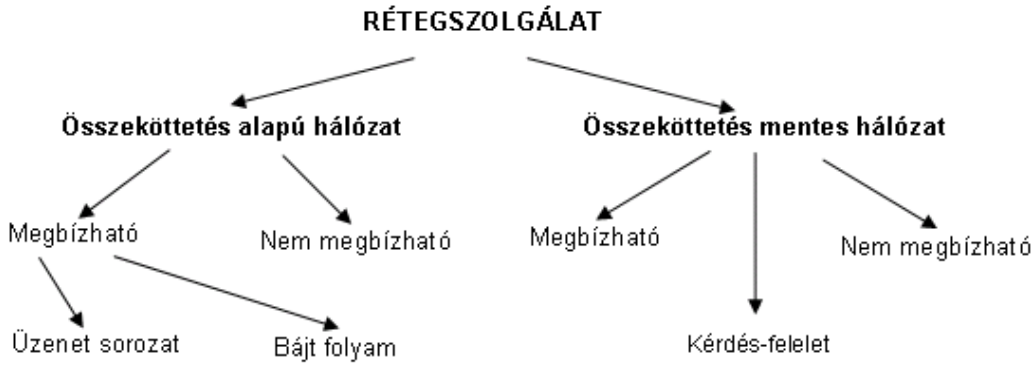
1.2.2.2. Összeköttetés alapú és összeköttetés mentes szolgálatok

A rétegek *összeköttetés alapú* és *összeköttetés mentes* szolgálatot nyújthatnak a felettük lévő rétegek számára.

Az összeköttetés alapú szolgálat (connection-oriented service) során először egy összeköttetést létesítünk az adó és a vevő között, majd elküldjük az adatokat, és amilyen sorrendben küldjük az információt, a vevő ugyanolyan sorrendben kapja meg. Az átvitel befejezése után a kapcsolatot lebontjuk. Mivel az összeköttetés kialakítása időt vesz igénybe, ezért ezt a szolgálatot csak akkor érdemes alkalmazni, ha nagyobb mennyiségű adatot szeretnénk átvinni.
Legjobb példa rá a telefonrendszer: a beszélgetéshez fel kell emelni a kagylót, tárcsázni kell a számot, majd a beszélgetés végén le kell tennünk a telefont. Az információ átvitel sorrendjét tehát szigorúan az adó határozza meg.

Az összeköttetés mentes szolgálat (connectionless service) során az információ az adó és a vevő között a vevő címét is tartalmazó csomagok segítségével kerül átvitelre. A csomagok nem feltétlen az adás sorrendjében érkeznek meg, különböző útvonalon haladhatnak.
Jól modellezi ezt a postai levélkézbesítő szolgálat. A különböző szolgálatokat jellemezhetjük egy szolgálati minőséggel (quality of service). Egy megbízható szolgálat például sosem veszít el adatot. Ezt általában nyugtázási rendszerrel valósítják meg, amikor is az adó minden egyes elküldött csomagról nyugtát vár a vevőtől. Ez plusz idővel és késleltetéssel jár, de sok esetben elengedhetetlen. Ez alapján megkülönböztetünk nyugtázott, azaz megbízható és nem nyugtázott, azaz megbízhatatlan szolgálatokat, mely a távírószolgálathoz hasonló jellege miatt a *nyugtázott datagram szolgálat* (acknowledged datagram service) és a *datagram szolgálat* (datagram service) nevet kapta. A nem nyugtázott átvitel jól használható videokonferenciá-

nál, vagy videó átvitelnél, mert néhány pixel kiesése még nem teszi felismerhetetlenné a képet. Fontos megemlíteni, hogy megbízható átvitel során üzenetsorozat, vagy bájtfolyam útján is érkezhetnek az adatok. Az előbbi esetben az üzenethatárok természetesen megmaradnak, míg egy bájtfolyamnál ezek elvesznek. Egy újabb szolgálat a *kérés-felelet* szolgálat (request-reply service). Általában a kliens-szerver modellben használják, mely során a kliens kér valamit, a szerver pedig válaszol a kérésre. A rétegszolgálatok fenti osztályozását a 1.5. ábra is szemlélteti.



1.5. ábra. A rétegszolgálatok osztályozása

Összeköttetésalapú: Az üzenetek sorrendje nem változik(pl. telefon)
- Megbízható: Nem veszt adatot (nyugtázott)
  - Üzenet sorozat: üzenet határok megmaradnak
  - Bájt folyam: nincsenek üzenethatárok
- Nem megbízható: Nincs nyugtázás (pl. telefon)

Összeköttetés mentes:Az üzenetek sorrendje változhat (pl. levelezés)
- Megbízható: nyugtázott datagramm (pl. tértivevényes levél)
- Nem megbízható: egyszerű datagramm szolgálat (pl. levél)
- Kérdés felelet: (pl. adatbázis lekérdezés)

1.2.2.3. Szolgálat primitívek

A szolgálatot formálisan olyan primitívek (primitives), azaz elemi műveletek (operations) halmazával írhatjuk le, amelyek a szolgálatot elérhetővé teszik a

felhasználó vagy más entitások számára. Ezek a primitívek utasítják a szolgáltatót arra, hogy hajtson végre egy feladatot, vagy számoljon be egy társentitás tevékenységéről. A szolgálat primitíveket négy osztályba soroljuk.

Primitív	Jelentése
Kérés	Egy funkcionális elem kéri a szolgáltatótót egy tevékenység elvégzésére
Bejelentés	A szolgáltató tájékoztatja a túloldali funkcionális elemet a kérésről
Válasz	A funkcionális elem válaszol a kérésre (a szolgáltatónak)
Megerősítés	A szolgáltató közli a kérésre adott választ a funkcionális elemmel.

Ha a kapcsolat létrehozását a CONNECT, az adapt átvitelt a DATA és a kapcsolat lebontását a DISCONNECT szavakkal jelöljük, akkor egy 10 szolgálat primitívből álló összeköttetés alapú szolgálat a 1.1 .táblázat szerint néz ki (a könnyebb érthetőség kedvéért egy telefon híváson keresztül bemutatva az egyes fázisokat).

CONNECT.kérés	A hívó összeköttetés létesítést kér.	tárcsázod a telefonszámot
CONNECT.bejelentés	Jelez a hívott félnek.	kicsöng a telefon
CONNECT.válasz	A hívott a hívást elfogadja/elutasítja.	a hívott fél felveszi a kagylót
CONNECT.megerősítés	Közli a hívóval, hogy a kérést elfogadták-e.	hallod, hogy a csöngés abbamaradt
DATA.kérés	Az adat küldését kéri.	beszélsz
DATA.bejelentés	Az adat érkezését jelzi.	a hívott hallja a beszédet
DATA.kérés	Az adat küldését kéri.	a hívott beszél
DATA.bejelentés	Az adat érkezését jelzi.	hallod amit mond
DISCONNECT.kérés	Az összeköttetés lebontását kéri	leteszed a kagylót
DISCONNECT.bejelentés	Jelzi a kérést a társnak.	hallja, hogy letetted a kagylót

1.1. táblázat. Az összeköttetés alapú szolgálat primitíveinek bemutatása

Ebben a példában a CONNECT egy megerősített szolgálat (amelyre kifejezetten válaszolni kell), míg a DISCONNECT egy megerősítetlen szolgálat (nincs szükség válaszra).

1.2.3. Hivatkozási modellek

A különböző rendszerekben az egyes szintek és szolgáltatások különbözőek lehetnek. Ezeket a megoldási módokat hivatkozási modelleknek nevezzük. Ilyenek pl. az OSI, a TCP/IP stb.

1.2.3.1. Az OSI hivatkozási modell

Ez a modell, a Nemzetközi Szabványügyi Szervezet (International Standards Organization - ISO) által kidolgozott ajánláson alapul, amely a protokollok nemzetközi szabványosítása felé tett első lépésként értékelhető.

A modellt ISO OSI (Open System Interconnection - nyílt rendszerek összekapcsolása) hivatkozási modellnek nevezzük, mely hét rétegből áll. Az alapelvek, amelyek végül is elvezettek a hét réteg kialakulásáig, a következők voltak:

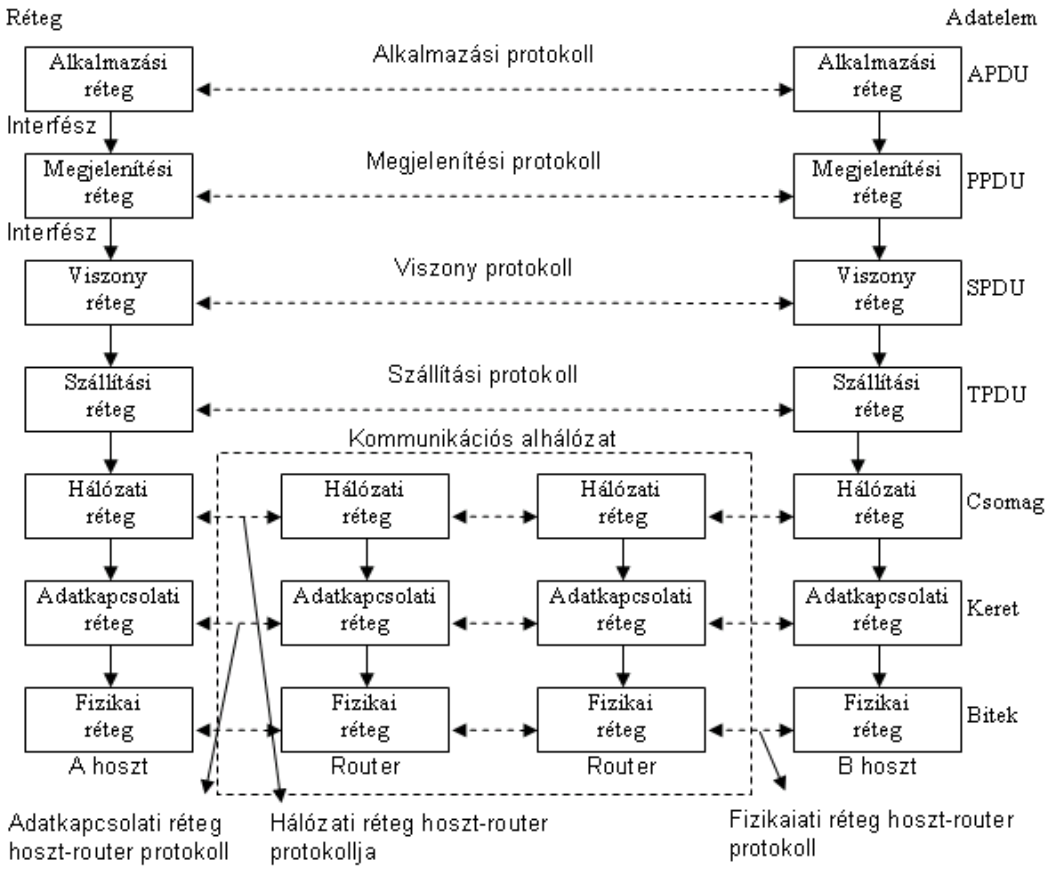
- A rétegek különböző absztrakciós szinteket képviseljenek.
- Minden réteg jól definiált feladatot hajtson végre.
- A rétegek feladatának megválasztásakor nemzetközi szabványok kialakítására kell törekedni.
- A réteghatárok megválasztásakor a rétegek közti információcsere minimalizálására kell törekedni.
- A rétegek számának megfelelően nagynak kell lennie ahhoz, hogy különböző feladatok ne kerüljenek szükségtelenül egy rétegbe, ugyanakkor elég kicsinek kell lennie ahhoz, hogy a szerkezet ne váljon nehezen kezelhetővé.

Maga az OSI modell nem egy hálózati architektúra, hiszen nem határoz meg konkrét protokollokat és szolgálatokat az egyes rétegekben. Csakis azt mondja meg, hogy az egyes rétegeknek mit kellene tenniük. OSI modellen alapuló hálózati architektúrát az 1.6. ábrán láthatunk. A továbbiakban a modell egyes rétegeit fogjuk egyenként bemutatni úgy, hogy a legalsóval kezdjük.

1.2.3.1.1. A fizikai réteg(physical layer).

Ezen a rétegen zajlik a tényleges fizikai kommunikáció. Biztosítania kell azt, hogy az egyik oldalon elküldött 1-es bit a másik oldalon is 1-ként érkezzen meg, és ne pedig 0-ként.

Az információ megjelenési formája változó lehet, pl.: elektromos vezeték esetén, a



1.6. ábra. Az OSI model

vezetéken eső feszültség értéke, vagy éppen a feszültség változásának iránya. Az információt hordozó közeg is sokféle lehet: fénykábel, rádióhullám, stb.

Ez a réteg tipikusan olyan kérdésekkel foglalkozik, pl. hogy milyen feszültség szinteket kell használni a logika értékek átviteléhez, mennyi ideig tart egy bit továbbítása, miként épül föl és bomlik le a kapcsolat, milyen legyen az alkalmazott csatlakozó fizikai, mechanikai kialakítása, stb. A tervezési szempontok itt főleg az interfész mechanikai, elektromos és eljárási kérdéseire, valamint a fizikai réteg alatt elhelyezkedő fizikai átviteli közegre vonatkoznak. A fizikai réteget a későbbiekben részletesen tárgyaljuk.

1.2.3.1.2. Az adatkapcsolati réteg (data link layer).

Alapvető feladata az, hogy egy tetszőlegesen kezdetleges adatátviteli eszközt olyan

adatátviteli vonallá transzformáljon, amely a hálózati réteg számára átviteli hibától mentesnek tűnik.
Ez úgy teljesül, hogy a küldő fél a bemenő adatokat adatkeretekké tördeli (tipikusan néhány száz byte hosszúságú), ellátja kiegészítő cím, egyéb és ellenőrző információval, a kereteket sorrendhelyesen továbbítja, végül a vevő által visszaküldött nyugtakereteket feldolgozza.

Szintén erre a rétegre hárul az adatkeretek határainak létrehozása és felismerése, amely speciális bitmintáknak a keret elé, ill. mögé, illesztésével valósul meg. Ha ezek a bitminták az adatok között is előfordulhatnak, akkor a kétértelműség elkerülésére védelmi eljárásokat kell kidolgozni. Amennyiben egy keret megsérül (tönkremegy), akkor a forrásgép adatkapcsolati rétegszoftverének újra kell adnia a keretet. Ugyanannak a keretnek többszöri újraadása azonban kettőzött keretek megjelenését eredményezheti. Ilyen helyzet pl. akkor alakulhat ki, amikor a vevő által az adónak küldött nyugtakeret elvész. E réteg feladata az ilyen megsérült, elveszett vagy kettőzött keretek miatt felmerülő problémák megoldása.

Egy másik, az adatkapcsolati rétegben (de a legtöbb magasabb szintű rétegben is) előforduló probléma az, hogy a gyors adók adatelárasztással fenyegetik a lassú vevőket. Valamilyen forgalomszabályozási mechanizmust kell beépíteni annak érdekében, hogy az adók tudhassák, hogy a vevők egy adott pillanatban mekkora szabad puffer területtel rendelkeznek (gyakran a forgalomszabályzást és a hibavédelmet együtt valósítják meg).

Ha a vonal kétirányú átvitelre is használható, akkor a gond az, hogy a B-A irányú nyugtakeret-forgalom az A-B irányú adatkeret-forgalommal a vonal használati jogáért versenyez. Ennek a bonyodalomnak a kezelése szintén az adatkapcsolati réteghez tartozik.

Az adatkapcsolati réteg speciális alrétege, a közeghozzáférési alréteg pedig az osztott csatornához való hozzáférést szabályozza.

1.2.3.1.3. A hálózati réteg (network layer).

A kommunikációs alhálózatok működését vezérli. Legfontosabb feladata a csomagok útvonalának megválasztása a forrástól a célállomásig.
Az útvonalválasztás lehet statikus, amikor táblázatokat "behuzaloznak" a háló-

zatba, ugyanakkor dönthetnek róla minden egyes párbeszéd előtt, vagy éppen lehet dinamikus, mikor az útvonal az aktuális leterheltség figyelembevételével kerül kiválasztásra.

Az egyes csomagtorlódásos helyzetek kivédése, és az eltérő protokollokkal rendelkező hálózatok (heterogén hálózatok) összekapcsolása szintén e réteg feladata. Gyakran számlázási eljárást is beépítenek a hálózati rétegbe. Üzenetszórásos hálózatokban az útvonal-választási mechanizmus igen egyszerű, így a hálózati réteg általában vékony, sokszor nem is létezik.

1.2.3.1.4. Szállítási réteg (transport layer).

Feladata a hosztok közötti átvitel megvalósítása. A viszonyrétegtől kapott adatokat szükség esetén kisebb darabokra vágja, majd átadja a hálózati rétegnek. Általában a viszony réteg által igényelt összes szállítási összeköttetéshez egy külön hálózati összeköttetést hoz létre. Ugyanakkor, ha egy összeköttetés nagy átbocsátóképességet igényel, akkor a szállítási réteg létrehozhat több hálózati összeköttetést is, amelyek közt az áteresztőképesség növelése érdekében szétosztja az adatokat. Ha a nagyszámú összeköttetés túl költséges, vagy nem megvalósítható, akkor több szállítási összeköttetést is összenyalábolhat (multiplexelhet) egyetlen hálózati összeköttetéssé.

Ezeket a műveleteket azonban a viszony réteg számára átlátszóan kell végrehajtania. A felhasználóknak nyújtott szolgálatokat szintén a szállítási réteg határozza meg. A szállítási összeköttetés lehet két pont közötti, melynél az elküldött üzenetek sorrendje nem változik meg, ugyanakkor létezik olyan összeköttetés is, mely a küldéskor fennálló sorrendet nem garantálja a vevőnek.

Az információ áramlásának szabályozására is létezik egy mechanizmus, ami szintén ehhez a réteghez kötődik. A forgalomszabályozás (flow control) arra ügyel, hogy a gyorsabb hosztok ne árasszák el a lassabbakat.

1.2.3.1.5. Viszony réteg (session layer).

Feladata a felhasználók között viszony létesítése. A szállítási réteghez képest plusz szolgáltatásokat nyújt, ilyen például a fájlmozgatás két gép között, egy távoli

rendszerbe való belépés, vagy a párbeszéd megszervezése.
E réteg feladata a vezérjelkezelés (token management) is, ami akkor hasznos, ha a két oldal egyszerre ugyanazt a műveletet akarja végrehajtani. Az ilyen eseteket oldja meg a viszony réteg vezérjelekkel. A kritikus műveletet így az végezheti el, akinél a vezérjel van.
A szinkronizálás (synchronization) szintén az együttműködési réteg feladata. Erre akkor van szükség, ha egy átvitel megszakad, és nem akarjuk, vagy épp nem tudjuk elölről kezdeni, mert tisztában vagyunk vele, hogy ismét megszakadna. Ekkor célszerű ellenőrzési pontokat beszúrni az adatfolyamba, ezzel lehetővé válik, hogy csak az utolsó ellenőrzési ponttól keljen újraadni az adatokat a kapcsolat megszakadása esetén.

1.2.3.1.6. Megjelenítési réteg (presentation layer).

Az alsó rétegektől eltérően, amelyek csak a bitek megbízható ide-oda mozgatásával foglalkoznak, a megjelenítési réteg az átviendő információ szintaktikájával és szemantikájával foglalkozik.

Az adatok szabványos kódolása tipikus példája a megjelenítési réteg által nyújtott szolgálatoknak. A legtöbb felhasználói program nem véletlenül előállított, bináris bitfüzéreket küld egymásnak, hanem neveket, dátumokat, pénzösszegeket, számlákon szereplő adatokat stb. Ezeket a tételeket karakterfüzérként, egész és lebegőpontos számokként, és kisebb egységekből álló bonyolult adatstruktúrákként ábrázolják. A különböző számítógépek különböző kódokat használnak a karakterfüzérek (pl. ASCII és EBCDIC), az egész számok (egyes komplemens és kettes komplemens) stb. ábrázolására. Azért, hogy a különböző ábrázolásmódú számítógépek is kommunikálni tudjanak, a kicserélendő adatstruktúrákat egy, a "vonalon" használandó szabványos kódolással absztrakt módon kell definiálni. Ezeknek az absztrakt adatstruktúráknak a kezelését, valamint a számítógépek egyedi adatábrázolásának egymásba konvertálását is a megjelenítési rétegnek kell elvégeznie.

A megjelenítési réteg az információábrázoláson kívül még magába foglalja pl. az adatátvitel hatékonyabbá tételét elősegítő adattömörítést, és a hitelesítést és titkosítást lehetővé tevő kriptográfiát.

1.2.3.1.7. Alkalmazási réteg (application layer).

Ez a réteg kapcsolódik a legszorosabban a felhasználóhoz, így itt kell megvalósítani a hálózati felhasználói kapcsolatok megoldását.

A világon felhasznált több száz egymással nem kompatibilis terminháltípus egységes kezelhetőségére létre kell hozni egy virtuális hálózati terminált (network virtual terminal), és minden programot úgy írunk meg, hogy ezen képes legyen dolgozni.

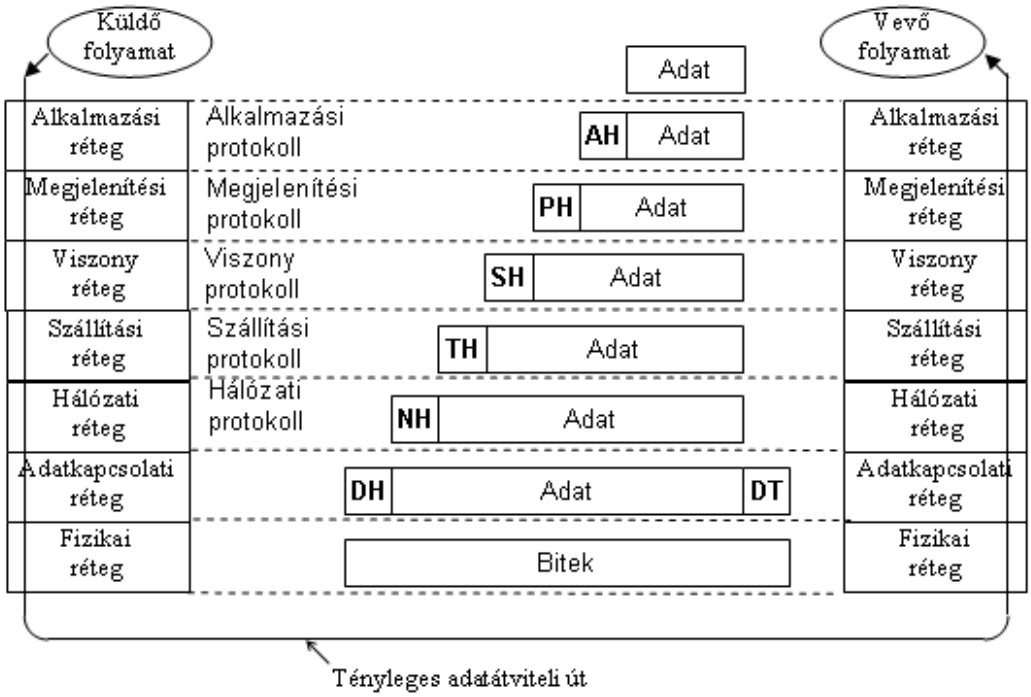
A réteg feladata még fájlátvitelkor (file transfer) az eltérő névkonvenciók kezelése, az elektronikus levelezés, a távoli munkabevitel, a katalóguskikeresés stb. és általában minden feladat, amit Internet szolgáltatásként ismerünk.

1.2.3.1.8. Adatátvitel az OSI hivatkozási modellben.

Az 1.7. ábrán az adattovábbítás mikéntje látható az OSI modellben. Ha a küldő folyamat adatot akar küldeni, akkor azt átadja az alkalmazási rétegnek, mely az adat elé beilleszti az AH fejrészt. Ezután a megjelenítési réteghez kerül az adat, ahol megkapja a PH fejrészt. Ez a folyamat addig ismétlődik, amíg a különböző fejrészekkel kiegészült adat el nem jut a fizikai réteghez, ahol aztán továbbítódik a vevő géphez. Ott -miközben az adat egyre magasabbra jut az egyes rétegek közt- a fejrészek leválasztódnak róla. Fontos tudni, hogy például a megjelenítési réteg nem tudja, hogy az alkalmazási rétegtől kapott adatok mely része AH, és melyik felhasználói adat. Az egyes rétegek úgy működnek, mintha vízszintes irányban továbbítanák az adatokat, míg a tényleges adatátvitel függőleges irányban történik. Például ha a küldő szállítási rétege kap egy üzenetet a viszony rétegtől, akkor ő azt (a fejrésszel kiegészítve) a vevő szállítási rétegének küldi tovább (bár az üzenetet a saját gépén lévő hálózati rétegnek továbbítja). Erre egy jó példa az lehet, hogy amikor egy "tagalog" nyelven beszélő diplomata felszólal az ENSZ-ben, akkor ő azt hiszi, hogy a többi jelenlévő diplomatához szól. Az, hogy ő valójában a tolmácsához beszél, csak részletkérdés.

1.2.3.1.9. Az OSI modell értékelése.

Az OSI modell protokolljai nem tökéletesek, bírálatokat nap mint nap hallhatunk



1.7. ábra. Adatátvitel az OSI modellben

róla, melyek egy része helyénvaló. Ami miatt az OSI modell "nem váltotta meg a világot" az a következő okoknak volt köszönhető: a rossz időzítés, a rossz technológia, a rossz implementálás és végül a rossz üzletpolitika.

Rossz időzítésről azért beszélhetünk, mert mire az OSI protokollok megjelentek, addigra a vetélytárs TCP/IP protokollok már széles körben elterjedtek a kutatóegyetemeken. Mivel az oktatási szférában a piac már igen nagy volt, a kereskedők elkezdték árusítani a TCP/IP termékeket. Mire az OSI megjelent a piacon nem akart támogatni egy második protokollkészletet, így kezdetben nem tudta eladni a termékeit.

Rossz technológia: egy másik probléma volt a modellel, mégpedig az, hogy a protokollok nem tökéletesek. A viszony réteget alig használják alkalmazások, a megjelenítési réteg szinte teljesen üres. Ugyanakkor az adatkapcsolati és hálózati rétegeknek annyi feladata van, hogy több alrétegre kellett őket bontani. Az egyes rétegekhez rendelt funkciók sokszor nem egyértelműek. Többek közt ezek miatt került sor különböző kiegészítések beiktatására.

Rossz implementálásról is beszélhetünk az OSI modell kapcsán, mivel rendkívüli bonyolultsága miatt kezdetben lassúak, kezelhetetlenek és igen terjedelmesek voltak az implementációk. Így rövid időn belül egy igen negatív kép alakult ki a modellről, amit a később egyre javuló termékek sem tudtak orvosolni.

Rossz üzletpolitika: az OSI nem terjedhetett el az oktatási szférában, mivel ott sokan azt hitték, hogy a TCP/IP a UNIX része, márpedig a UNIX ott igen népszerű volt.

Ezek, és más kevésbé fontos okok egyértelművé tették az OSI modell alulmaradását a TCP/IP-vel szemben. Ennek ellenére többek közt néhány távközlési szolgáltató a mai napig használja az 1994-es kiadású javított OSI modellt.

1.2.3.2. A TCP/IP hivatkozási modell

	OSI
7	Alkalmazási réteg
6	Megjelenítési réteg
5	Viszony réteg
4	Szállítási réteg
3	Hálózati réteg
2	Adatkapcsolati réteg
1	Fizikai réteg

TCP/IP
Alkalmazási réteg
Szállítási réteg
Internet réteg
Hoszt és hálózat közötti réteg

A TCP/IP-t az amerikai védelmi minisztérium (U.S. Department of Defense, DoD) által támogatott ARPANET nevű kísérleti hálózat részére fejlesztették ki. A modellel szemben támasztott követelmények így elsősorban a megbízhatóságra, illetve sok hálózat zökkenőmentes összekapcsolására irányultak. Az OSI modellel összehasonlítva azonnal feltűnik, hogy a TCP/IP-ből hiányzik a megjelenítési és viszony réteg, ugyanakkor a hálózati réteg helyett az Internet réteg, míg az adatkapcsolati és fizikai rétegek helyett a hoszt és hálózat közötti réteg található.

1.2.3.2.1. Az Internet réteg.

Ennek a rétegnek az a feladata, hogy egy hoszt bármilyen hálózatba csomagokat tudjon küldeni, illetve a csomagokat a célállomástól függetlenül (lehetőleg egy másik hálózaton) képes legyen továbbítani. Az sem gond, ha a csomagok nem az

elküldés sorrendjében érkeznek meg, ugyanis, ha erre van szükség, akkor a magasabb rétegek visszarendezik őket a megfelelő sorrendbe.
Az Internet réteg meghatároz egy hivatalos csomagformátumot, illetve egy protokollt, amelyet Internet protokollnak (Internet Protocol, IP) hívnak. Az Internet réteg feladata az, hogy ahová csak lehetséges, kézbesítse az IP csomagokat. A csomagok útvonalának meghatározása, valamint a torlódások elkerülése itt most a legfontosabb feladat.

1.2.3.2.2. Szállítási réteg.

Feladata, hogy lehetővé tegye a forrás- és célállomásokban található társentitások közti párbeszédet. Két különböző szállítási protokollt definiálunk a következőkben.

Az egyik az átvitelvezérlő protokoll (Transmission Control Protocol, TCP), amely egy megbízható összeköttetés alapú protokoll. Feladata az, hogy hibamentes bájtos átvitelt biztosítson bármely két gép között az Interneten.
A TCP forgalomszabályozást is végez annak érdekében, hogy egy gyors forrásállomás csak annyi üzenetet küldjön egy lassabb célállomásnak, amennyit az fogadni képes.

A másik protokoll ebben a rétegben a felhasználói datagram protokoll (User Datagram Protocol, UDP), amely egy nem megbízható, összeköttetés nélküli protokoll. Elsősorban olyan kliens-szerver típusú kérés-válasz alkalmazásokban terjedt el, ahol a gyors válasz sokkal fontosabb, mint a pontos válasz. Ilyen alkalmazás például a beszéd- vagy videó átvitel.

1.2.3.2.3. Alkalmazási réteg.

Az OSI modellel kapcsolatos tapasztalatok azt mutatták, hogy a viszony, és megjelenítési réteget viszonylag kevés alkalmazás használja, így a TCP/IP-be be sem kerültek. A szállítási réteg fölött az alkalmazási réteg található. Ez tartalmazza az összes magasabb szintű protokollt: virtuális terminál (TELNET), fájltranszfer (FTP), elektronikus levelezés (SMTP), Domain Name Service (DNS), hírlevelek szétküldése (NNTP), vagy a http, amely a World Wide Web oldalak letöltését segíti.

1.2.3.2.4. A TCP/IP modell értékelése.

A TCP/IP modellnek és protokolljainak szintén megvannak a maga hibái.

Először is megfelelő szoftvermérnöki tapasztalat hiányában nem tudunk különbséget tenni a specifikáció és az implementáció között.

Másodsorban a TCP/IP modell egyáltalán nem általános érvényű, és a TCP/IP-n kívül más protokollkészletek leírására nem igazán alkalmas.

Harmadsorban a TCP/IP nem különbözteti meg a fizikai és adatkapcsolati réteget, pedig egy jó modellnek külön kellene kezelnie mindkettőt.
Mindezek ellenére még mindig rengetegen használják, hiszen végső soron egy alaposan átgondolt modellről van szó.

1.2.3.3. Hálózatszabványosítás

A hálózatok megjelenésekor minden számítógépgyártó saját hálózati protokollokkal rendelkezett. Ennek következtében, azok a felhasználók, akik több gyártótól származó géppel rendelkeztek, nem tudták azokat egyetlen hálózatba összekötni. A szabványok nemcsak különböző számítógépek kommunikációját teszik lehetővé, de szélesítik a szabványhoz ragaszkodó termékek piacát, amely végső soron a tömeggyártáshoz, a gyártás gazdaságosságának a növeléséhez stb. vezet.

1.2.3.4. Néhány jellegzetes hálózatról röviden

A világon ma számos számítógép-hálózat működik, közülük néhány nyilvános hálózat, melyet közszolgáltatók üzemeltetnek, mások kísérleti-, kutató-, kereskedelmi-, és testületi hálózatok stb. Pl. az APRANET az USA Hadügyminisztériuma által létrehozott hálózat, de ezen kívül még számos másik is létezik, úgymint a USENET, CSNET, BITNET stb.

1.2.3.5. Adatkommunikációs példák

Telefontársaságok és más cégek különböző hálózati szolgáltatásokat kínálnak az előfizetőiknek. Az alhálózat, amely az ügyfelek hosztjait és termináljait kiszolgálja, az üzemeltető tulajdonában van. Az ilyen rendszert nyilvános hálózatnak hívják. Néhány ilyen szolgáltatás: SMDS, X.25 hálózatok, Frame relay, Szélessávú ISDN és ATN, stb.

1.3. A fizikai réteg

Ebben a fejezetben a 2.8.ábrán látható hierarchia legalsó rétegével foglalkozunk. Ezek után az átviteli közegekről lesz szó, ismertetjük a vezetékes (réz- és üvegszálas), illetve a vezeték nélküli közegeket.

1.3.1. Az adatátvitel elméleti alapjai

Információt úgy lehet vezetéken továbbítani, hogy az adó megváltoztatja a csatorna fizikai közegének valamilyen tulajdonságát. Ez a közegen továbbterjed, és a vevő ezt a fizikai közegváltozást érzékeli (ezt nevezzük jelnek).
Például vezetékek esetén az átfolyó áram változhat, vagy a feszültség, vagy ha elektromágneses hullámot használunk, akkor a hullám amplitúdója, frekvenciája, vagy kezdeti fázisszöge. Ha a feszültség vagy az áramerősség változását egy egyváltozós időfüggvénnyel, f(t)-vel írjuk le, akkor modellezni tudjuk a jelek viselkedését, és matematikai eszközökkel elemezni.

1.3.1.1. Fourier-analízis

A 19. század elején Jean-Baptiste Fourier francia matematikus bebizonyította, hogy bármely T periódusidővel rendelkező, periodikus g(t) függvény előállítható szinuszos és koszinuszos tagok összegeként.

$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \sin(2\pi n f t) + \sum_{n=1}^{\infty} b_n \cos(2\pi n f t)$$

$$a_n = \frac{2}{T}\int_0^T g(t)\sin(2\pi n f t)\,dt$$
$$b_n = \frac{2}{T}\int_0^T g(t)\cos(2\pi n f t)\,dt$$
$$c = \frac{2}{T}\int_0^T g(t)dt$$

ahol f =1/T az alapfrekvencia, a_n és b_n pedig az n-edik harmonikus (tag) szinuszos, illetve koszinuszos amplitúdója. Ezt a felbontást Fourier-sornak nevezzük. A Fourier-sor alapján az eredeti függvény visszaállítható.

1.3.1.2. Sávkorlátozott jelek

Tegyük fel, hogy egy 8 bites bájt formájában kódolt ASCII "b" karaktert akarunk elküldeni. A továbbítandó bitminta a '01100010'. A 1.8. (a) ábra bal oldalán azt láthatjuk, hogy a számítógép kimenetén hogyan változik a feszültség értéke. A jel Fourier-sora az alábbi együtthatókat tartalmazza:

$$a_n = \frac{1}{\pi n}\left[\cos\left(\pi\frac{n}{4}\right) - \cos\left(3\pi\frac{n}{4}\right) + \cos\left(6\pi\frac{n}{4}\right) - \cos\left(7\pi\frac{n}{4}\right)\right]$$

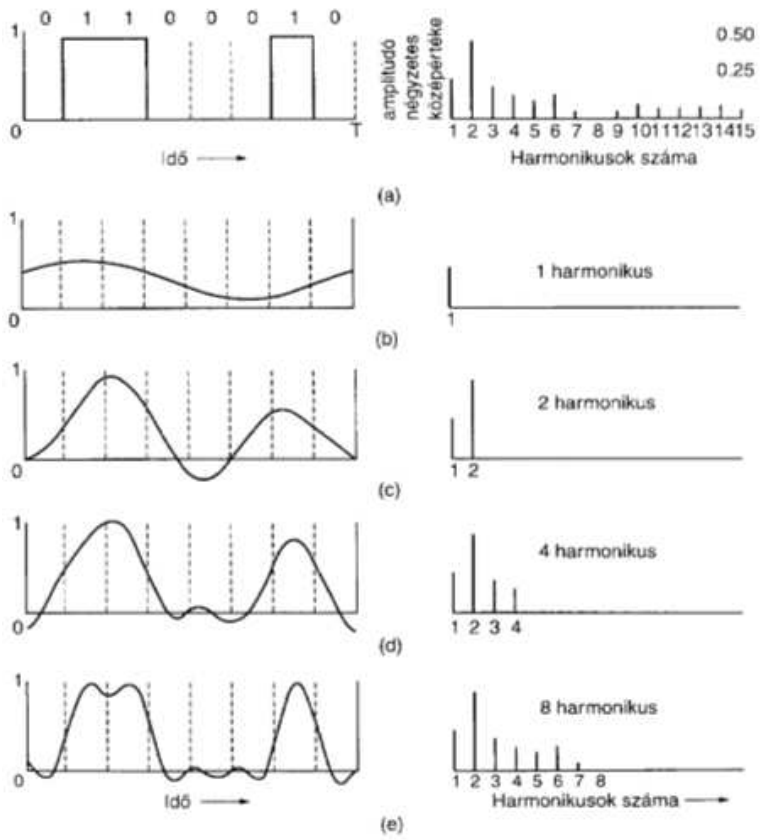
$$b_n = \frac{1}{\pi n}\left[\sin\left(3\pi\frac{n}{4}\right) - \sin\left(\pi\frac{n}{4}\right) + \sin\left(7\pi\frac{n}{4}\right) - \sin\left(6\pi\frac{n}{4}\right)\right]$$

$$c = \frac{3}{4}$$

A 1.8.(a) ábra jobb oldalán az első néhány harmonikus amplitúdójának négyzetes középértékét láthatjuk. Ezek azért fontosak, mert négyzetösszegük arányos az adott frekvencián továbbított energiával.

A 1.8.(b) ábrán az a jelalak látható, amelyet akkor kapnánk, ha a csatorna csak az első harmonikust (az alapharmonikust) továbbítaná. A 1.8.(c)–(e) ábrákon a továbbított jel spektruma és visszaállítás utáni jelalakja látható a nagyobb sávszélességű csatornák esetén.

Egy karakter T átviteli ideje függ a kódolás módjától és a jelzési sebességtől (signaling rate). A jelzési sebesség a másodpercenkénti jelváltások (pl. feszültségszintváltozások) száma. A jelzési sebesség mértékegysége a baud. Egy b baudos vonal nem biztos, hogy b bitet visz át másodpercenként, ugyanis egy jelváltás akár több bitet is jelenthet. Ha a bitsebesség b b/s lenne, akkor pl. 8 bit elküldése 8/b másodpercig tartana, tehát az alapharmonikus frekvenciája b/8 Hz lenne. A hagyományos telefonvonalat szándékosan úgy alakították ki, hogy 3000 Hz körül legyen a vágási frekvenciája. Ez a korlátozás azt jelenti, hogy a legmagasabb áteresztett felharmonikus frekvenciája durván 3000/(b/8) Hz-es, azaz 24000/b Hz-es, tehát a levágás nem túl éles. A sávszélesség korlátozása korlátozza az adatátviteli sebességet is, és ez még zajmentes csatorna esetén is igaz. Vannak olyan kódolási eljárások, amelyek több különböző feszültség szintet használnak, és jóval nagyobb adatátviteli sebességet lehet velük elérni.



1.8. ábra. Egy jel Fourier analízise (a) Digitális jel és Fourier-együtthatóinak négyzetes középértéke. (b)-(e) Az eredeti jel sorozatos közelítése

1.3.1.3. Maximális adatátviteli sebesség

Ha egy tetszőleges jelet egy H sávszélességű aluláteresztő szűrőn bocsátunk át, akkor a szűrt jelből másodpercenként vett 2H minta alapján az eredeti jel helyreállítható. Másodpercenként 2H mintánál többet nem érdemes venni a jelből, mivel a szűrő kiszűrné azokat a magasabb frekvenciájú komponenseket, amelyeket a mintavételezéssel helyre tudnánk állítani. H. Nyquist 1924-ben fölállított egy képletet a végső sávszélességű, zajmentes csatornán másodpercenként átvihető maximális adatmennyiség kiszámítására. Eszerint, ha a jelnek V különböző diszkrét szintje van, akkor a maximális adatsebesség:

$D_{Smax} = 2Hlog_2V[b/s]$.

Például egy zajmentes, 3 kHz sávszélességű csatornán bináris (azaz kétszintű) jelek továbbítása esetén nem lehet 6000 b/s-nál nagyobb adatsebességet elérni.

A csatorna zajának mértékét a jel és a zaj teljesítményének arányával, vagy más szóval a jel-zaj viszonnyal (signal-to-noise ratio) jellemezhetjük. Ha a jel teljesítményét S-sel, a zaj teljesítményét pedig N-nel jelöljük, akkor a jel-zaj viszony S/N. Általában nem a teljesítmények hányadosát szokták megadni, hanem a $10log_{10}S/N$ értéket, aminek a mértékegysége a decibel (dB). Ha az S/N hányados 10, akkor az 10 dB-t jelent, ha a hányados 100, akkor az 20 dB, ha a hányados 1000, akkor az 30 dB stb. A maximális adatátviteli sebesség egy olyan zajos csatornán, amelynek sávszélessége H, jel-zaj viszonya pedig S/N, a maximális adatsebesség:

$D_{Smax} = Hlog_2(1 + S/N)[b/s]$.

Ezt 1984-ben Claude Shannon terjesztett ki.

Például egy 3000 Hz sávszélességű csatornán 30 dB jel-zaj viszony esetén (ami a távbeszélőrendszerek analóg részében egy tipikus érték) az adatátviteli sebesség sosem lehet több, mint 30 000 b/s.

1.3.2. Az átvitel fizikai közege

Mindegyik közegnek megvan a maga sajátossága a sávszélességet, a késleltetést, az árát, a kiépítés nehézségeit és az üzemeltetést illetően. Az átviteli közegeknek két nagy csoportja van: az egyik a fizikailag összekötött (bounded), a másik pedig a fizikailag nem összekötött (unbounded) közegek csoportja.

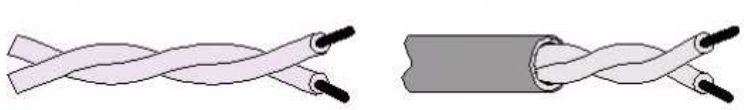
1.3.2.1. Mágneses hordozó

Két számítógép között az adatátvitel gyakori módja az, amikor az adatokat az egyik gépen mágnesszalagra, mágneslemezre vagy hordozható memóriára írjuk fel, ezt fizikailag átvisszük a másik gépre, ahol beolvassuk az adatokat.

1.3.2.2. Sodort érpár (twisted pair)

Bár sávszélesség szempontjából a mágnesszalag kiváló, sajnos igen jelentős késleltetéssel rendelkezik. Az adatátviteli időt percekben vagy órákban lehet mérni, nem pedig milliszekundumokban. A legtöbb alkalmazás esetén on-line összeköttetésre van szükség.

A legrégebbi, de még ma is a legelterjedtebb átviteli közeg a csavart érpár melynek szerkezete az alábbi ábrán látható.



1.9. ábra. Csavart érpár

A csavart érpár két szigetelt rézhuzalból áll, melyek általában 1 mm vastagságúak. A rézhuzalok spirálszerűen egymás köré vannak tekerve. A két eret azért sodorják össze, hogy csökkentsék a kettő közötti elektromágneses kölcsönhatást. (Ugyanis két párhuzamos huzal mágneses hurokként működik, szemben a csavart érpárral.)

A csavart érpárt leggyakrabban a távbeszélőrendszerekben használják. A csavart érpárt akár több kilométeres szakaszon is erősítés nélkül lehet használni, de nagyobb távolságok esetén már szükség van erősítőkre. Amikor hosszabb távolságon keresztül több csavart érpár fut egymás mellett, akkor a csavart érpárokat egy kötegbe fogják, és ezt a köteget mechanikai védelemmel látják el. Ha az érpárok nem lennének csavarva, akkor a kötegen belül zavarnák egymás forgalmát. A csavart érpár alkalmas mind analóg, mind digitális jelek átvitelére.

A vezetékek sávszélessége a vastagságától és az áthidalt távolságtól függ, de sok esetben néhány Mb/s sebességet is el lehet velük érni pár kilométeres távolságon belül.

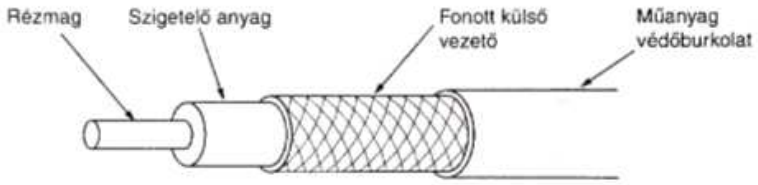
A csavart érpárnak számos változata van, a számítógép hálózatok szempontjából csak a 3-as és 5-ös kategóriájúnak van jelentősége, ezeket árnyékolatlan csavart érpárnak (unsielded twisted pair, UTP) is nevezik.

1.3.2.3. Alapsávú koaxiális kábel

Egy másik, széles körben használt átviteli közeg a koaxiális kábel (coaxial cable), amit egyszerűen csak "koax"-nak hívnak. Mivel ez jobb árnyékolással rendelkezik, mint a csavart érpár, ezért nagyobb távolságot lehet vele áthidalni, ugyanakkor nagyobb sebességgel rendelkezik.

Kétfajta koaxiális kábel létezik. Az egyik az $50\Omega - os$ kábel, amelyet elsősorban digitális átvitelhez használnak, a másik a $75\Omega - os$ kábel, amelyet viszont elsősorban analóg átvitel esetén használnak.

A koaxiális kábel közepén lévő tömör rézhuzal magot szigetelő veszi körül, melytől nagymértékben függ a kábel hullámimpedanciája. A szigetelő körül sűrű szövésű hálóból álló vezető található. A külső vezetőt mechanikai védelmet is biztosító műanyag burkolattal vonják be. A koaxiális kábel szerkezetét a 1.10. ábrán láthatjuk.
A koaxiális kábel szerkezetének és árnyékolásának köszönhetően kifejezetten alkalmas nagy sávszélességű jelek továbbítására, miközben a zajérzékenysége igen csekély. Egy 1 km-es kábellel akár 1-2 Gb/s-os átviteli sebességet is elérhetünk. A koaxiális kábeleket széles körben alkalmazzák a telefonhálózatoknál, bár az utóbbi időkben a nagy távolságú átviteli rendszerekben egyre inkább optikai kábelekre cserélik le azokat. Ugyanakkor a koaxiális kábel még mindig nagyon elterjedt a kábeltelevíziózásnál és néhány lokális hálózatnál. Csatlakozásra BNC (Bayone-



1.10. ábra. Koaxiális kábel

Neil-Councelman) dugókat és aljzatokat használnak a reflexiók elkerülése végett. Ezek a 1.11. ábrán láthatók. Mivel a csatlakozások mindig a kábelezés legkritikusabb pontjai, célszerűbb a biztonságosabb kötést biztosító sajtolt (krimpelt) csatlakozók használata, a csavaros, vagy forrasztott BNC csatlakozókkal szemben.

1.11. ábra. BNC csatlakozók

1.3.2.4. Szélessávú koaxiális kábel

A szélessávú (broadband) koaxiális kábeles rendszer analóg átvitelt valósít meg a szabványos kábeltelevíziós rendszerben. Mivel a szélessávú hálózatok a szabványos kábeltelevíziós technikát használják, így a kábelek 300 MHz (sőt, gyakran 450 MHz) sávszélességet biztosítanak, és az analóg átvitelnek köszönhetően közel 100 km-es távolság áthidalására képesek, ugyanis az analóg jelátvitel sokkal kevésbé kritikus, mint a digitális.

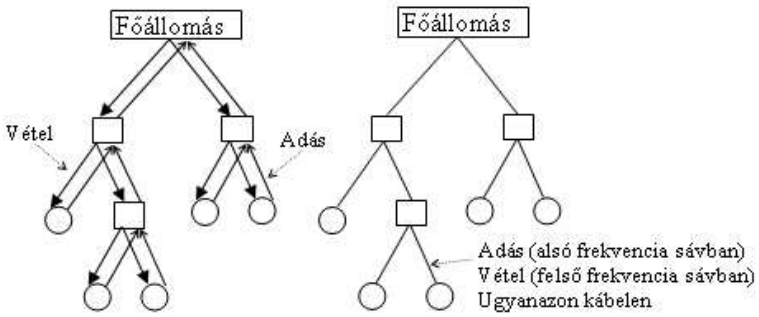
Ahhoz, hogy egy analóg hálózaton digitális adatokat továbbíthassunk, valamennyi hálózati interfésznek olyan elektronikával kell rendelkeznie, amely a kimenő bitfolyamot analóg jelekké, a beérkező analóg jeleket pedig bitfolyammá alakítja át. Az elektronikus áramkörök típusától függően 1 b/s adatsebesség durván 1 Hz sávszélességet igényel. Fejlettebb modulációs eljárásokkal magasabb frekvencián több bit is átvihető másodpercenként.

A szélessávú rendszereket általában több csatornára osztják fel. A televíziós műsorszórásban rendszerint 6 MHz-es csatornákat használnak. Az alapsávú és a szélessávú rendszerek között az a leglényegesebb különbség, hogy a szélessávú rendszerek általában nagy területet fednek le, ezért szükség van a jelek rendszeres analóg erősítésére. Ezek az erősítők a jeleket csak egy irányba tudják továbbítani, így a számítógép nem tudja elküldeni a csomagjait egy "fölötte" levő másik gépnek, amennyiben a két számítógép között csak egy erősítő van.

Ennek a problémának a megoldására két különböző szélessávú rendszert fejlesztettek ki: a kétkábeles és az egykábeles rendszert (lásd 1.12.(a) és(b) ábrák).

A kétkábeles rendszerben két azonos kábel fut párhuzamosan egymás mellett. Adatok küldésekor a számítógép az 1-es kábelre teszi ki az adatokat, amelyek a fa topológiájú kábelezés gyökerénél található főállomáshoz (bead-end) jutnak el. Ezt követően a főállomás a jeleket a 2-es kábelen lefelé továbbítja a fában. Minden számítógép az 1-es kábelen küldi, és a 2-es kábelen veszi az adatokat.

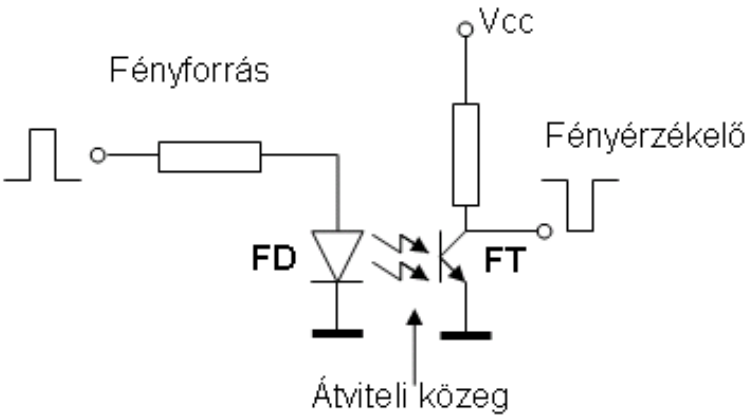
Az egykábeles rendszerben egyetlen kábelen különböző frekvenciasávokban történik az adatok elküldése, illetve vétele . Alsómetszésű (subsplit) rendszerekben az adás az 5 és 30 MHz közötti frekvenciasávban történik, míg a vétel a 40 és 300 MHz közötti frekvenciasávban. A középmetszésű (midsplit) rendszerekben az adósávok 5 és 116 MHz között, míg a vevősávok 168 és 300 MHz között vannak.



1.12. ábra. Szélessávú hálózatok. (a) Kétkábeles. (b) Egykábeles

1.3.2.5. Fényvezető szálak

A jelenlegi legkorszerűbb fizikailag összekötött adatátviteli módszer az üvegszál technológia alkalmazása. Az információ fényimpulzusok formájában terjed egy fényvezető közegben, általában egy üvegszálon. A rendszernek 3 komponense van: a fényforrás, az átviteli közeg, és a fényérzékelő (1.13. ábra). A fényimpulzus



1.13. ábra. Az optikai adatátvitel alapelve

megléte szokás szerint a logikai 1 bitet jelenti, míg az impulzus hiánya a logikai 0 bitet.

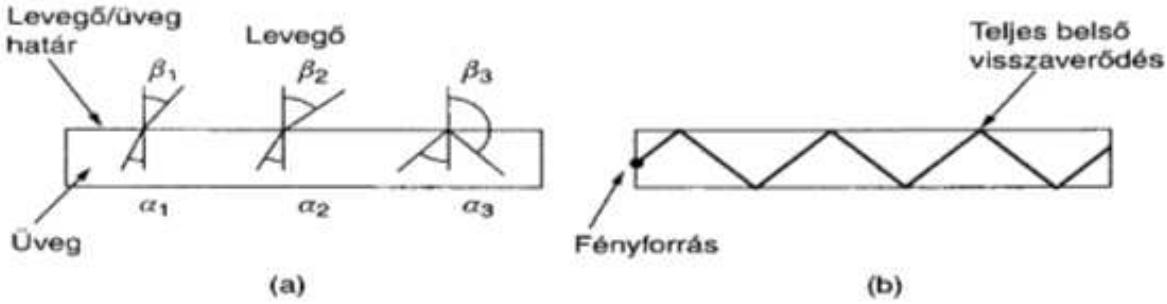
Az átviteli közeg egy rendkívül vékony üvegszál. Ha a detektorba fény jut, akkor a detektor villamos jelet állít elő. Ha az üvegszál egyik végére fényforrást, a másik végére pedig detektort teszünk, akkor egy olyan egyirányú adatátviteli rendszert kapunk, amely villamos jeleket fogad, átalakítja azokat fényimpulzusokká, tovább-

bítja a fényimpulzusokat, majd a kábel másik végén a fényimpulzusokat visszaalakítja villamos jelekké.

A látható fény frekvenciája közel 10^8 MHz, így egy optikai adatátviteli rendszer sávszélessége potenciálisan óriási. Amikor a fény egyik közegből átlép egy másik közegbe, mondjuk üvegből a levegőbe, akkor az üveg és a levegő találkozásánál a fény megtörik, ahogy ez a 1.14.(a) ábrán is látható. Az ábra egy olyan fénysugarat mutat, amely $\alpha 1$ szögben érkezik meg a határfelülethez, és $\beta 1$ szögben halad tovább. A visszaverődés mértéke függ a két közeg fizikai jellemzőitől (elsősorban azok törésmutatójától). Ha a beesési szög nagyobb egy bizonyos határértéknél, akkor a fény nem lép ki a levegőbe, hanem visszaverődik az üvegbe. Így ha a fénysugár beesési szöge egyenlő a határszöggel vagy nagyobb annál, akkor a fénysugár az üvegszálon belül marad, ahogy ezt a 1.14.(b) ábra is szemlélteti, és akár több kilométert is megtehet gyakorlatilag veszteség nélkül. A 1.14.(b) ábrán csak egyetlen fénysugár látható, de mivel a határszöggel azonos vagy annál nagyobb szögben beeső sugarak mind az üvegszálon belül maradnak, ezért egyszerre sok, különböző szögben visszaverődő fénysugár halad az üvegszálban. Minden egyes sugárnak más és más az ún. módusa, ezért az ilyen üvegszálat többmódusú szálnak nevezik.

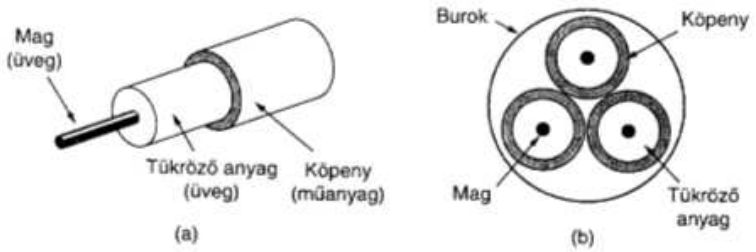
Ha viszont az üvegszál átmérőjét néhány fényhullámhossznyira lecsökkentjük, akkor az üvegszál hullámvezetőként viselkedik, és a fény visszaverődés nélkül, egyenes vonal mentén terjed a vezetékben. Az ilyen üvegszálat egymódusú szálnak nevezik. Az egymódusú szálak jóval drágábbak, viszont nagyobb távolságok áthidalására használhatók. A jelenleg kapható egymódusú üvegszálak másodpercenként néhány gigabitet képesek 30 km-re továbbítani.

A fényvezető kábel a fonott árnyékolástól eltekintve hasonlít a koaxiális kábelre. A 1.15.(a) ábra oldalnézetben mutat egy fényvezető szálat. Középen található az üvegmag, amiben a fény terjed. Többmódusú szál esetén a mag 50 mikron átmérőjű, azaz körülbelül olyan vastag, mint egy emberi hajszál. Egymódusú szál esetén a mag 8-10 mikron átmérőjű. Az üvegmagot olyan üvegköpeny veszi körül, amelynek a törésmutatója kisebb, mint a magé, így a fénysugár a magon belül marad. A szálat kívülről műanyag védőburkolattal látják el a köpeny védelme érdekében. A fényvezető kábelben általában több szálat fognak össze, és azokat egy műanyag csőbe helyezve védik a külső behatásoktól. A 1.15.(b) ábrán egy háromszálas kábel keresztmetszetét láthatjuk.



1.14. ábra. A fénysugár útja az üvegszálban. (a) Egy üvegszál belsejében a fénysugár három különböző szögben érkezik az üveg és a levegő határához. (b) A teljes visszaverődés miatt a fénysugár az üvegszálon belül marad

A szárazföldi fénykábeleket általában egy méter mélyre fektetik, a tengeri kábeleket a partok közelében vízi eke segítségével beszántják a tengerfenék alá, míg a mélyebb vizekben pedig egyszerűen csak leengedik a kábeleket a tengerfenékre.



1.15. ábra. Optikai szálak. a) Fényvezető szál oldalnézetben. (b) Három fényvezető szálból álló kábel keresztmetszete

A fényvezető szálakat háromféleképpen lehet egymáshoz csatlakoztatni. Az egyik módszer az, hogy a fényvezető szál végeit megfelelő csatlakozókkal látjuk el, és ezeket dugjuk össze. A csatlakozók 10-20viszont megkönnyítik a rendszer újrakonfigurálását.

A második lehetőség, hogy a szálakat mechanikusan egymáshoz illesztjük. Ennek a módszernek az a lényege, hogy mindkét szálat meghatározott szögben óvatosan lenyessük, majd a nyesett végeket összeillesztjük, és egy szorítóval összefogjuk. Az illesztés pontossága úgy javítható, hogy az egyik üvegszálba belevilágítunk, és a két szálat finoman addig mozgatjuk, amíg a kijövő jel intenzitása a lehető legnagyobb nem lesz.

A harmadik lehetőség az, hogy a két kábelt összeforrasztjuk. A forrasztott szál majdnem olyan jó, mint egy gyárilag húzott szál, de azért még itt is van némi csillapítás. Mindhárom csatlakoztatási mód esetén van egy kis visszaverődés az illesztésnél, és a visszaverődött fény interferálhat az eredeti jellel.

A fényimpulzusok előállítására kétféle fényforrást használnak: az egyik a LED (Light Emitting Diode), a másik pedig a félvezető lézer. Mint azt az alábbi táblázat mutatja, a két fényforrás sokban különbözik egymástól.

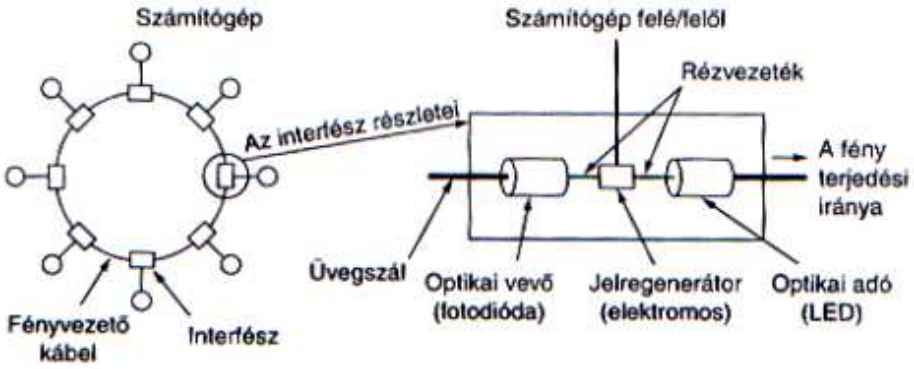
Jellemző	LED	Félvezető lézer
Adatátviteli sebesség	Alacsony	Magas
Módus	Többmódusú	Több vagy egymódusú
Távolság	Kicsi	Nagy
Élettartam	Hosszú	Rövid
Hőmérséklet érzékenység	Kicsi	Jelentős
Ár	Olcsó	Drága

Ethernet hálózatokban az üvegszálas kábelt 10BaseF néven definiálták. Fényvezető szálas kábeleket mind a lokális, mind pedig a nagy kiterjedésű hálózatokban jól lehet alkalmazni, bár ezekhez a hálózatokhoz történő csatlakozás korántsem olyan egyszerű, mint az Ethernet esetében.

Az interfészeknek két típusa van. A passzív interfész két, az üvegszálra kapcsolódó csatlakozóból áll. Az egyik csatlakozón egy LED, a másikon egy fotodióda van. Az illesztő teljesen passzív, így rendkívül megbízható, segítségével jeleket tudunk a fénykábelből kivenni, illetve jeleket tudunk a kábelbe juttatni. Az illesztés természetesen fényveszteséggel jár, ezért meg kell határozni, hogy adott távolságon hány illesztés használható. Ha a LED, vagy a fotodióda meghibásodik, akkor a gyűrű nem szakad meg, csak a számítógép kapcsolódik le a gyűrűről.

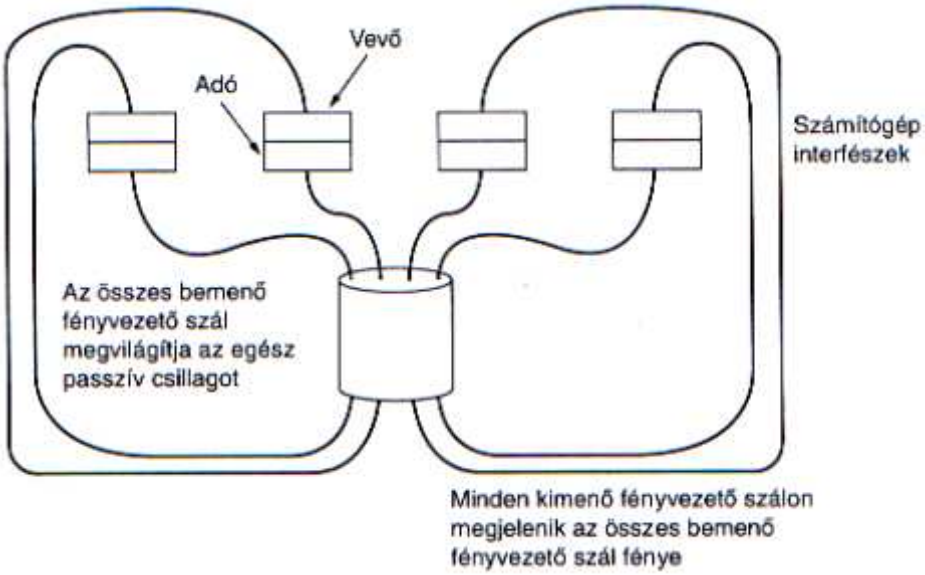
Az aktív illesztő jelismétlőként is működik, azaz a beeső fényjelet a VEVŐ villamos jellé alakítja, majd az ADÓ részén ezt LED segítségével felerősítve továbbsugározza. Mivel a regenerálás folyamán a kábelen haladó fényjel villamos jelként is megjelenik, ezért ez közvetlenül elektromos jelillesztésre is használható. A számítógép és a jelgenerátor közötti interfész egy hagyományos rézvezeték. Ha egy aktív ismétlő meghibásodik, akkor a gyűrű megszakad, és a hálózat működése leáll. Ugyanakkor a gyűrű teljes mérete nagyságrendekkel nagyobb lehet, mint passzív illesztés esetén, ugyanis itt mindegyik interfész újra előállítja a jelet. Egy gyűrű

topológiájú, aktív ismétlőkkel ellátott fényvezető szálas hálózatot mutat a 1.16. ábra.



1.16. ábra. Fényvezető szálas gyűrű aktív ismétlővel

A fényvezető szálas lokális hálózat nem csak gyűrű topológiájú lehet. A 1.17. ábrán látható passzív csillag (passive star) topológia segítségével hardver alapú adatszórást is meg lehet valósítani. Amikor valamelyik interfész kiad egy fényimpulzust, akkor az a passzív csillagban szétterjed, és eljuttatja a fényimpulzust az összes vevő egységhez. Így jön létre az adatszórás.



1.17. ábra. Passzív csillag egy fényvezető szálas hálózatban

Mint látjuk, a fényvezető szál rengeteg előnnyel rendelkezik a hagyományos rézvezetékkel szemben:

- Nagyobb sávszélesség
- Kis csillapítás, nagyobb szegmenshossz
- Külső zavaró hatásokra való érzéketlenség (áram, elektromágneses zavarok, elektromos hálózati kimaradások, korrodálódásra való érzéketlenség)
- Kis méretűek és súlyúak

Hátránya viszont, hogy a legtöbb mérnök nem rendelkezik a hálózat kiépítéséhez szükséges szaktudással. Ugyanakkor duplex kommunikációhoz két szálra van szükség (vagy két frekvenciasávra), de az amúgy sem olcsó illesztő, illetve kábelárak mellett ez szintén negatívum. Mindezek ellenére a jövőben nyilvánvalóan egyre több fényvezető kábeleket használnak majd a néhány méternél nagyobb távolságot áthidaló adatátviteli rendszerekben.

1.3.3. Vezeték nélküli adatátvitel

1.3.3.1. Rádiófrekvenciás átvitel

A rádióhullámok egyszerűen előállíthatók, nagy távolságra jutnak el, és könnyen áthatolnak az épületek falain, így széles körben használják ezeket mind kültéri, mind beltéri alkalmazásokban.

A rádióhullámok minden irányba terjednek, így az adót és a vevőt nem kell fizikailag precízen egymáshoz illeszteni.

A rádióhullámok terjedési tulajdonságai frekvenciafüggők.
Alacsony frekvencián a rádióhullámok minden akadályon áthatolnak, viszont a teljesítményük a forrástól távolodva erősen - a levegőben nagyjából $1/(r^3)$ szerint - csökken.
A nagyfrekvenciás rádióhullámok egyenes vonal mentén terjednek, és a tárgyakról visszaverődnek. Az eső elnyeli a nagyfrekvenciás rádióhullámokat.

A rádióhullámokat a villamos motorok és más elektronikus berendezések minden frekvenciatartományban zavarják. Mivel a rádióhullámok nagyon messzire eljutnak, ezért komoly problémát jelent a felhasználók közötti interferencia. Emiatt minden országban szigorúan engedélyhez kötik a rádióadóval ellátott eszközök használatát.

1.3.3.2. Mikrohullámú átvitel

100 MHz felett az elektromágneses hullámok egyenes vonal mentén terjednek, és ezért jól fókuszálhatók. Ha e hullámokat egy parabolaantenna (amilyen a házakon látható műholdas televízió antenna) segítségével egy keskeny nyalábba fogjuk össze, akkor a jel-zaj viszony sokkal jobb lesz, viszont ehhez az adó és vevő antennáját nagyon pontosan egymáshoz kell igazítani. Ráadásul, ez az irányítottság lehetővé teszi azt, hogy több, egymás mellé helyezett adóegység interferencia nélkül tudjon kommunikálni az egymás közelében levő vevőegységekkel.

Mivel a mikrohullámok egyenes vonal mentén terjednek, ezért a földfelszín görbülete problémát jelent, ha az adótornyok túlságosan messze vannak egymástól. Ezért meghatározott távolságonként ismétlőkre van szükség.

Az alacsony frekvenciás rádióhullámokkal szemben a mikrohullámok nem képesek áthatolni az épületek falain. Az adás minősége függ az időjárástól és a frekvenciától. Az egyre növekvő frekvenciaigény a műszaki fejlesztést arra sarkallja, hogy az átvitelt egyre magasabb frekvenciákon is lehetővé tegye. Ma már otthonosan mozgunk 10 GHz-es tartományban is, bár igaz, hogy 8 GHz környékén új problémával találjuk magunkat szemben, ez pedig az, hogy a víz elnyeli a mikrohullámú sugarakat.

1.3.3.3. Infravörös és milliméteres hullámú átvitel

A vezeték nélküli infravörös és milliméteres hullámokat elsősorban a kistávolságú adatátvitelben használják előszeretettel. Az infravörös hullám viszonylag jól irányítható, olcsó és könnyen előállítható. Azonban van egy óriási hátránya; szilárd testeken nem képes áthatolni. Másfelő1 persze előnynek is vehetjük, hogy az infravörös hullámok nem hatolnak át a falakon. Ez ugyanis azt jelenti, hogy az egyik szobában levő infravörös rendszer nem zavarja a szomszédos szobában levő másik ilyen rendszert. Pontosan emiatt az infravörös rendszerek biztonsági szempontból nagyobb védelmet jelentenek, mint a rádióhullámú rendszerek. Ha az infravörös kommunikációs rendszereket a szabadban kívánjuk használni, tervezéskor figyelembe kell venni, hogy a Nap ugyanolyan erősen süt az infravörös tartományban, mint a látható fény tartományában.

1.3.3.4. Látható fényhullámú átvitel

A vezeték nélküli fényjelzést már évszázadok óta használják. Paul Revere nevezetes útja előtt bináris fényjeleket küldött a bostoni Old North Church tornyából. Ennek egy modern változata az, amikor két épület lokális hálózatát a tetejükre szerelt lézerek segítségével kapcsoljuk össze. A lézert alkalmazó koherens optikai adatátvitel alapvetően egyirányú, így mindkét épületnek külön lézerforrásra és fényérzékelőre van szüksége. Ez a megoldás igen nagy sávszélességgel rendelkezik, és nagyon olcsó. Viszonylag egyszerű egy ilyen rendszert kiépíteni, és szemben a mikrohullámmal, nincs szükség hivatalos engedélyeztetésre. A lézer egyik nagy hátránya, hogy esőn és sűrű ködön nem képes áthatolni.

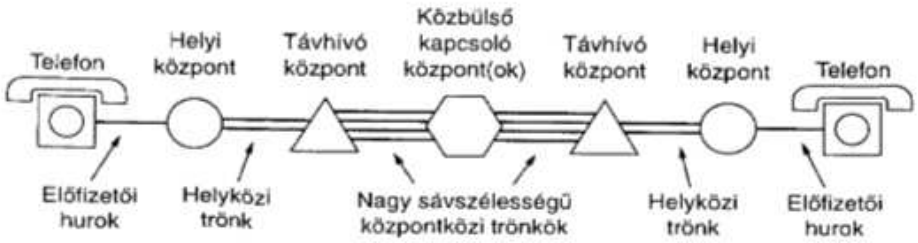
1.3.3.5. Távközlési műholdak

A távközlési műholdat felfoghatjuk úgy, mint egy világűrben levő mikrohullámú ismétlőt. A műhold számos transzponderrel (transponder) rendelkezik, amelyek közül mindegyik a frekvenciatartomány egy meghatározott részét figyeli, felerősíti a beérkező jeleket, és a beérkező jelek zavarásának elkerülése érdekében egy másik frekvencián visszasugározza azokat. Egy megközelítőleg 36 000 km magasságban, az Egyenlítő fölött keringő műhold sebessége megegyezik a Föld forgási sebességével (Geostacionárius műholdak. A kör alakú geostacionárius pálya felszín feletti magassága átlag 35 794 km).

1.3.4. Analóg átviteli rendszerek

Bizonyos esetekben a távbeszélőrendszer annyira összefonódik a (nagytávolságú) számítógép-hálózatokkal, hogy érdemes egy kis időt szentelni a bemutatásukra. A távbeszélő hálózatot, kiváltképp a nyilvános kapcsolt telefonhálózatot (Public Switched Telephone Network, PSTN) elsősorban az emberi beszéd továbbítására tervezték. Ezek számítógépek közötti kommunikációra való alkalmassága gyenge, viszont a fényvezető szálak és a digitális technika megjelenése következtében a helyzet rohamosan kezd megváltozni. A fő probléma az, hogy egy telefonvonalon az adatátviteli sebesség nagyságrendileg 10^4 b/s, a hibaarány pedig 10^{-5}, attól függően, hogy milyen régi az igénybevett telefonközpont. Ezzel szemben két számítógép között az adatátviteli sebesség tipikusan $10^7 - 10^8$ b/s, míg a hibaarány kb.

$10^{12} - 10^{13}$ bit/hiba. A kutatások jelenlegi iránya az, hogy miként lehet a mintegy 11 nagyságrenddel kisebb teljesítményű rendszert hatékonyan felhasználni.



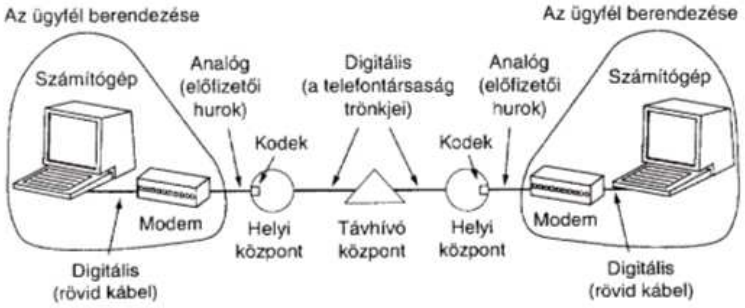
1.18. ábra. Tipikus áramköri út egy közepes távolságú hívás esetén

A több, mint 100 éves fejlesztéseknek köszönhetően a távbeszélőrendszer ma egy rendkívül redundáns, többszintű hierarchikus rendszer. Minden telefonkészülékből közvetlenül két rézvezeték megy a telefontársaság legközelebbi végközpontjába (end office), amit gyakran helyi központnak (local central office) is hívnak. A készülék és a központ közötti távolság 1 km-től 10 km-ig terjedhet. Ez a távolság a városokban általában kisebb, vidéken pedig nagyobb. Az előfizetők készüléke és a helyi központ közötti kétvezetékes összeköttetést előfizetői huroknak (local loop) nevezik. Ha egy adott helyi központhoz kapcsolódó állomásról olyan állomást hívunk, amelyik ugyanahhoz a helyi központhoz kapcsolódik, akkor a kapcsolás során a két előfizetői hurok között közvetlen elektromos kapcsolat jön létre a helyi központon belül. Ez a kapcsolat a hívás ideje alatt végig fennmarad. Ha viszont a hívott fél készüléke egy másik végközponthoz kapcsolódik, akkor az előző módszer nem használható. Minden végközpont kapcsolatban áll egy, vagy több közeli ún. távhívó központtal (toll office), amit tandem központnak (tandem office) is hívnak, ha ugyanazon a körzeten belül helyezkedik el, mint a helyi központ. A helyi központ és a távhívó központ közötti vonalakat helyközi trönköknek (toll connecting trunk) hívják.

Amennyiben a hívó és a hívott fél helyi központja ugyanahhoz a távhívó központhoz csatlakozik a helyközi trönkön keresztül, akkor az összeköttetés a távhívó központon belül jön létre.

Következésképpen, ha egy számítógép digitális adatot akar elküldeni a telefonhálózaton keresztül, akkor az előfizetői hurkon történő átvitelhez egy modem segítségével analóg jelekké kell átalakítani a digitális jeleket, majd a nagytávolságú trönkökön való továbbításhoz ezeket az analóg jeleket megint digitális jelekké kell

alakítani. A másik oldalon ugyanezt visszafelé kell elvégezni, tehát a trönkökön továbbított digitális jeleket analóg jelekké, majd modem segítségével ismét digitális jelekké kell alakítani ahhoz, hogy a továbbított adatokat a számítógép tárolni tudja. Ez az elrendezés látható a 1.19. ábrán.



1.19. ábra. Analóg és digitális adatátvitel két számítógép között

Az átalakításokat a modemek és a kodekek végzik.

1.3.4.1. Modemek

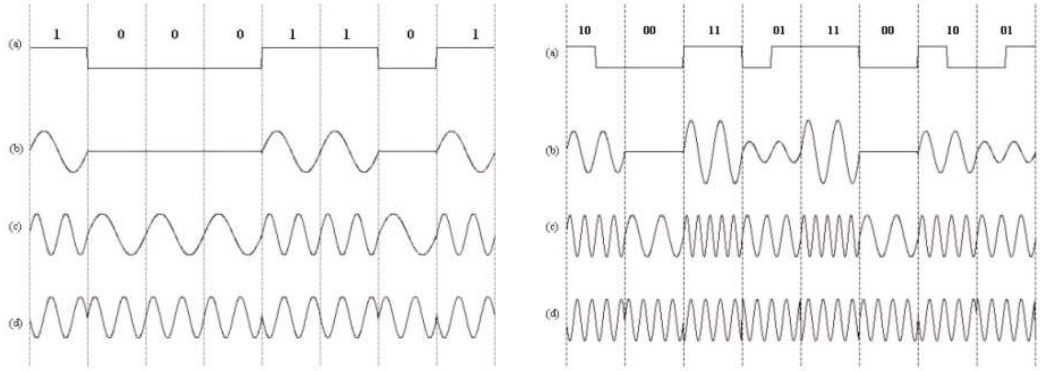
A fenti problémák - különösen a csillapítás és a terjedési sebesség frekvenciafüggése - miatt nem túl szerencsés, ha a jel széles frekvenciatartománnyal rendelkezik. Sajnos azonban a négyszöghullámok, tehát a digitális jelek spektruma igen széles, ezért ezeknél jelentős a csillapítás és a vonalkésleltetésből adódó jeltorzulás. Emiatt az alapsávú (egyenáramú, DC) jelzés digitális átvitel esetén nem járható út, legfeljebb kis adatátviteli sebességnél és kis távolságon belül használható.

Az egyenáramú átvitel problémáját a telefonvonalak esetében úgy oldották meg, hogy váltakozó áramú jelzést használtak. Bevezettek egy szinuszos vivőjelet (sine wave carrier), ami egy 1000 Hz és 2000 Hz közötti folytonos jel. A szinuszos vivőjel amplitúdójának, frekvenciájának vagy fázisának változtatása információ továbbítását teszi lehetővé.

Amplitúdómoduláció (amplitude modulation) esetén két feszültségszintet használnak a logikai 0 és 1 ábrázolására. *Frekvenciamodulációnál* (frequency modulation) - amit frekvenciabillentyűzésnek (frequency shift keying) is neveznek - két vagy több különböző frekvenciát használnak. A *fázismoduláció* (phase modulation) legegyszerűbb változatánál a vivőjel fázisát egyenlő időközönként szisztematikusan

45, 135, 225 vagy 315 fokkal eltolják. Minden egyes fázistolással 2 bitet lehet továbbítani. A 1.20. ábrán az előbb említett modulációs eljárások láthatók.

Azt az eszközt, amely a bemenetére érkező bitfolyamból modulált jeleket állít elő a kimenetén (vagy fordítva), modemnek (modulátor-demodulátor) nevezzük.

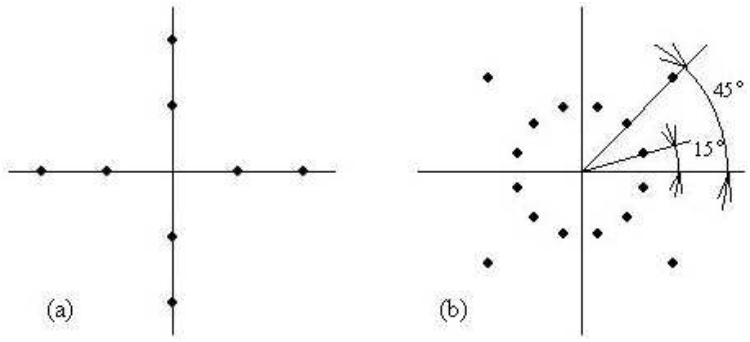


1.20. ábra. A moduláció formái. (a) Digitális jel. (b) Amplitúdómoduláció. (c) Frekvenciamoduláció. (d) Fázismoduláció 1 illetve 2 biten kódolva

Ahhoz, hogy egyre nagyobb sebességeket érhessünk el, nem elegendő csak a mintavételi frekvencia növelése. A legtöbb korszerű modemben különböző modulációs technikákat ötvöznek annak érdekében, hogy több bitet is át tudjanak vinni egy jelváltással. A 1.21.(a) ábrán a pontokkal jelölt 0, 90, 180, 270 fokos fázisok mindegyikéhez két különböző amplitúdó tartozik. Az amplitúdó esetünkben az origótól mért távolság. A 1.21.(b) ábrán egy ettől eltérő modulációs sémát láthatunk, amely 16 különböző amplitúdójú és fázisú értéket használ. A 1.21.(a) ábrán látható megoldás 8 érvényes kombinációt tartalmaz, amivel jelváltásonként 3 bitet lehet továbbítani. A 1.21.(b) ábrán látható megoldás viszont 16 érvényes kombinációt tartalmaz, és ezzel jelváltásonként már 4 bitet lehet továbbítani. Amikor 2400 baud-os vonalon 9600 b/s-mal továbbítanak adatokat, akkor a 1.21.(b) ábrán látható kvadratúra amplitúdómodulációt (Quadrature Amplitude Modalation, QAM) alkalmazzák.

Azokat a 1.21. ábrához hasonló diagramokat, amelyek az érvényes amplitúdó- és fázisértékeket mutatják, *amplitúdó-fázis diagramnak*, vagy csillagkép mintázatnak (constellation pattern) nevezzük.

Még modemek használata esetén is fellép a telefonvonalakon a visszhanghatás. Egy hosszú vonal végén a jel megérkezésekor valamennyi visszaverődik belőle, és



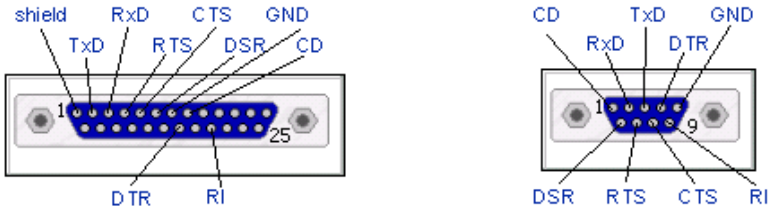
1.21. ábra. Kombinált moduláció. a) 3 bit/baud-os moduláció. (b) 4 bit/baud-os moduláció

ez pont olyan hatást kelt, mint a magas hegyekben megfigyelhető akusztikus visszhang. A visszhanghatás miatt van az, hogy telefonálás közben kis késleltetéssel a saját hangunkat halljuk vissza. A probléma kiküszöbölése érdekében a 2000 km-nél hosszabb vezetékeknél *visszhangelnyomókat* építettek be a rendszerbe. (Rövidebb vonalaknál a visszhang olyan gyorsan ér vissza, hogy az embereket általában nem zavarja.)

1.3.4.2. RS-232-C és RS422-A

A fizikai réteg jól ismert szabványa az RS-232-C, ami az eredeti RS-232 szabvány harmadik javított változata. A nemzetközileg elfogadott változatot a CCITT V.24 szabványa tartalmazza. A szabványokban a számítógépet vagy terminált hivatalosan adatvég-berendezésnek (Data Terminal Equipment, DTE), a modemet pedig adatáramkört lezáró berendezésnek (Data Circuit-Terminating Equipment, DCE) nevezik. A mechanikai leírás a 1.22. ábrán látható 25 és 9 tűs csatlakozót adja meg.

Az RS-232-C villamos leírása szerint a -3V és -15V közötti feszültség a bináris 1-et, míg a +3 V és +15 V közötti feszültség a bináris 0-t jelenti. Az adatátviteli sebesség legfeljebb 20 kb/s lehet, a kábel maximális hossza pedig 15 méter. A funkcionális leírás azt tartalmazza, hogy milyen áramkör kapcsolódik az egyes tűkhöz, és hogy azoknak mi a funkciója. A 1.23. ábrán 9 tűhöz tartozó áram-



1.22. ábra. RS-232-C

kör funkciója látható. Ezeket az áramköröket mindig megvalósítják, míg a többit általában elhagyják.

1.23. ábra. Néhány alapvető RS-232-C áramkör

Amikor a számítógépet bekapcsoljuk, akkor az aktiválja az Adatterminál kész (Data Terminal Ready) jelet. Amikor a modem bekapcsolódik, akkor a Modem kész (Data Set Ready) jelet aktiválja. Ha a modem vivőjelet érzékel a telefonvonalon, akkor a Vivőérzékelés (Carrier Detect) jelet aktiválja. Az Adáskérés (Request to Send) jel azt jelzi, hogy a terminál adatot akar küldeni. Az Adásra kész (Clear to Send) jel azt jelzi, hogy a modem felkészült az adatok fogadására. Az adatok elküldése az Adatátadás (Transmit) áramkörön, míg az adatok vétele az Adatvétel (Receive) áramkörön történik. A többi áramkör funkciója többek közt a sebesség kiválasztása, a modem tesztelése, az adatok ütemezése, a csengető jelek érzékelése

vagy adatok ellentétes irányú továbbítása egy másodlagos csatornán. Ezeket a funkciókat a gyakorlatban szinte sosem használják.

Ha két gépet szeretnénk RS-232-C segítségével összekötni, úgy azt egy null-modem segítségével tehetjük meg, vagyis az egyik gép adási vonalát a másik gép vételi vonalával kell összekötnünk.

Az RS-422-A már szimmetrikus átvitelt (balanced transmission) valósít meg. Szimmetrikus átvitel esetén minden fontosabb áramkör két vezetékkel rendelkezik, tehát a földelésük nem közös. Az eredmény az lett, hogy az RS-422-A szabvány 2 Mb/s-os adatátvitelt tesz lehetővé egy legfeljebb 60m-es kábelen.

1.3.5. Digitális átvitel

A digitális elektronika és a számítógépek gyors fejlődésének köszönhetően, az iparilag fejlett országokban, a központok közötti nagysebességű trönkökön folyamatosan a digitális átvitelre tértek át (azaz folyamatos jelek helyett 0-kból és 1-ekből álló sorozatok haladnak a vonala-kon), mely jobb az analóg átvitelnél. Először is nagyon kicsi a hibaaránya.

Analóg áramkörök esetén erősítőket használnak a vonalon fellépő csillapítások ellensúlyozására, ami azonban soha nem lehet tökéletes, különösen akkor nem, ha a csillapítás mértéke különböző. Mivel a hiba halmozódik, ezért a sok erősítőn átmenő jelek várhatóan komolyan torzulnak.

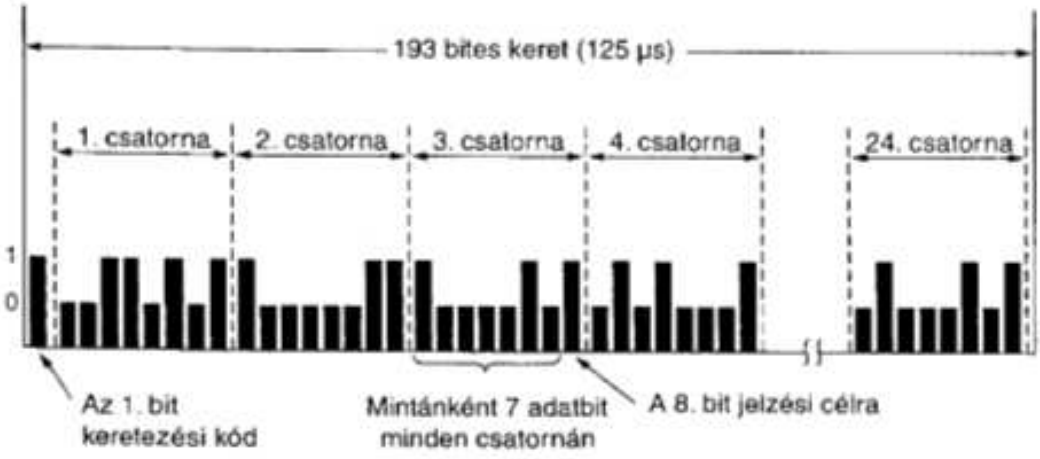
Ezzel szemben a digitális jelek tökéletesen helyreállíthatók, hiszen két lehetséges értékük van, az 1 és a 0. A digitális jelek helyreállításakor nem lép fel halmozódó hiba. A digitális átvitel egy másik nagy előnye az, hogy egyetlen eszköz hatékonyabb kihasználását megengedve, különböző típusú adatok (hang, zene, kép stb.) multiplexált (kevert) átvitelét teszi lehetővé. További előny az, hogy a már meglevő vonalakon is nagyobb átviteli sebesség érhető el.

1.3.5.1. PCM (Impulzuskód-moduláció)

Az analóg jeleket a helyi központban található speciális berendezés, a *kodek (kódoló-dekódoló)* segítségével digitalizálják. A kodek 7 vagy 8 bites számot állít elő. A kodek másodpercenként 8000 mintát vesz az analóg jelből (125 /minta), mivel a Nyquist-tétel értelmében ennyi kell ahhoz, hogy minden információt ki-

nyerjünk egy 4 kHz sávszélességű telefoncsatornából. Kisebb mintavételi frekvencia esetén elveszne információ, viszont nagyobb mintavételi frekvencia esetén sem jutnánk több információhoz. Ezt az eljárást impulzuskód-modulációnak (Pulse Code Modulatian, PCM) nevezik.

A mai távbeszélőrendszereknek a PCM a lelke. Ebbó1 következik az, hogy a távbeszélő-rendszerekben látszólag minden időköz a 125 s többszöröse. Az Egyesült Államokban és Japánban használt T1 vivőt a 1.24. ábrán láthatjuk. A T1 vivő 24 hangcsatornát multiplexel egybe. Az analóg jeleket általában előbb körforgó prioritással mintavételezik, és aztán küldik az analóg bitfolyamot a kodek bemenetére, és nem pedig 24 különböző kodek kimenetét multiplexelik egyetlen digitális kimenetre. A 24 csatorna mindegyike 8 bitet továbbít a kimeneti adatfolyamban. Ebből hét bit adatbit, egy bit pedig vezérlésre szolgál, így csatornánként összesen 7 x 8000 = 56 000 bit adatot és 1 x 8000 = 8000 bit vezérlő információt lehet másodpercenként továbbítani. A keret 24 x 8 = 192 bitből, plusz egy keretszinkronizáláshoz használt bitből áll, tehát 193 bitet kell továbbítani 125 s-onként. Ez tulajdonképpen végeredményben 1,544 Mb/s-os adatátviteli sebességet jelent.



1.24. ábra. A T1 vivő (1,544 Mb/s)

1.3.5.2. Kódolási rendszerek

A *különbségi impulzuskód-moduláció* (differential impulse code modulation) olyan eljárás, amelynél a kimenetre nem a digitalizált jel amplitúdója kerül, hanem az

aktuális jel és az azt megelőző jel amplitúdójának különbsége. Mivel egy 128 kvantálási szinttel rendelkező skálán a 16 szintes vagy annál nagyobb ugrások nem túl valószínűek, ezért 7 bit helyett 5 is elegendő. Ha a jel alkalmanként mégis nagyobbat ugrik, akkor a kódoló logika csak több mintavételi periódus után éri utol a jelet. Beszéd esetén az ilyen hibák elhanyagolhatók.

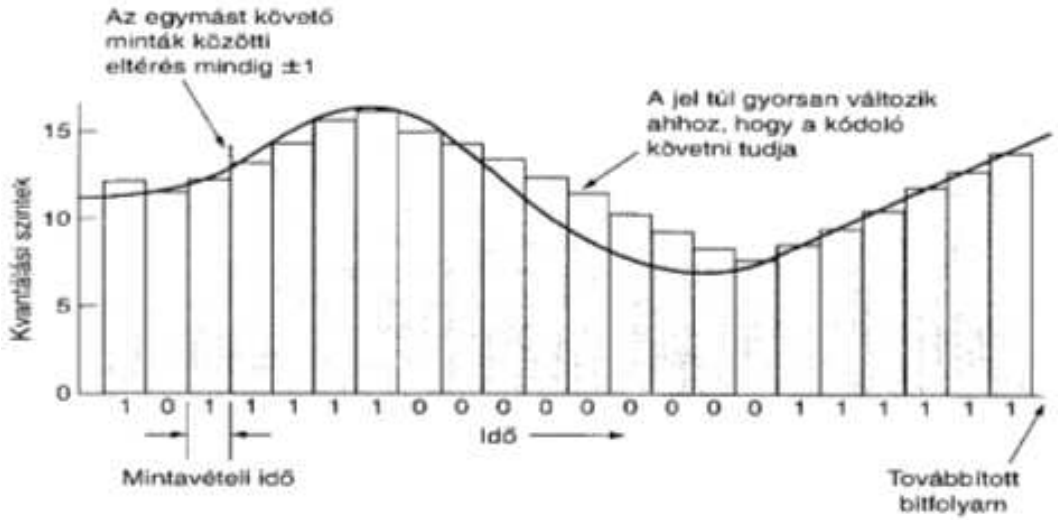
Ennek a tömörítési eljárásnak van egy olyan változata, amelynél az egymást követő minták között a különbség vagy +1 vagy -1 lehet. Ilyenkor elegendő egyetlen bitet elküldeni, amely azt mondja meg, hogy az új minta az előzőnél nagyobb-e, vagy annál kisebb. Ez az eljárás a 1.25. ábrán látható *deltamoduláció* (delta modulation). Hasonlóan az összes többi olyan tömörítési eljáráshoz, amely az egymást követő jelek közötti kis eltérésen alapul, a deltamoduláció esetében is gondot okoz az, hogyha a jel túl gyorsan változik, miként ezt a 1.25. ábrán is láthatjuk. Ilyen esetekben információ vesztés történik.

A különbségi impulzuskód-moduláció egyik továbbfejlesztett változata a korábbi minták értékét extrapolálva megjósolja a következő minta értékét, és a valódi érték, valamint a megjósolt érték közötti eltérést kódolja. Természetesen az adó és a vevő oldalnak ugyanazt a jóslási algoritmust kell használnia. Az ilyen kódolási eljárásokat *prediktív kódolásnak* (predictive encoding) nevezik, és azért hasznosak, mert lecsökkentik a kódolandó számok nagyságát, és ennek köszönhetően kevesebb bitet kell továbbítani.

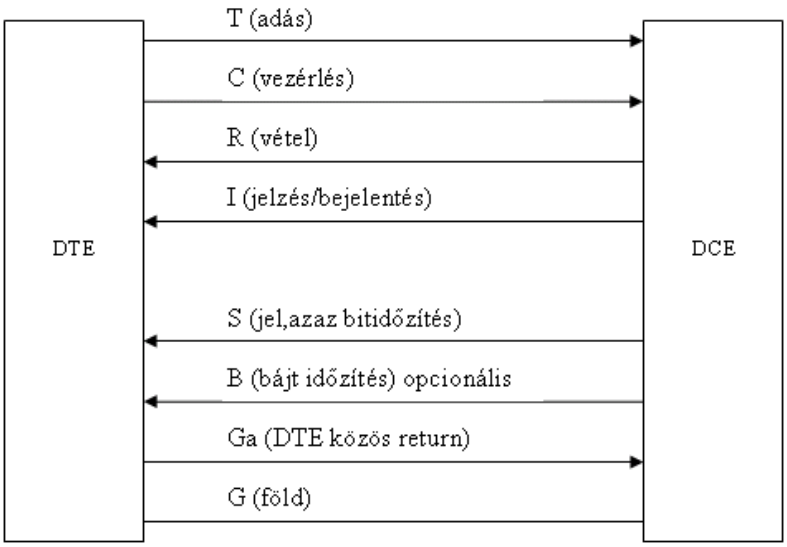
1.3.5.3. Az X.21 digitális interfész

A CCITT már 1969-ben felismerte, hogy a szolgáltatók a jövőben igazi digitális vonalakat fognak telepíteni. Egységes egymással működni képes használatuk bátorítására a CCITT egy digitális interfészajánlást adott ki 1976-ban, az X.21-et. Ez az ajánlás a felhasználói számítógép, a DTE, és a közszolgáltatói készülék, a DCE közötti hívásokat, valamint az azok kiadásához és törléséhez szükséges jelcseréket specifikálja.

Az X.21 által definiált 8 vezeték elnevezését és funkcióját a 1.26. ábra tartalmazza. A csatlakozónak 15 tűje van, de nincs mindegyik kihasználva. A DTE a T és C vonalakat használja az adat- és vezérlőinformációk cseréjére. A DCE az R és az I vonalakat használja az adatok, ill. a vezérlőadatok számára. Az S vona-



1.25. ábra. Deltamoduláció



1.26. ábra. Az X.21-ben használt vezetékek

lon a DCE olyan időzítési információkat tartalmazó bitfolyamot küld a DCE-nek, amelyből az, az egyes bitintervallumok kezdetét és végét állapíthatja meg. A szolgáltató választásától függően létezhet egy B vonal is, amely bitek 8-bites keretekké való formálására használható. Ha ez az opció él, akkor a DTE-nek és a DCE-nek minden karaktert egy kerethatáron kell kezdenie. Ha az opció nem él, akkor a DTE-nek és a DCE-nek is minden vezérlőszekvenciát legalább két SYN-karakterrel

Lépés	C	I	Esemény a telefon analógiában	DTE küldi a T-n	DCE küldi az R-n
0	Off	Off	Nincs összeköttetés a vonal csendes	T = 1	R = 1
1	On	Off	A DTE felveszi a kagylót	T = 0	
2	On	Off	A DTE búgó hangot ad		R = + + ...+
3	On	Off	A DTE tárcsázza a számokat	T = cím	
4	On	Off	Cseng a távoli telefon		R = hívás folyamata
5	On	On	Felveszik a távoli telefont		R = 1
6	On	On	Beszélgetés	T = adat	R = adat
7	Off	On	A DTE elköszön	T = 0	
8	Off	Off	A DCE elköszön		R = 0
9	Off	Off	A DCE leteszi		R = 1
10	Off	Off	A DTE leteszi	T = 1	

1.2. táblázat. Példa az X.21 használatára

kell kezdenie, azért, hogy a másik fél felismerhesse az aktuális kerethatárokat. A tény azonban az, hogy a vezérlőszekvenciák előtt akkor is el kell küldeni a két SYN-t, ha érvényben van a byte-időzítési opció, ugyanis csak így tartható fenn a kompatibilitás azokkal a hálózatokkal, amelyekben nem létezik ez az opció. A 1.2. táblázatban látható egyszerű példában bemutatjuk, hogy egy DTE hogyan ad fe1 egy hívást egy távoli DCE-nek, és hogyan törli azt, amikor befejezte. A hívási és törlési eljárásokat egy közönséges telefonbeszélgetés analógiáján mutatjuk be.

Amikor a vonal tétlen (azaz nincs rajta hívás), akkor mind a négy vonal 1-be van állítva. A C és I vonalakra való hivatkozásnál a CCITT gyakorlatát követve a logikai egyet OFF-nak, a nullát ON-nak fogjuk nevezni. Ha a DTE hívást kíván feladni, akkor T-t 0-ba és C-t ON-ba helyezi, ami megfelel annak, mint amikor egy személy felveszi a kagylót, hogy felhívjon valakit. Amikor a DCE hívásfogadásra kész, akkor ASCII "+" karaktereket kezd el küldeni az R vonalon, jelezve ezzel a DTE-nek, hogy elkezdheti a "tárcsázást". A DTE számok "tárcsázását" a távoli DTE címének bitenkénti elküldésével végzi. A címet ASCII karakterek sorozataként bitenként a T vonalon küldi el. Ezután a DCE ún. hívás-folyamatban jeleket (call progress signals) küld a DTE-nek, hogy informálja a hívás eredményéről. Ha a hívás sikerült, akkor a DCE ON-ba állítja az I-t, ezzel jelzi, hogy elkezdődhet az adatátvitel. Ezen a ponton kialakult már a duplex összeköttetés, és mindkét

oldal szabadon küldhet. C vonalának OFF-ba állításával bármelyik DTE kezdeményezheti a kapcsolatbontást. Miután ezt megtette, több adatot már ő maga nem küldhet, de venni még vehet mindaddig, amíg a túloldali DTE küld neki. A 3.21. ábra 7. lépésében a kezdeményező DTE köszön el először. Helyi DCE-je ezt I vonalának OFF-ba állításával nyugtázza. Amikor a távoli DTE szintén OFF-ba állítja a C vonalát, akkor a kezdeményező oldal DCE-je az R vonalat 1-be teszi. Végül a DTE nyugtázásként T-t 1-be helyezi, majd az interfész új hívásra várakozva tétlen állapotba kerül.

Bejövő híváskor a kimenő hívással analóg eljárás zajlik. Ha egyszerre fordulna elő egy kimenő és egy bemenő hívás, akkor a bemenő hívás törlődik, és csak a kimenő hívás jut érvényre. A jelenséget egyébként hívás ütközésnek (call collision) nevezik. A CCITT ezt a döntést azért hozta, mert az ütközés időpontjában lehetnek már olyan DTE-k, amelyek addigra már kiosztották (allokálták) az erőforrásokat a kimenő hívás számára, így azok újrakiosztásához már túl késő van.

1.3.6. Frekvenciaosztásos és időosztásos multiplexelés

A telefontársaságok olyan módszereket fejlesztettek ki, amelyek segítségével egyetlen fizikai csatornán egyszerre több beszélgetés is lebonyolítható. Ezen multiplexáló módszereket FDM-nek (Frequency Division Multiplexing - *frekvenciaosztásos multiplexelés*) és TDM-nek (Time Division Multiplexin - *időosztásos multiplexelés*) nevezzük.

Az FDM-ben a frekvenciaspektrum több logikai csatorna között van kiosztva, és az egyes felhasználóknak ezekhez a frekvenciasávokhoz kizárólagos hozzáférési joga van.

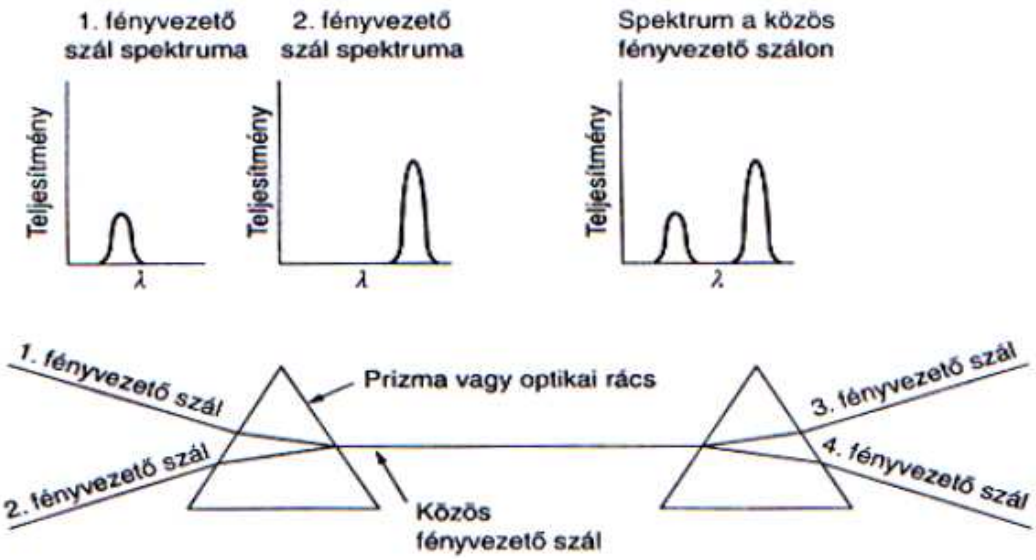
A TDM-ben a felhasználók periódikusan, időben egymás után kerülnek sorra ciklikusan, amelynek során rövid időre a teljes sávszélességgel rendelkeznek.

Az AM rádiós műsorszórás mindkét fajta multiplexelést használja, az FDM módszerre pedig a hangátviteli telefoncsatorna példáját említhetjük. A világon használt FDM módszerek valamennyire szabványosítottak. A jelenlegi távbeszélőhálózatot emberek, nem pedig számítógépek közötti kommunikáció lebonyolítására hozták létre. A TDM, vagy FDM technikát használják, melyek nem alkalmasak

adatforgalom közvetítésére,ezért más jellegű kapcsolási technikára van szükség (két alapvető megoldás a vonal- és csomagkapcsolási technika).

1.3.6.1. Hullámosztásos multiplexelés

(Wavelength Division Multiplexing, WDM) Fényvezető szálaknál alkalmazzák a frekvenciaosztásos multiplexelés ezen speciális változatát. Fényvezető szálakkal az FDM eljárás megvalósításának legegyszerűbb módja a 1.27. ábrán látható. Itt két fényvezető szál csatlakozik egy prizmához. Mindkét szál más frekvenciában továbbítja a fényenergiát. A két sugár áthalad egy prizmán és ezt követően együtt haladnak tovább a célpontig, ahol újból szétválasztják őket. Mivel a rendszer teljesen passzív, így rendkívül megbízható.

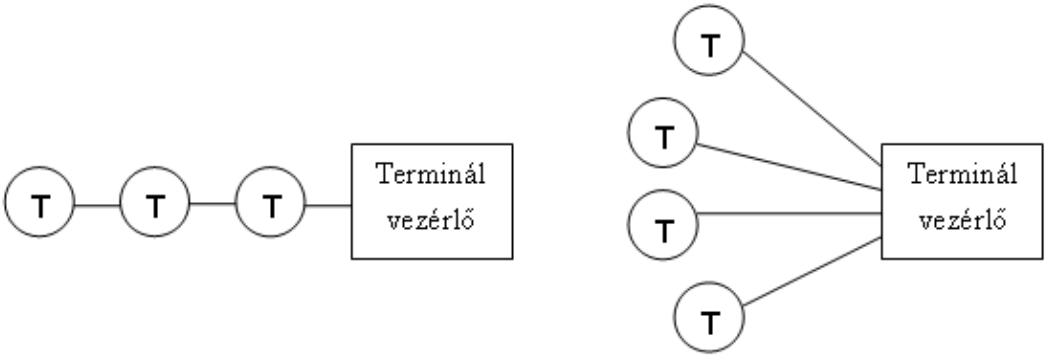


1.27. ábra. Hullámosztásos multiplexelés

1.3.7. Terminálok kezelése

Sok alkalmazás esetén a kommunikációs vonalak ára meghaladja a vonalhoz kötött berendezésnek az árát. A kommunikációs költségek csökkentése érdekében sok hálózat módot ad arra, hogy egyetlen kommunikációs vonalon több terminál

osztozhasson. Az elvi modell a 1.28. ábrán látható, amelyben egy terminál vezérlő (terminal controller) a hozzá kapcsolódó terminálcsoporttól érkező adatokat egyetlen vonalon továbbítja, valamint fordítva: a vonalról érkező adatokat a terminálcsoport felé küldi tovább.



1.28. ábra. Terminálvezérlő. a) Többpontos vonalat használó; b) Kétpontos vonalakat használó

1.3.7.1. Lekérdezés

A lekérdezés részletei a többpontú és a kétpontos (csillag-) vezérlő esetén különböznek. Két lekérdezési módszer ismeretes.

Az elsőben, amelyet *körbekérdezésnek* (roll-call polling) neveznek, a vezérlő sorban egymás után üzenetet küld minden terminálnak, amelyben megkérdezi, hogy az adott terminálnak van-e mondanivalója. A lekérdező üzenet egy helyszíncímet (site address) vagy állomáscímet (station address) tartalmaz, amely a megcímzett terminált azonosítja. Minden terminál ismeri a saját címét, és bár minden lekérdező üzenetet vesznek, de csak a nekik szólókra válaszolnak. Ha a lekérdezett terminálnak van elküldeni való adata, akkor elküldi. Ha nem, akkor egy speciális "lekérdezés-visszautasítás" (poll reject) üzenetet küld válaszul. A vezérlők rendszerint ciklikusan végzik a lekérdezést, de egyes esetekben fontos lehet, hogy egyes terminálok egy ciklus alatt többször is szóhoz juthassanak.

Fél-duplex vonalon terminálonként két vonallekérdezésre van szükség, az első a vezérlő adását, a második a terminál adását engedélyezi. Mivel a mindkét irányú vonallekérdezés ideje - a visszhangelnyomók irányváltási idejét is beleszámolva -

SYN	SYN	SOH	Fejrész	STX	Adat	ETB vagy ETX

SYN = SYNchronize (szinkronizál)
SOH = Start Of Header (fejrész kezdete)
STX = Start Of Text (szöveg kezdete)
ETB = End of Transmission Block (átviteli blokk kezdete)
ETX = End of Text (szöveg vége)

1.3. táblázat. BISYNC üzenetformátuma

akár több száz milliszekundum is lehet, ezért az összes terminál körüljárási ideje már túl hosszú időt igényelhet, még akkor is, ha a legtöbb terminál tétlen.

E probléma megkerülésére találták ki a másik lekérdezési módszert, az ún. *központ felé haladó* lekérdezést (hub polling). Ebben a módszerben a vezérlő a legtávolabbi terminált kérdezi le. A megcímzett terminál megfordítja a vonali átvitel irányát. Ha van elküldeni való adata, elküldi a vezérlőnek, ha nincs, akkor a szomszédjának egy lekérdező üzenetet küld. Ha ez a terminál szintén tétlen, akkor ez is lekérdező üzenetet küld (a vezérlő felé eső) szomszédjának. A lekérdezés így terminálról terminálra halad előre addig, amíg a lekérdezés visszaér a vezérlőhöz. E módszer előnye az, hogy a tétlen terminálok nem okoznak késleltetést, hiszen a felfedezésükhöz - a körbekérdezéssel ellentétben - nem kell a vonal irányát folyamatosan cserélgetni.

1.3.7.2. BISYNC protokoll

Az IBM által kifejlesztett BISYNC (BInary SYNchronous Communication - bináris szinkron kommunikáció) protokollt széles körben használják a számítástechnikai iparban távoli terminálok lekérdezésére valamint egyéb alkalmazásokra is. Félduplex vonalakhoz fejlesztették ki, de egyaránt működik többpontos és kétpontos típusú vezérlők esetén is. A BISYNC üzenetformátumát a 1.3. ábrán láthatjuk. A fejrész mezőinek tartalma az aktuális hálózattól függ, a protokoll nem definiálja azokat. Az ETB több egymást követő blokk esetén egy blokk lezárását jelenti. Az ETX az utolsó blokkot zárja le. Többpontos vonalon levő terminálok megcímzését nem egy fejrészben elhelyezkedő cím, hanem egy külön vezérlőüzenet végzi.

Amikor ASCII kódot használnak, akkor a paritásbit beállításra kerül, és az ellenőrzőösszeg csak egy egyszerű vertikális paritásellenőrzést fejez ki. Az EBCDIC és a 6-bites Transpac esetén az egyes karakterek nem paritásellenőrzöttek, helyette viszont ciklikus redundancia vizsgálattal végzik a hibaészlelést.

1.3.7.3. ISDN

Az ISDN integrált szolgáltatást nyújtó digitális hálózat (Integrated Services Digital Network, ISDN) egy analóg telefonrendszerek digitálissá tételére alkalmas rendszer, amely alkalmas mind hang és nem hang jellegű adatforgalomra is.

Az OSI modell terminológiájával élve, az ISDN otthonokból és hivatalokból származó fizikai rétegbeli bitfolyamot állít elő, amelyre a 2-7. szintű rétegek építhetők. Az ISDN alapja egy ún. digitális bitcső (digital bit pipe), ami nem más, mint egy képzeletbeli cső a szolgáltató és az ügyfél között. A csövön mindkét irányba, több független csatornán áramlanak a bitek. A rendszer kiépítése során a szolgáltató az ügyfél épületében elhelyez egy NT1 (Network Termination 1) hálózati végberendezést, és egy csavart érpár segítségével hozzákapcsolja a több kilométer távolságra lévő szolgáltatói ISDN központhoz. Az NT1-ből kilépő kábelre akár nyolc ISDN telefon, terminál, riasztó és egyéb eszköz fűzhető fel.

Az üzleti vállalkozásoknak gyakran nagyobb sávszélességre van szüksége, így gyakran egy NT2 (Network Termination 2) jelű alközpontot csatlakoztatnak az NT1-hez. Mivel az ISDN 64 kbit/s-os csatornákon alapul, ezért keskenysávú ISDN (Narrow-band ISDN, N-ISDN) névvel illetjük, és így különböztetjük meg a szélessávú ISDN (Broadband ISDN, B-ISDN)-től (ez az ATM).

1.4. A közeghozzáférési alréteg

A hálózatokat két kategóriába sorolhatjuk:

- *két pont közötti* összeköttetések és
- *üzenetszórásos* csatornák.

Üzenetszórásos hálózatban a közös használatú csatorna hozzáférési jogáért folyik a küzdelem. A leglényegesebb kérdés, hogy a közös használatú csatorna használati jogát ki kapja meg.
Példa: Képzeljünk el egy telefonkonferenciát, amelyben hat ember hat különböző telefon segítségével van összekötve úgy, hogy mindenki hallhat mindenkit, és mindenki beszélhet mindenkivel. Nagyon valószínű, hogy amikor közülük az egyik befejezi a mondandóját, egyszerre többen akarnak majd beszélni kezdeni, itt a soron következő beszélő meghatározása sokkal nehezebb.

Sok, erre a problémára megoldást kínáló protokoll ismert. Ezek alkotják e fejezet tartalmát. A hálózatokat nagyságuk szerint 3 nagy csoportba oszthatjuk:

1. A LAN (Local Area Network) hálózatok majdnem mindegyike véletlen hozzáférésű csatornát használ, mivel ezek kiterjedése néhány kilométernél nem nagyobb, és egyetlen szervezet tulajdonában vannak.

2. Ezzel szemben a WAN-ok (Wide Area Network) kétpontos kapcsolatokat használnak (kivéve természetesen a műholdas hálózatokat), országhatárokon ívelnek át és több szervezet birtokában vannak.

3. A LAN-ok és a WAN-ok között helyezkednek el a MAN-ok (Metropolitan Area Network).

1.4.1. Csatornakiosztás

1.4.1.1. Statikus csatornakiosztás

Osszunk meg egyetlen üzenetszórásos csatornát több egymással versengő felhasználó között.

Egy hagyományos, pl. telefonközpontoknál alkalmazott módszer az FDM (Frequency Division Multiplexing - frekvenciaosztásos multiplexelés): ha N felhasználó van, akkor a sávszélességet N egyenlő részre osztjuk és minden felhasználó pontosan

egy ilyen részt kap. Mivel minden felhasználónak különálló frekvenciasávja van, nincs közöttük interferencia. Ahol csak kevés, és rögzített számú felhasználó van, továbbá mindegyikük nagy terhelést eredményező (pufferelt) forgalmat bonyolít le (pl. közszolgáltatók kapcsolóközpontjai), ott az FDM egyszerű és nagyon jó hatásfokú csatorna kiosztási mechanizmus. Ha azonban az állomások száma nagy és folyamatosan változik, vagy a forgalom nem egyenletes, hanem lökésszerűen változik, akkor az FDM nem a legjobb megoldás.

Egyetlen csatorna statikus alcsatornákra vágása azonban még akkor sem hatékony, ha feltételezzük, hogy valahogyan sikerül a felhasználók számát egy konstans N értéken tartani. Az alapvető probléma az, hogy a tétlen, nem forgalmazó felhasználók sávszélessége elvész, ők nem használják, mások pedig nem férhetnek hozzá. Következésképpen a csatornák többsége az idő nagy részében kihasználatlan.

Szinkron időosztásos multiplexelés ellen ugyanazokat az érveket lehet felhozni, mint az FDM ellen. Minden egyes felhasználóhoz statikusan egy darab N-ed résznyi időrés tartozik. Ha a felhasználó nem használja ki a neki kiosztott időrést, akkor az egyszerűen kárba vész.

Az aszinkron időosztásos multiplexelés (koncentrálás) már használhatóbbnak tűnik, de problémák azért itt is vannak. A hagyományos ATDM séma szerint mindegyik terminál saját (dedikált) koncentrátorporttal rendelkezik. Ha két terminál egyszerre akar karaktereket küldeni, azok a koncentrátor különböző memóriakerületére töltődnek be a sorbaállás és az átvitel érdekében. Még ha az összes terminál küld is egyszerre, akkor sem lesz ütközés a koncentrátorban, ha elegendő memória áll rendelkezésre. Az összes terminál adatvesztés nélkül adhat.

Koordinálatlan, földrajzilag szétszórtan elhelyezkedő felhasználók esetén, akik csak egyetlen közös kommunikációs csatornával rendelkeznek és nincsenek saját portjaik, a két egyszerre bekövetkező adás szükségszerűen ütközni fog. A konfliktus elkerülése érdekében minden felhasználónak egy saját spektrumrészt oszthatnánk ki, de ez éppen az FDM, amelyről a fentiekben beláttuk, hogy impulzusszerű forgalmat előállító felhasználók esetén nem hatékony. Egyetlen hagyományos adatkommunikációs módszer sem használható itt. Telesen új csatorna-kiosztási módszerekre van szükségünk.

1.4.1.2. Dinamikus csatornakiosztás

Először felvázoljuk a kiosztási, allokálási problémát. Ezen a területen végzett munkák öt előfeltételen alapulnak:

1. Állomás modell, amely kimondja, hogy az állomások függetlenek, és a munka állandó sebességen folyik. Feltételezi, hogy minden állomáson csak egy felhasználó vagy program dolgozik úgy, hogy amíg az állomás blokkolódik, addig új keretet nem állítanak elő.
2. Egyetlen csatorna feltételezése. A kommunikációt egyetlen csatornán kell lebonyolítani. Minden állomás ezen adhat, és ezen vehet. A hardvert illetően az állomások egyenlők, de a protokollszoftver prioritást adhat egyeseknek.
3. Ütközés feltételezés. Ha két keretet egyszerre küldenek el, akkor időben átlapolódnak, és összekeveredett jeleket eredményeznek a csatornán. Ezt a jelenséget ütközésnek (collision) nevezik. Az összes állomás érzékelhet ütközést. Egy ütközést szenvedett keretet újra kell küldeni. Az ütközés által keltett hibán kívül más jellegű hiba nem keletkezhet.
4. a. Folyamatos idő. A keretek küldését bármelyik pillanatban el lehet kezdeni. Nincsen központi óra, amely az időt diszkrét intervallumokra osztaná.
   b. Résekre osztott idő. Az idő diszkrét intervallumokra (résekre) van felosztva. Egy keret küldése mindig csak egy ilyen időrés kezdetén kezdődhet el. A rés tartalmazhat 0, 1 vagy több keretet, azaz lehet egy tétlen vagy más néven üres, sikeres, ill. egy ütközéses.
5. a. Csatornafigyelés. Az állomások adás előtt képesek megállapítani, hogy a csatornát már használja-e valaki. Ha az állomások foglaltnak érzékelik a csatornát, akkor egyikük sem kezd el adni addig, amíg a csatorna üres nem lesz.
   b. Nincs csatornafigyelés. Az állomások nem képesek érzékelni a csatorna foglaltságát, mielőtt használni kezdenék. Csak később tudják megállapítani, hogy az átvitel sikeres volt-e vagy sem.

1.4.2. Helyi hálózatok protokolljai

1.4.2.1. Perzisztens és nemperzisztens CSMA

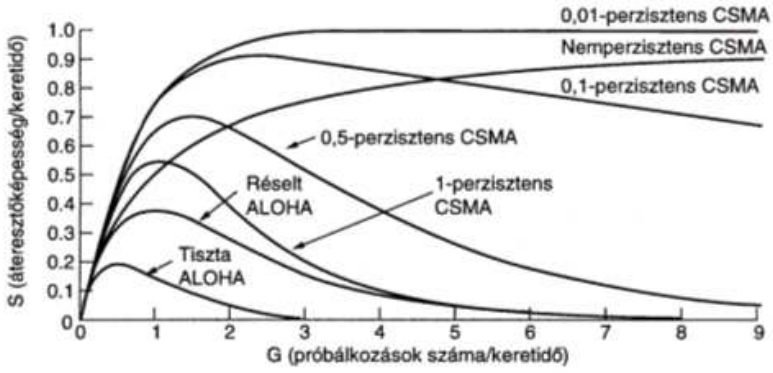
Azokat a protokollokat, amelyekben az állomások figyelik a csatornán áramló jeleket, és ennek megfelelően cselekszenek, csatornafigyelő protokolloknak (carrier sense protocoll) nevezik.

Az első csatornafigyelő protokoll az *1-perzisztens CSMA* (Carrier Sense Multiple Access - csatornafigyelő többszörös hozzáférés).
Amikor egy állomás adni készül, először belehallgat a csatornába azért, hogy eldönthesse folyik-e azon jelenleg átvitel vagy sem. Ha a csatorna foglalt, akkor az állomás addig vár, amíg ismét üres nem lesz. Amikor már üresnek érzékeli, elküldi a keretet. Ha ütközés következik be, akkor az állomás véletlenszerű ideig vár, majd újraadja a keretet. A protokollt azért nevezik l-perzisztensnek, mert egy állomás, amint tétlennek találja a csatornát, azonnal adni kezd, s ennek 1 a valószínűsége. A protokoll teljesítőképességét nagymértékben befolyásolja a terjedési késleltetés. Csak kis esélye van annak, hogy miután egy állomás elkezdte a küldést egy másik állomás, amelyik éppen ezután adásra kész állapotba kerül, érzékelje a csatorna foglaltságát. Ha az első állomás által küldött jel nem éri el a másodikat, akkor ez tétlennek érzékeli a csatornát, szintén elkezd adni, így ütközés áll elő. Minél hosszabb a terjedési késleltetés, annál fontosabbá válik ez a hatás, és annál rosszabb lesz a protokoll teljesítőképessége. Még ha a terjedési idő nulla is lenne, ütközés akkor is előfordulna. Ha a két állomás ugyanis egy harmadik állomás átvitele alatt lesz kész, akkor mindketten udvariasan kivárnák, amíg az átvitel befejeződik, majd pontosan egyszerre kezdenének el adni, ütközést eredményezve. Ha türelmesebbek lennének, akkor kevesebb ütközés keletkezne.

Egy másik csatornafigyelő protokoll a *nemperzisztents CSMA* (nonpersistent CSMA).
Ebben a protokollban már tudatosan arra törekedtek, hogy az állomások kevésbé legyenek mohóak. Abban az esetben, ha van forgalom, az 1-perzisztenssel ellentétben nem figyeli folyamatosan a csatornát, ha a csatorna foglalt, véletlen ideig várakozik, majd elölről kezdi az algoritmust.
Ez a módszer jobb csatorna kihasználtságot eredményez, mint az 1-perzisztens CSMA.

Az utolsó protokoll a *p-perzisztens CSMA* (p-persistent CSMA). Réselt csatornát alkalmaz.
Amikor egy állomás küldésre kész állapotba kerül, elkezdi figyelni a csatornát. Ha a csatorna tétlen, akkor p valószínűséggel elkezd adni, de q=1-p valószínűséggel visszalép az adástól, és megvárja a következő rést. Ha a következő rés szintén tétlen, akkor ismét ad vagy visszalép p ill. q valószínűségekkel. Ez a folyamat addig folytatódik, amíg vagy a keret átvitelre nem kerül, vagy egy másik állomással egyszerre kezd el adni. Az utóbbi esetben ütközés következik be, ekkor az állomás véletlenszerű ideig vár és kezdi elölről az algoritmust. Ha az állomás már kezdetben érzékeli a csatorna foglaltságát, akkor a következő résig vár, és ott kezdi el az előző algoritmust.

Véletlen hozzáférésű protokollok összehasonlítása a terhelés függvényében mért csatornakihasználtság alapján a 1.29. ábrán látható.



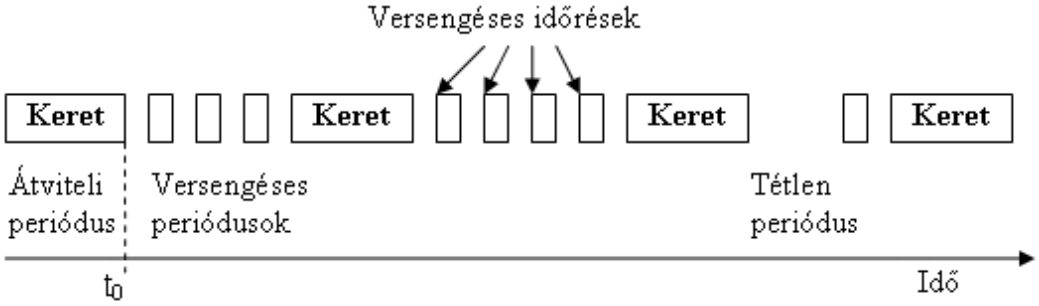
1.29. ábra. Csatornakihasználtság a terhelés függvényében

1.4.2.2. CSMA/CD - CSMA ütközésérzékeléssel

A CSMA/CD a Carrier Sense Multiple Acces with Colosion rövidítése.

A perzisztens és nemperzisztens protokollok egyértelmű előrelépést jelentenek az ALOHA rendszerhez képest, hiszen az állomások nem kezdenek el adni akkor, amikor érzékelik, hogy a csatorna foglalt. További fejlődés az, ha az állomások az ütközés érzékelése után már nem folytatják adásukat. Másképpen fogalmazva, ha két állomás tétlennek érzékelve a csatornát egyszerre kezd el adni, majd nemsokára érzékelik az ütközést, akkor nem fejezi be a keret csatornára küldését, hiszen

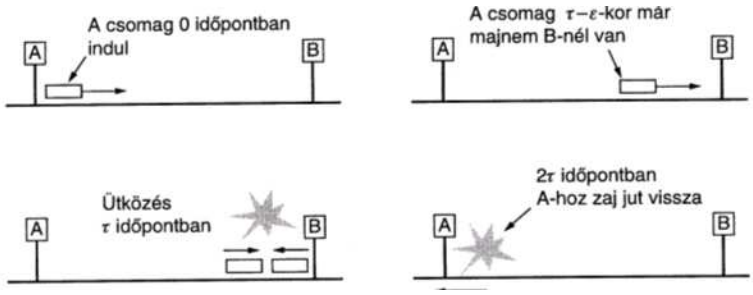
azok már úgyis visszavonhatatlanul megsérültek, hanem az ütközés érzékelését követően azonnal felfüggeszti tevékenységét.
A sérült keretek küldésének abbahagyása időt és sávszélességet takarít meg. Ezt a protokollt CSMA/CD-nek nevezik, és elterjedten használják LAN-ok MAC protokolljaként(1.30). Megjegyezzük, hogy a könyv felépítése az OSI modell rétegszerkezetét követeli. A "Közeghozzáférési alréteg" az ún. MAC (Medium Access Control - közeg-hozzáférési vezérlés) alréteggel foglalkozik. A MAC kifejezéssel a későbbiekben többször találkozni fogunk. A 1.30.ábrán t0-val



1.30. ábra. A CSMA-CD időrései

jelölt pontban egy állomás épp befejezte az adást. Ekkor az elküldendő kerettel rendelkező állomások elkezdenek adni. Amennyiben kettő vagy több állomás is forgalmazni kezd, ütközés jön létre. Ezt érzékelve megszakítják a forgalmazást és véletlen hosszú ideig várnak, majd újra próbálkoznak.
Látható, hogy három állapot váltja egymást, versengési, átviteli és tétlen, amikor senkinek sincs elküldendő kerete. Az ütközések érzékelése érdekében minimalizálták a keretek hosszát, és maximalizálták a kábelhosszt. Erre azért volt szükség, mert az állomás addig figyeli az ütközést, amíg ad. Ha ezalatt az idő alatt nem érzékel ütközést, akkor azt hiszi, hogy rendben elment a keret. Ha ütközés van, akkor az állomás, amelyik hamarabb érzékeli az ütközést, abbahagyja az adást és egy 48 bit hosszúságú zajlöketet (noise burst) küld, hogy a többi állomást figyelmeztesse az ütközésről.
A legrosszabb eset, ha B-hez nagyon közel történik az ütközés és B a legtávolabbi állomás A-hoz képest. Ekkor körülbelül 2τ idő kell A-nak a hiba érzékelésére, ha τ-val a két legtávolabbi állomás közti terjedési időt jelöljük. Ezután véletlenszerű ideig vár, majd újra próbálkozik. De ha a keret rövid, akkor az állomás hamarabb

befejezi az adást, és ezzel együtt az ütközésfigyelést, mielőtt az ütközést érzékelhetné. A fent leírtakat a 1.31. ábrán kísérhetjük figyelemmel. 4.3. ábra. Az



1.31. ábra. Az ütközés szemléltetése

ütközés szemléltetése

1.4.3. Ütközésmentes protokollok

1.4.3.1. AZ IEEE 802 helyi hálózati szabvány

Az IEEE több helyi hálózat szabványt is előállított. A szabványokat részekre osztották, és mindegyiket külön könyvben 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 a CSMA/CD-t,
- a 802.4 a vezérjeles sínt és
- a 802.5 a vezérjeles gyűrű szabványokat írják le

Az utóbbi három LAN szabvány a fizikai réteget és a MAC alréteget definiálja. A következő három szakasz e három rendszert mutatja be.

1.4.3.2. Az IEEE 802.3 szabvány és az ETHERNET

Az IEEE 802.3 szabvány egy 1-perzisztens CSMA/CD LAN-t definiál.
A Xerox, a DEC és az Intel összefogva létrehozta a 10 Mbit/s-os Ethernet szabványt. Ez a szabvány alkotja a 802.3 szabvány alapját is. A publikált 802.3 szabvány egy teljes 1-perzisztens CSMA/CD rendszercsaládot ír le, 1-től 10 Mbit/s-os

sebességig, különböző közegéken működve.
Az adatátvitelre egy 50 ohm-os koaxális kábelt szabványosítottak. Következőkben a 10 Mbit/s-os alapsávú verzióra összpontosítunk. Két különböző típusú koaxiális kábelt használnak elterjedten. Népszerű nevükön a vastag Ethernetet, ill. a vékony Ethernetet. A vastag Ethernet egy sárga kerti locsolócsőre emlékeztet, amelyen 2,5 m-enként a csatlakozási pontot megjelölték. A 802.3 szabvány ténylegesen nem követeli meg, hogy a kábel színe sárga legyen, de ajánlja. A vékony Ethernet vékonyabb és sokkal hajlékonyabb, továbbá megcsapolásos csatlakozás (vámpír-csatoló) helyett szabványos ipari BNC csatlakozókat használ. A vékony Ethernet sokkal olcsóbb, de csak kisebb távolságok áthidalására alkalmas. A két, típus kompatibilis és több módon is összeköthető. Bizonyos korlátozó feltételek mellett sodrott érpár is használható koaxiális kábel helyett.
Bármilyen legyen is a közeg, a szakadt kábelek, a rossz megcsapolások, a laza csatlakozók érzékelése komoly problémát okoz. Alapvetően egy ismert alakú jelet bocsátanak a kábelre. Ha a jel akadályba vagy a kábel végébe ütközik, akkor visszaverődik. A jel kibocsátási és visszaérkezési idejét precízen mérve a visszaverődés keletkezési helye meghatározható. Ezt a technikát időbeli reflektometriának (time domain reflectometry) nevezik.

Az összes 802.3 implementáció, beleértve az Ethernetet is, *Manchester* kódolást használ. A bitek közepén levő átmenet segítségével a küldő szinkronba hozhatja a vevőt. Bármelyik időpontban a kábel a következő három állapot egyikében van: 0-ás bit átvitele (alacsonyból magasba való átmenet), 1-és bit átvitele (magasból alacsonyba való átmenet), vagy tétlen (0 V). A jel magas szintjét +0,85 V, alacsony szintjét -0,85 V jelenti. Az adó-vevő olyan elektronikával rendelkezik, amely csatornafigyelésre és ütközésérzékelésre alkalmas. Ütközést érzékelve az adó-vevő ugyancsak érvénytelen jelet, küld ki a kábelre azért, hogy a többi adó-vevő is biztosan érzékelni tudja az ütközést.

Az adó-vevőt egy adó-vevő kábel (transceiver cable) köti össze a számítógépben levő interfészkártyával. Az adó-vevőkábel legfeljebb 50 méter hosszú lehet, és öt különállóan árnyékolt sodrott érpárt tartalmaz. Ezekből két pár a be- és kimenő adatok számára van kijelölve. További kettő a be- és kimenő vezérlőjelek számára. Az ötödik párral a számítógép árammal láthatja el az adó-vevő elektronikáját.

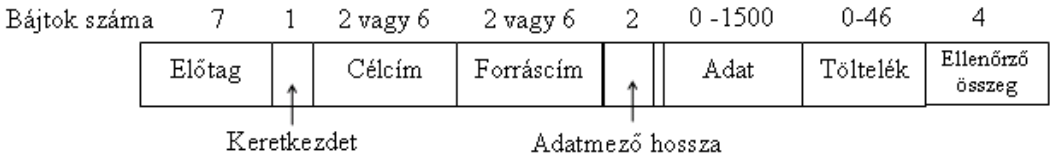
Néhány adó-vevőhöz, az adó-vevők számának csökkentése érdekében, nyolc számítógép kapcsolható egyidejűen.
Az interfészkártya egy vezérlőcsipet tartalmaz, amely kereteket vesz ill. kereteket küld az adó-vevőnek. A vezérlő felelős a kimenő keretek adatokból való összeállításáért, a kimenő keretek ellenőrzőösszegének kiszámításáért és a bejövő keretek ellenőrzőösszegének ellenőrzéséért.

A 802.3 által engedélyezett legnagyobb kábelhossz 500 m. A hálózat által átfogott távolság növelése érdekében az egyes kábeleket ismétlők (repeater) segítségével össze lehet kötni. Mindkét irányból veszi, felerősíti és továbbítja a jeleket. A szoftver, szemszögéből az ismétlőkkel összekötött kábelszegmensek ekvivalensek egyetlen kábellel (eltekintve az ismétlő okozta plusz késleltetéstől). Egy rendszer több szegmenst és több ismétlőt tartalmazhat, de nem lehet két olyan adó-vevő, amely 2,5 km-nél távolabbra helyezkedik el egymástól, ill. nem lehet olyan adó-vevő közötti út, amely négynél több ismétlőn halad keresztül.

A kábelhálózatok építésének egy másik lehetséges módja az, amikor a különálló szegmenshalmazokat hidak (bridge) segítségével kötik össze. A közönséges ismétlőkkel ellentétben, amelyek a biteket azok megvizsgálása nélkül továbbítják, a hidak megvizsgálják a kereteket. A híd ismeri a rendszerben megtalálható gépek címeit, így ha egy csomag akar áthaladni, akkor azt csak abban az esetben engedi tovább, ha a cím abban a résztartományban helyezkedik el. Több részre oszthatja fel a rendszert, így akár több kommunikáció is folyhat párhuzamosan, mert a lezárt részt szabadnak látja a csatornafigyelő és elkezd adni. Így hatékonyabbá válik a rendszer.

1.4.3.2.1. A 802.3 MAC-protokollja.

A 802.3 (IEEE, 1985) keretszerkezetét a következő 1.32. ábrán láthatjuk. 1.



1.32. ábra. A CSMA/CD MAC protokollja

Minden keret egy 7-byte-os előtaggal (preamble) kezdődik, amely 10101010 min-

tájú. Ekkor történik meg az adó és a vevő szinkronizációja.

2. Ezután következik a keretkezdet (start of frame) byte, amely a keret kezdetét jelöli ki az 10101011 mintával.

3. A keret két címet tartalmaz, egy célcímet és egy forráscímet. A célcím legfelső helyértékű bitje közönséges címek esetén 0, csoportcímek esetén 1 értékű. A csoportcímek több állomás egyetlen címmel való megcímzését teszik lehetővé. Amikor egy keretet csoportcímmel küldünk el, akkor a keretet a csoport minden tagja veszi. Az állomások egy meghatározott csoportjának való keretküldést többes küldésnek (multicast) nevezik. A csupa 1-esekből álló cím az üzenetszóráshoz (broadcast) van fenntartva. A célcímben csupa 1-est tartalmazó keretet az összes állomás veszi, és a hidak is automatikusan továbbítják azokat. A legmagasabb helyértékű bit melletti 46. bit a helyi és a globális címeket különbözteti meg. A helyi címeket a hálózatmenedzserek jelölik ki és a helyi hálózaton kívül nincs jelentőségük. A globális címeket ellenben 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 1013 globális cím létezik.

4. A hosszmező (length field) az adatmezőben található adatbyte-ok számát adja meg. A minimum 0, a maximum 1500 byte. Egy keretnek minimum 64 bájtnak kell lennie, így ha az adat rész nulla, akkor a töltelékmezőből (pad) kell a maradék 46 bájtot használni. Pl.: 7 + 1 + 2 + 2 + 2 + 0 + 46 + 4 = 64 (Nulla adat például visszajelzés küldése esetén lehet).

5. Az ellenőrzőösszeg (checksum) mező az adatok 32-bites hasítókódja (hash code). Ha néhány bit (a kábelen keletkező zaj miatt) hibásan érkezik meg, akkor az ellenőrzőösszeg majdnem biztosan rossz lesz, így a hiba felfedezhető. Az ellenőrző összeg algoritmus a CRC-n alapul.

Ütközés esetén az állomás véletlenszerűen 0 vagy 1 időrés után próbálkozik újból. Ha ekkor is ütközés következik be, akkor ez azt jelenti, hogy a két ütközőállomás hasonló időrést választott, így megnövelik a lehetőséget, és a következő próbálkozást 0, 1, 2 vagy 3 időegység múlva kezdik. A választható intervallum 0 és 2n-1 közé esik, ahol n az ütközések száma. n=10 után már nem nő az intervallum, marad 0-1023 közt. Ha n=16 után sem sikerül elküldeni a keretet, akkor a vezérlő hibaüzenetet küld a számítógépnek. A hibajavítást ettől kezdve felsőbb rétegek

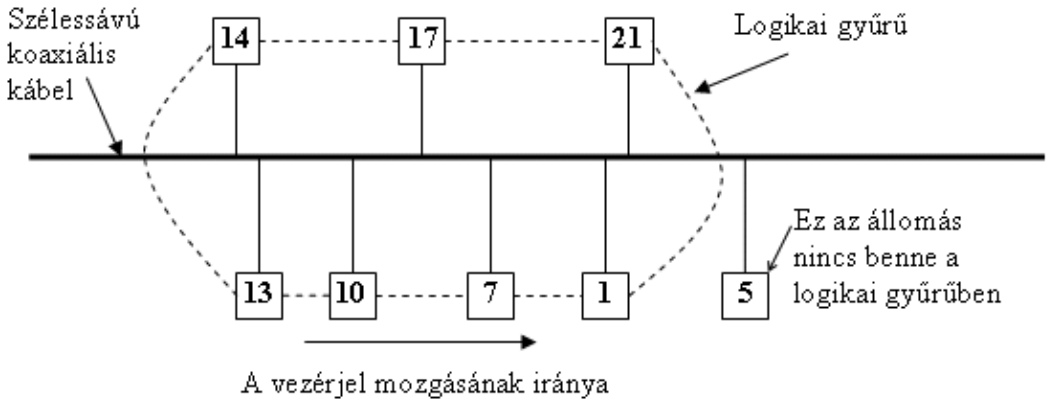
feladata. Az algoritmust kettes exponenciális visszalépésnek (binary exponential backoff) nevezik.

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 ellenőrzőösszeget, és amennyiben ez hibátlan, akkor erről a tényről egy nyugtakeret küldésével értesítenie kell a forrást. Rendes körülmények között ez a nyugtázás egy másik keretet igényelne, amelynek elküldése érdekében meg kellene szereznie a csatorna-hozzáférési jogot. A versengési algoritmus módosításával a sikeres adásokat követő versengési rések közül az elsőt a célállomás kapja.

1.4.3.3. Az IEEE 802.4 vezérjeles sín (token bus) protokoll

A 802.3-as 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 probléma az volt, hogy a 802.3 kereteknek nincs prioritása, ami alkalmatlanná teszi azokat valósidejű rendszerekben való használatra. Ott ugyanis egy fontos keretet nem tarthatnak fel kevésbé fontos keretek.

Fizikailag a vezérjeles sín egy lineáris vagy fa elrendezésű kábel, amelyre számozott állomásokat csatlakoztatnak. Erre példa a 1.33.ábrán látható. A számozásnak csak logikai jelentése van, nem utal az állomás fizikai elhelyezkedésére. Az állomások cím szerint csökkenő sorrendben kerülnek be egy logikai gyűrűbe. A logikailag gyűrűbe szervezet állomások mindegyike ismeri az előtte levő és az utánna következő állomásának a 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 joga csak a vezérjelet aktuálisan birtokló állomásnak van. Mivel egyszerre csak egy állomás birtokolhatja a vezérjelet, ezért ütközés sohasem fordulhat elő. Fontos megértenünk, hogy az állomások fizikai sorrendje lényegtelen. Mivel a kábel egy üzenetszórásos közeg, ezért minden állomás minden keretet vesz, de nem veszi figyelembe azt, amelyik nem neki szól. Amikor



1.33. ábra. Vezérjeles sín

egy állomás továbbadja a vezérjelet, akkor úgy küldi el a közvetlen logikai gyűrűszomszédjának címzett vezérjel-keretet, hogy valójában nem ismeri annak fizikai helyét.
Azt is érdemes megjegyeznünk, hogy amikor az állomásokat bekapcsolják, nem kerülnek azonnal a gyűrűbe (pl. az 5-ös állomás a 1.33. ábrán), az állomások gyűrűbe juttatása, ill. az onnani törlése a MAC-protokoll feladata.

A fizikai réteghez a vezérjeles busz a kábeltelevíziózásban alkalmazott 75Ω-os szélessávú koaxiális kábelt használja. Mind az egy-, mind a kétkábeles rendszer engedélyezett, főállomással vagy anélkül. A lehetséges sebességek: 1, 5 és 10 Mbit/s.

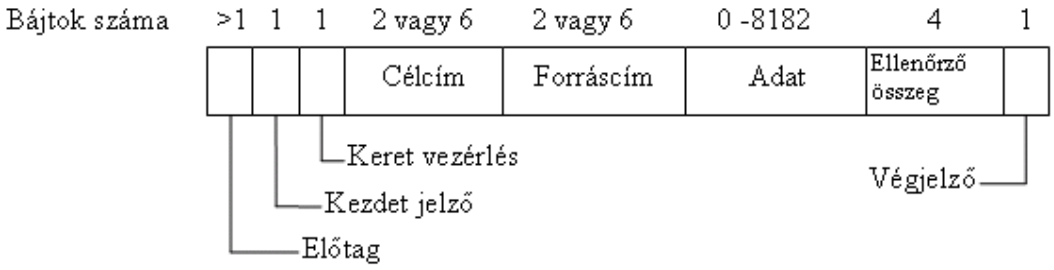
1.4.3.3.1. A vezérjeles sín MAC-protokollja.

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 elküldése 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-ást, 2-est, 4-est és 6-ost, ahol a 0-ás a legalacsonyabb, a 6-os a legmagasabb prioritású. A legkönnyebb ezt úgy elképzelnünk, hogy minden állomás belül négy alállomásra van osztva, 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. 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 rendelkezésére álló időre, akkor az alacsonyabb prioritású alállomások felhasználhatják az így felszabadult részt, azaz az nem vész kárba.

A vezérjeles sín keretformátuma eléggé különbözik a 802.3 keretformátumától (lásd a 1.34. ábrát) Az *előtag*, akárcsak a 802.3-ban a vevő órájának szinkro-



1.34. ábra. A vezérjeles sín keretformátuma

nizálását segíti elő, a különbség csupán annyi, hogy itt megengedett az 1 byte előtaghosszúság is.

A *kezdetjelző* és a *végjelző* mező a keret határait jelzik. Mindkét mező analóg kódolású szimbólumokat tartalmaz, amelyek a digitális 0 és 1 kódolásától jelentősen különböznek. Így kerülik el, hogy ezek a jelek véletlenül a felhasználói adatok között is előfordulhassanak. A határoló jeleknek köszönhetően nincs szükség adathossz mezőre.

A *keretvezérlés*-mező az adat- és a 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. Ezen a jelzés nélkül a célállomás nem küldhetne semmit, hiszen nincs nála a vezérjel! Vezérlőkeretek esetén a keretvezérlés-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 gyűrűbe léptetését vagy azok gyűrűből való kilépését jelző kerettípusokat stb. Vegyük észre, hogy ezzel ellentétben a 802.3 protokoll egyáltalán nem rendelkezett vezérlőkeretekkel. Ott a MAC alréteg csak a keretek kábelre juttatását biztosította, azok tartalmával nem foglalkozott.

A *célcímet* és a *forráscímet* hordozó mező ugyanolyan, mint a 802.3-ban. Akárcsak a 802.3-ban, egy adott hálózatban vagy csak 2-byte-os, vagy csak 6-byte-os címeket használhatnak az állomások, a kevert használat nem megengedett. 2 bájtos címek esetén 8182 bájt, míg 6 bájtos címek esetén legfeljebb 8174 bájt hosszú lehet az adatmező. Megjegyezzük, hogy ez több mint ötször olyan hosszú, mint a legnagyobb 802.3-beli keret. 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 802.3-nál.
A vezérjeles sínen az időzítéseket fel lehet használni "önző" állomások ellen, egyébként viszont nagyon kényelmes hosszú kereteket küldeni akkor, ha valósidejű a követelmény.
Az átviteli hibák kiszűrésére az *ellenőrzőösszeg*-mező szolgál. Ugyanazt az algoritmust használja, és ugyanúgy többtagú, mint a 802.3-nál.

1.4.3.3.2. A logikai gyűrű karbantartása.

A gyűrű karbantartásáért mindig a vezérlőjel birtokosa a felelős. Új állomás felvétele esetén a vezérlőjel birtokosa ajánlatot kér egy keret küldésével a gyűrűben nem szereplő állomásoktól. Ez a keret tartalmazza a küldő és az utána következő címét, és az ebbe a tartományba eső állomások kérhetik a felvételüket a gyűrűbe, így megmarad a cím szerinti csökkenő sorszám.
Ha egy állomás belépést kér, akkor az belép és beékelődik a vezérjelet birtokló állomás és az azt követő közé. Ha több állomás kéri egy időben a felvételét, akkor ütközés következik be, és egy most nem részletezett algoritmus alapján történik a felvételük.űű Kijelentkezés esetén az állomás egy olyan üzenetet küld az őt megelőző állomásnak, amelyben tudatja, hogy ezentúl nem ő következik, hanem a sorban következő állomás. Ezek után abbahagyja a forgalmazást.

A gyűrű felépítése, ha még nem létezett, úgy történik, hogy egy állomás érzékeli, hogy bizonyos ideje már nem volt forgalom. Ezután küld egy üzenetet, amire nem kap választ, mert senki nincs a rendszerben. Így létrehozhat egy vezérjelet, és felépíti a gyűrűt, amelynek jelenleg ő az egyetlen tagja, majd rendszeres időközönként lekérdezi, hogy van-e bejelentkező.
Ha felépült a gyűrű, figyelni kell a vezérjelet, nehogy elvesszen. Ezt a vezérjel birtokosa úgy ellenőrzi, hogy miután továbbküldte a jelet, figyeli a következő ál-

00000000	Claim_token	Vezérjel igénylés a gyűrű indításakor
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 több belépő esetén
00001000	Token	Vezérjel átadása
00001100	Set_successor	Állomás kilépésének engedélyezése

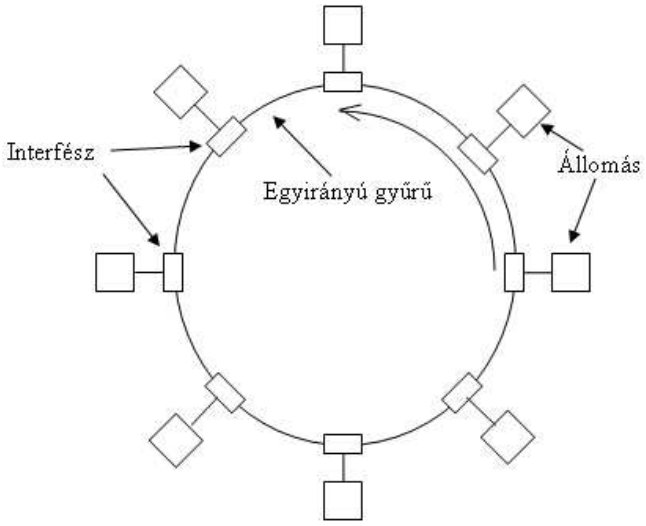
1.4. táblázat. A vazérjeles sín tokenjei

lomást, hogy ad-e ki vezérjelet, vagy keretet, ha nem, akkor újabb jelet küld. Ha ekkor sem jön "válasz", akkor egy általános kérdést küld, él-e még? Amennyiben nem, úgy továbbküldi a vezérjelet a következő állomásnak.
Hasonló a probléma, amikor a vezérjel birtokosa megy tönkre és így veszik el a jel. Ezt a gyűrű beindítási algoritmus oldja meg. Olyan eset is előfordulhat, hogy több vezérjel van jelen a rendszerben. Ekkor az a vezérlőjellel rendelkező állomás, amelyik először észleli, hogy másik vezérjel is van a rendszerben, azonnal eldobja a sajátját, így ismét csak egy vezérjel marad. A vezérjelek részletei a 1.4. táblázatban láthatók.

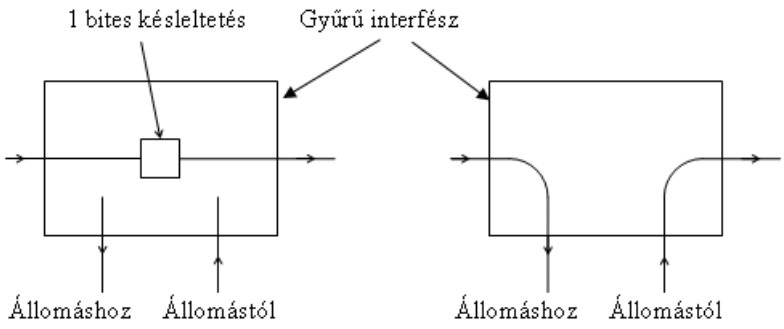
1.4.3.4. IEEE 802.5 szabvány: vezérjeles gyűrű

Ebben a szakaszban először általánosan mutatjuk be a vezérjeles gyűrűket, majd az IEEE 802.5-öt részletesen is. A szakasz végén röviden más típusú gyűrűket is megvizsgálunk. Egy vezérjeles gyűrű felépítését láthatjuk a 1.35.ábrán. A gyűrű valójában két pont közötti kapcsolatokkal összekötött gyűrűinterfészekből áll. Minden gyűrűinterfészhez érkező bit egy ideiglenes pufferbe kerül, ahonnan az adott állomás ismét a gyűrűbe küldi ki. A pufferben levő bitet a gyűrűbe való kiírás előtt az állomás megvizsgálhatja, szükség szerint módosíthatja is. A bitek interfészeknél való pufferelése, másolása 1-bites késleltetést eredményez minden egyes állomásnál. A 1.36. ábrán egy gyűrű interfészeit mutatjuk be.

A gyűrűinterfésznek két üzemmódja van, a vételi és az adási. Az ábra bal oldali része a vételi üzemmódot ábrázolja:
a bemeneti bitek 1-bites késleltetéssel egyszerűen a kimenetre másolódnak. A jobb



1.35. ábra. Gyűrű topológia



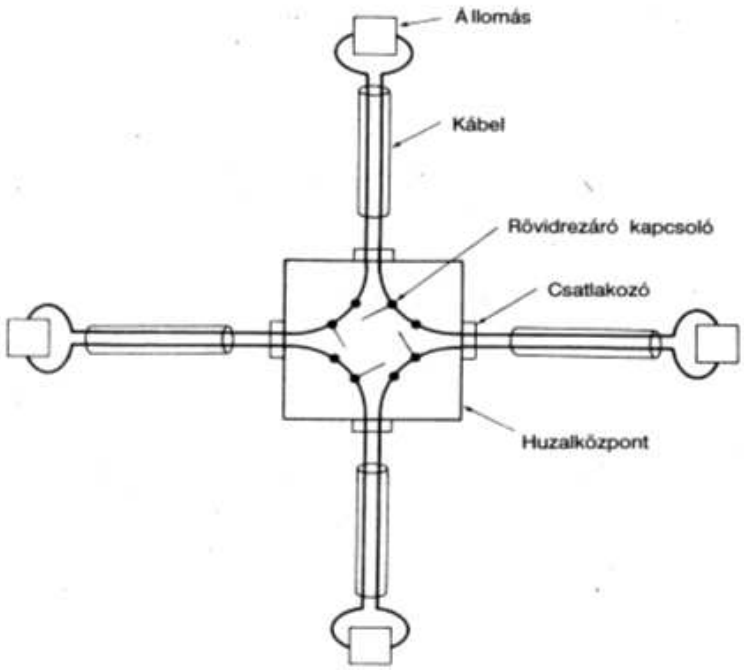
1.36. ábra. A gyűrű interfésze

oldali részen az adási üzemmód látható:
ez a vezérjel megszerzése után következik be, az interfész megszakítja a kapcsolatot a bemenet és a kimenet között, és saját adatait teszi ki a gyűrűre. A gyűrűben így körbejáró biteket a küldő állomásnak kell majd eltávolítania.

A vezérjeles gyűrűben, ha az állomások tétlenek, egy speciális bitminta, az ún. vezérjel (token) jár körbe (lásd később). Amikor egy állomás keretet akar küldeni, még a küldés előtt meg kell szereznie a vezérjelet, és el is kell távolítania a gyűrűből. Mivel csak egyetlen vezérjel van, ezért csak egyetlen állomás adhat, így a csatorna-hozzáférés ugyanúgy oldódik meg, mint vezérjeles sín esetén.

A nyugtázás magától értetődően megoldható a gyűrűn. Ehhez a keretformátumnak egyetlen 1-bites mezőt kell tartalmaznia, amely kezdetben nulla. Amikor a célállomás megkapja a keretet, ezt a mezőt 1-be állítja. Ha a nyugta jelentése az, hogy az ellenőrzőösszeg helyes, akkor ennek a mezőnek az ellenőrzőösszeg után kell következnie, és a gyűrűinterfésznek képesnek kell lennie arra, hogy az utolsó bit beérkezése után az ellenőrzőösszeget azonnal ellenőrizze.
Amikor a forgalom kicsi, a vezérjel a működési idő legnagyobb részében a gyűrűben körbe-körbe fut. Alkalomszerűen egy-egy állomás kivonja a gyűrűből, kereteit elküldi, majd ismét visszahelyezi a gyűrűbe a vezérjelet. Ha azonban a forgalom olyan nagy, hogy az egyes állomásoknál sorok keletkeznek, ilyenkor, ha egy állomás befejezi adását, és a vezérjelet visszahelyezi a gyűrűbe, a következő állomás, figyelve azt, azonnal kivonja a gyűrűből. Ily módon az adási engedély egyenletesen körbehalad a gyűrűben.

A vezérjeles gyűrűk általános ismertetése után fordítsuk figyelmünket a 802.5 szabványra!
A fizikai rétegben a 802.5 1 vagy 4 Mbit/s-os sebességre alkalmas árnyékolt sodrott érpárt használ. A jeleket a *különbségi Manchester*-kódolással kódolják. A magas és alacsony logikai értékeket 3,0-4,5 V közötti pozitív, ill. negatív jelek képviselik. A különbségi Manchester-kódolás magas-alacsony és alacsony-magas váltásokat használ a bitek jelzésére. Minden bitidő közepén van váltás, a 0 jel átvitelénél a bitidő elején is van váltás.
A 802.5 bizonyos vezérlő-byteokban (pl. keretek elejének és végének jelzésére) alacsony-alacsony és magas-magas átmeneteket is használ. Ezek a nem adat jellegű jelek csak egymást követő párokban fordulnak elő azért, hogy ne idézzenek elő egyenfeszültségű komponenst a gyűrűn. A gyűrűhálózatok egyik kritikája az, hogy a kábel bárhol előforduló megszakadása esetén az egész gyűrű használhatatlan lesz. Ez a probléma egyszerűen megoldható egy ún. huzalközponttal (wire center). Míg logikailag az állomások gyűrűt alkotnak, addig fizikailag állomásonként (legalább) két sodrott érpárral a huzalközpontba csatlakoznak. Az egyik érpár az állomások felé, a másik érpár az állomásoktól a központ felé irányuló adatok átvitelét végzi.(1.37.ábra) Bár a 802.5 szabvány formálisan nem követeli meg ennek a gyakran csillag alakú gyűrűnek (star-shaped ring) nevezett (Saltzer és mások, 1983) gyűrűtípusnak a használatát, de a gyakorlatban elvárják, hogy a 802.5 LAN-ok a

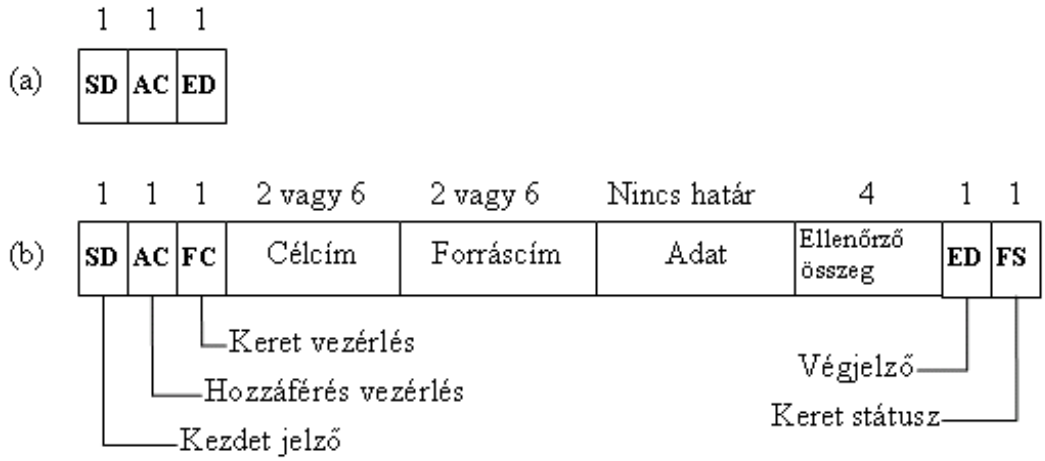


1.37. ábra. Csillag topológiájú gyűrű

megbízhatóság és a karbantarthatóság növelése érdekében huzalközponttal rendelkezzenek.

1.4.3.4.1. A vezérjeles gyűrű MAC-protokollja.

A MAC alréteg alapműködése nagyon egyszerű. Amikor nincs forgalom, akkor a gyűrűn egy 3-byte-os vezérjel kering körbe-körbe addig, amíg valamelyik állomás meg nem szerzi a második byte-ja egy adott, 0 értékű bitjének 1-be állításával. Ezáltal az első két byte keretkezdet szekvenciává alakul át. Az állomás ezután a 1.38. ábrán látható módon egy normál adatkeret további részeit kezdi el küldeni. 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, az állomás további kereteket is elküldhet. Ha az összes keret elküldése befejeződött, vagy a vezérjeltartási idő lejárt, akkor az állomásnak vissza kell állítania a 3-byte-os vezérjelet, és vissza kell helyeznie a gyűrűre.



1.38. ábra. A vezérjeles gyűrű a) vezérjel-formátuma, b)adatkeret-formátuma

Az ábrán látható *kezdetjelző* és *végjelző* mezők a keretek elejét és végét jelzik. Az adatbyte-októl való megkülönböztethetősége érdekében érvénytelen különbségi Manchester mintákat (HH és LL) tartalmaznak.
A *hozzáférési vezérlés* mező tartalmazza a vezérjelet, valamint a figyelőbitet, a prioritásbiteket és a lefoglalásbiteket.
Az adatkereteket a vezérlőkeretektől a *keretvezérlésbyte* különbözteti meg.
Ezeket a *célcím* és a *forráscím* mezők követik, amelyek ugyanazok, mint 802.3-ban és 802.4-ben.
Ezután az *adatmező* következik, amely tetszőleges hosszúságú lehet, hosszát csak a vezérfeltartási idő korlátozza.
Az *ellenőrzőösszeg* mezője megegyezik a 802.3-aséval és 802.4-esével.
A *végjelző* egy E bitet tartalmaz, amelyet akkor billent be egy interfész, ha hibát érzékel (pl. egy nem engedélyezett Manchester-mintát fedez fel). Tartalmaz még egy olyan bitet is, amelynek segítségével egy logikai sorozat utolsó keretét lehet megjelölni, azaz hasonló jellegű, mint egy állományvége (end-of-file) jel.

A 802.5-ös kidolgozott, többszintű prioritáskezelésre alkalmas elrendezéssel rendelkezik. A 3 byte-os vezérjel középső byte-jának egyik mezője a vezérjel prioritását adja meg. Amikor egy állomás egy n prioritású keretet akar küldeni, addig kell várnia, míg egy olyan vezérjelet el nem tud kapni, amelynek prioritása kisebb, vagy egyenlő n-nel. Az állomás a következő vezérjel lefoglalását megkísérelheti

úgy is, hogy az éppen áthaladó keret lefoglalásbitjeit olyan prioritásúvá írja át, amilyen prioritású keretet éppen el szeretne küldeni.

1.4.3.4.2. A vezérjeles gyűrű karbantartása.

Minden gyűrűben van egy felügyelő állomás (monitor station), amely a gyűrű karbantartásáért felelős. A felügyelő felelős:

- a vezérjel-vesztés figyeléséért,
- a gyűrűszakadáskor elvégzendő teendők elvégzéséért,
- az összekeveredett keretek eltávolításáért és
- az árván maradt keretek kiszűréséért.

Árvakeret akkor keletkezik, amikor egy állomás egy rövid keretet elküldött, de kivonására már nem képes, mert időközben meghibásodott, vagy kikapcsolták. Ha ezt nem szűrné ki, akkor a keret a végtelenségig körbe forogna. A keret kiszűrése úgy történik, hogy minden keresztülhaladó keret hozzáférési vezérlés mezőjében bebillenti a *felügyelőbitet*. Ha egy bejövő keretben ez a bit már beállított, 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 halad át a felügyelő-állomáson. A felügyelő-állomás ekkor maga távolítja el ezt a keretet.
A vezérjel-vesztést a felügyelő állomás úgy figyeli, hogy ha a teljes vezérjel idejének lejárta után nem jelenik meg a rendszerben a vezérjel, akkor a felügyelő megtisztítja a gyűrűt, és egy új vezérjelet állít elő.
Az összekeveredett, ill. meghibásodott kereteket érvénytelen formátumuk, vagy helytelen ellenőrzőösszegük révén lehet felismerni. A felügyelő ekkor magán keresztül bocsátva felnyitja, majd megtisztítja gyűrűt, és új vezérjelet bocsát ki.

1.4.3.4.3. Réselt gyűrűk.

A vezérjeles gyűrű nem az egyetlen használt gyűrűtípus. További LAN gyűrűtípus a réselt gyűrű (slotted ring), amelyet azért neveznek így, mert a gyűrű több rögzített méretű keretre van felosztva. Hacsak a gyűrű nem elegendően hosszú, vagy az állomások száma nem nagyon nagy, akkor valószínűtlen, hogy a gyűrűnek elegendő a késleltetése több keret egyidejű keringtetéséhez, ezért mesterséges késleltetéseket kell beépíteni. Az interfészekben léptetőregisztereket alkalmazva ez

könnyen elérhető. Minden keretben van egy jelzőbit, amely a keret feltöltöttségét vagy ürességét mutatja. Amikor egy állomás adni kíván, akkor egyszerűen megvárja, amíg egy üres keret érkezik a gyűrűn, majd telítettnek jelöli meg, és feltölti az adataival. Természetesen az adatoknak bele kell férnie egy keretbe, ellentétben a vezérjeles gyűrűvel, amely tetszőleges hosszúságú kereteket támogat.

1.5. Az adatkapcsolati réteg

Olyan algoritmusokkal foglalkozik, amelyek két szomszédos, adatkapcsolati rétegben levő állomás között megbízható, hatékony kommunikációt létesítenek. A szomszédos alatt azt értjük, hogy a két állomást fizikailag egy kommunikációs csatorna köti össze, amely úgy viselkedik, mint egy huzal. "Huzalszerűvé" egy csatornát az az alapvető tulajdonsága teszi, hogy a bitek pontosan ugyanolyan sorrendben érkeznek meg a célhoz, mint ahogyan elküldték azokat.

1.5.1. Az adatkapcsolati réteg tervezési kérdései

Az adatkapcsolati réteg feladata egy összeállított keret átvitele két csomópont között. Az adatokat a hálózati rétegtől kapja az adatkapcsolati réteg, és az általa összeállított kereteket átadja a fizikai rétegnek, ami bitenként küldi át a fizikai közegen.

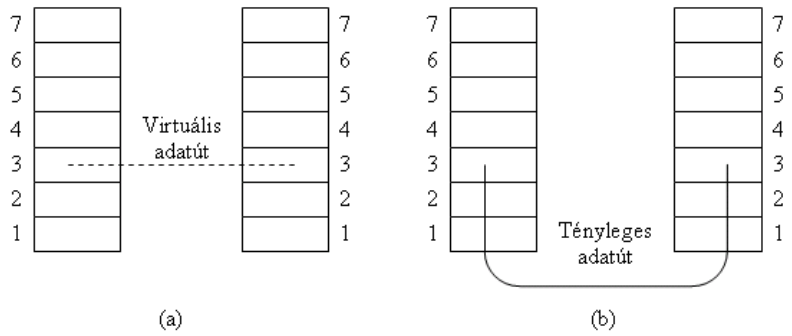
Az adatkapcsolati rétegnek számos funkciót kell megvalósítania:

- jól meghatározott szolgálatinterfészt kell biztosítania a hálózati réteg számára,
- meg kell határoznia a fizikai rétegtől érkező bitek kertté csoportosításának módját,
- foglalkoznia kell az átviteli hibákkal,
- szabályoznia kell a keretek áramlásának sebességét azért, hogy a lassú vevőket ne rasszák el a gyors adók, és végül
- általános kapcsolatmenedzselést is kell végeznie.

1.5.1.1. A hálózati rétegnek biztosított szolgálat

A hálózati réteg az 1.39. ábra (a) része szerint látja az adatkapcsolatot, míg valójában ez a fizikai rétegen keresztül valósul meg, amint azt a 1.39. ábra (b) részén láthatjuk. Az adatkapcsolati réteg különböző szolgálatok ellátására tervezhető. A ténylegesen nyújtott szolgálat rendszerről rendszerre változik. Három ésszerű lehetőség a következő:

1) Nyugtázás nélküli, összeköttetés-mentes szolgálat.
2) Nyugtázott, összeköttetés-mentes szolgálat.
3) Nyugtázott, sszeköttetés-alapú szolgálat.



1.39. ábra. A virtuális és a tényleges adatút

Nyugtázás nélküli, összeköttetés-mentes szolgálat esetén a forrásgép egymástól független kereteket küld a célgépnek úgy, hogy az elküldött keretekre nem vár nyugtát. Előtte nincs összeköttetés-létesítés, utána nincs lebontás. Ha egy keret a vonali zaj miatt elvész, akkor az adatkapcsolati rétegben nem történik kísérlet annak korrigálására. Ez a szolgálati osztály akkor megfelelő, ha a hibaarány nagyon alacsony, és ha a hibás állapotból való kikerülést a felsőbb rétegek kezelik. Szintén alkalmas lehet valósidejű forgalom, pl. beszédátvitel esetén, amikor is a késve érkező adatok rosszabbak, mint a hibásak. Sok LAN adatkapcsolati rétegében ilyen, nyugtázatlan összeköttetés-mentes szolgálattal rendelkezik.

A nyugtázott, összeköttetés-mentes szolgálat sem nyújt összeköttetést, de minden egyes elküldött keretre külön nyugta érkezik. Ily módon a küldő tudja, hogy az általa elküldött keret biztosan megérkezett. Ha egy adott intervallumon belül nem érkezett nyugta, akkor a keret újraküldhető. Ez a szolgálat megbízhatatlan csatornák esetén hasznos. (l: vezeték nélküli rendszerek).

Összeköttetés-alapú szolgálat során a forrás és a célgép az adatok küldése előtt összeköttetést létesítenek egymással. Az összeköttetésen küldött keretek sorszámozottak, és az adatkapcsolati réteg garantálja, hogy az elküldött kereteket valóban meg is kapják a címzettek. Garantálja azt is, hogy a célok a kereteket pontosan egyszer kapják csak meg, és hogy a keretek vételi sorrendje azonos a küldési sorrendjükkel. Ezzel szemben az összeköttetés-mentes szolgálatnál elképzelhető, hogy egy elveszett nyugta az adott keret többszöri elküldését és vételét eredményezi. Az összeköttetés-alapú szolgálat azonban megbízható keretfolyamot (bitfolyamot) nyújt a hálózati rétegbeli folyamatok számára.

Amikor összeköttetés-alapú szolgálatot használunk, akkor az átvitel három különböző fázisból áll:

- Az első fázisban mindkét oldal, a keretek elküldésének nyomkövetésére való változók és számlálók kezdeti értékének beállítása után, létrehozza az összeköttetést.
- A második fázisban zajlik a keretek tényleges átvitele.
- A harmadik és legutolsó fázisban az összeköttetés lebomlik, aminek következtében felszabadulnak a változók, a pufferek, és minden más, az összeköttetés fenntartásához szükséges erőforrás.

A hálózati és az adatkapcsolati réteg közötti kommunikáció szabványos OSI szolgálati primitíveket használ:

kérés, bejelentés, válasz, megerősítés.

A *kérés* primitívvel a hálózati réteg valamilyen tevékenység elvégzésére kéri az adatkapcsolati réteget. Ilyen pl. egy összeköttetés létesítése vagy lebontása vagy keretek küldése.

A *bejelentés* primitív tudatja a hálózati réteggel, hogy valamilyen esemény történt, pl. egy másik gép összeköttetést akar létesíteni vagy lebontani, vagy egy keret érkezett.

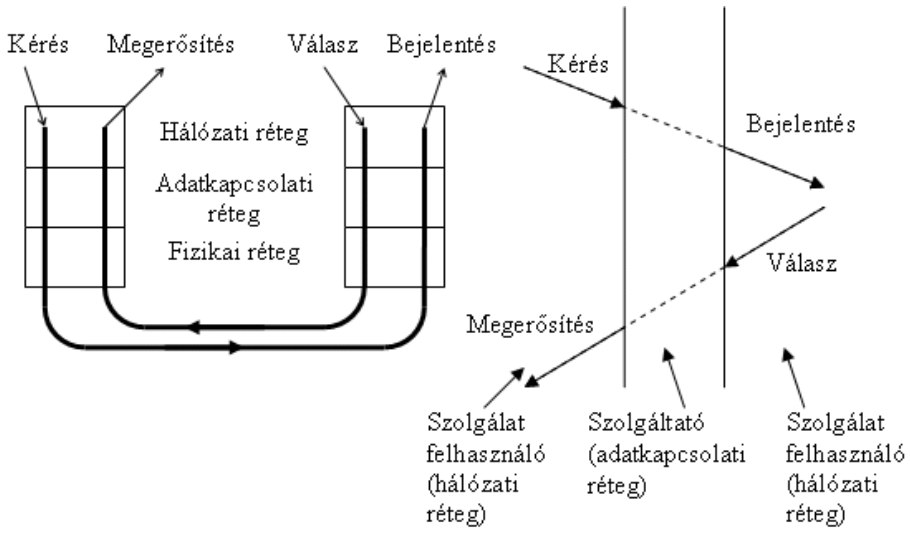
A *válasz* primitívet a vevőoldali hálózati réteg arra használja, hogy az előző bejelentés primitívre válaszoljon.

A *megerősítés* primitív a kérőoldali adatkapcsolati réteg számára megerősítheti a kérés sikeres végrehajtásának tényét, vagy ha nem volt sikeres, akkor annak okát közölheti.

Az alábbi 1.40.ábrán a négy primitívet mutatjuk be két különböző módon. Az ábra bal részén a rétegezés az adatáramlásnak megfelelően hátható. A jobb oldali ábrán a bal oldali függőleges vonaltól balra a küldő oldali adatkapcsolati szolgálat felhasználója helyezkedik el. A két vonal között az adatkapcsolati szolgálat szolgáltatója található. Végül a vonalaktól jobbra az adatkapcsolati réteg vevőoldali felhasználója foglal helyet. Az idő lefelé halad, azaz a feljebb ábrázolt események régebben történtek, mint a lejjebb levők.

1.5.1.1.1. Keretképzés.

Az adatkapcsolati réteg szokásos megközelítése szerint a bitfolyamokat diszkrét keretekké kell tördelni, és mindegyik keretre ellenőrzőösszeget kell képezni.



1.40. ábra. A szolgálat primitívek két különböző ábrázolása

Amikor egy keret megérkezik a célhoz, akkor az ellenőrzőösszeg újra kiszámításra kerül. Ha ez különbözik a keretben levőtől, akkor az adatkapcsolati réteg értesül a hibáról, és megteszi a szükséges lépéseket (pl. eldobja a hibás keretet és hibajelentést küld vissza). Mivel nagyon kockázatos az időzítésre hagyatkozva megjelölni egy keret elejét és végét, ezért más módszerek használata tanácsos. Az e szakaszban vizsgált négy közismert módszer:

- 1. karakterszámlálás
- 2. kezdő- és végkarakterek karakterbeszúrással
- 3. kezdő- és végjelzők bitbeszúrással
- 4. a fizikai réteg kódolásának megsértése

1.5.1.1.2. Karakterszámláló módszer.

A keret fejlécében megadjuk a keretben lévő karakterek számát. Ez a célállomás oldalán meghatározhatóvá teszi a keret végét. A módszer problémája, hogy a karakterszámot tartalmazó mező is megsérülhet az átvitel során.

1.5.1.1.3. Kezdő és végkarakterek alkalmazása karakterbeszúrással.

Az előző módszernél a keret karaktereinek vételénél egy számlálót is folyamatosan kell egyesével csökkenteni (dekrementálni), amely kezdeti értékét is a keretből

töltjük fel. Amikor a számláló értéke nulla, akkor értük el a keret végét. Jobb megoldás az, ha egy speciális karaktersorozattal jelöljük a keret kezdetét és végét. Szokásos megoldás a DLE STX karakterkettőssel jelezni a keret kezdetét és DLE ETX-el a keret végét. Ezek speciális, az ASCII kódtáblában megtalálható karakterek (DLE=16 STX=2 és ETX=3), és keret adatrészében lévő esetleges szövegekben nem fordulnak elő.
Más a helyzet, ha karakteralapú módszerrel bináris adatokat (pl. egy programkódot) kívánunk átvinni. Ilyenkor, mivel bármilyen bináris bitcsoport előfordulhat, az adatmezőben megjelenhet a fenti két karakterkombináció, és ez hibás kerethatárt jelez. A megoldás az, hogy az adó a kert összeállításakor az adatmezőben megjelenő minden DLE kód után, azonnal beszúr még egy DLE karaktert. A vevő pedig, ha a DLE karakter vétele után ismét DLE következik, nem tekinti vezérlő karakternek, és a második DLE-t egyszerűen eldobja. A hálózati réteg által küldött üzenet:

I T T E Z DLE V O L T
Az adó adatkapcsolati réteg keretképzése és karakter beszúrása:
DLE STX
I T T E Z DLE **DLE** V O L T
DLE ETX
Az vevő adatkapcsolati rétege leválasztja a kettőzött beszúrt karaktert:
DLE STX
I T T E Z DLE V O L T
DLE ETX
A vevő hálózati rétegének átadott üzenet:
I T T E Z DLE V O L T.

1.5.1.1.4. Kezdő és végkarakterek bitbeszúrással.

Itt a kezdő és a végjelző bitsorozat egy 01111110 alakú bitsorozat. Az átvitt adatok között nem lehet ilyen sorozat, ezért az adó oldalon, ha legalább 5 db 1 értékű bit jön egymás után az adatok k0z0tt, beszúrásra kerül egy 0 értékű bit. A vevő minden 5 db egymás után érkező 1 értékű bit mögül, ha ott 0 van, kiveszi a beszúrt 0 értékű bitet. Ha itt nem 0 jön, akkor az a kezdet vagy a végjelző bitsorozat. Pl.

Legyen az elküldendő bitsorozat:
0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1 ,
akkor az elküldött bitsorozat a kezdő és végjelző bitekkel együtt a következő lesz:
0 1 1 1 1 1 1 0
0 1 0 1 1 1 1 1 **0** 0 1 1 1 1 1 **0** 1 1 1 0 0 1
0 1 1 1 1 1 1 0

1.5.1.1.5. A fizikai réteg kódolásának megsértése .

Ezt a megoldást a 802.4 és a 802.5 leírásánál már láttuk, előbbi esetben egy a keret kezdet és végjel egy analóg kódolású érték, míg a vezérjeles gyűrűnél egy, a Manchester kódolásban nem használt HH ill. LL érték jelöli a keret kezdetét és a végét.

1.5.1.1.6. Hibavédelem.

A megbízható kézbesítés módja az, hogy a küldő oldal valamilyen visszacsatolás révén értesül a túloldali eseményekről. Tipikusan a protokoll előírja a vevőnek, hogy a hozzá érkező keretekről pozitív vagy negatív nyugtát tartalmazó, speciális vezérlőkereteket küldjön vissza az adónak. Ha a küldő pozitív nyugtát kap egy keretről, akkor tudja, hogy a keret biztosan megérkezett. A negatív nyugta viszont valami hibára hívja fel a figyelmet, és ekkor a keretet újra kell küldeni.
Amikor a küldő kibocsát egy keretet, rendszerint egy időzítő órát is elindít. Az időzítés elegendően hosszú ahhoz, hogy a keret elérje célját, ott feldolgozásra kerülhessen, és egy nyugta még visszaérkezhessen az adóhoz. Normális körülmények között a keret hiba nélkül megy át, és a nyugta még az időzítés lejárta előtt megérkezik a küldő félhez, aki ekkor leállítja az időzítő órát. Ha azonban a keret, vagy a nyugta elvész, akkor az időzítés lejár, ami a küldő figyelmét egy potenciális hibára hívja föl. Nyilvánvaló megoldásnak látszik a keret újraküldése. Ha azonban a keretet többször elküldik, akkor fennáll annak a veszélye, hogy a vevő ugyanazt a keretet többször is veszi, és a hálózati rétegnek többször át is adja. Ennek megakadályozásává általában a kimenő keretekhez sorszámot rendelnek, így a vevő meg tudja különböztetni az eredeti kereteket az újraadottaktól.

1.5.1.1.7. Forgalomszabályozás.

Előfordulhat, hogy egy gyorsabb, kevésbé terhelt gép küld adatot egy lassabb vagy

jobban leterhelt gépnek. Az adó addig küldi nagy sebességgel a kereteket, amíg a vevőt teljesen el nem árasztja és nem lesz képes a keretek további kezelésére, és kezdi elveszteni azokat.

Az adó küldési sebességének vevőhöz való igazítását rendszerint forgalomszabályozás (flow control) bevezetésével oldják meg. Ezt valamilyen visszajelzési mechanizmust igényel azért, hogy az adónak a tudomására jusson, hogy a vevő tudja-e folytatni a tevékenységet. Az erre definiált szabályok megtiltják egy keret elküldését, amíg a vevő engedélyt nem ad rá explicit, vagy implicit módon. Egy kapcsolat felépülése után a vevő megadhatja, hogy n keretet küldhet az adó, aztán álljon le a keretek küldésével, amíg újabb kérést nem kap.

1.5.1.1.8. Kapcsolatmenedzselés.

Menedzselni kell továbbá a kapcsolat konfigurációját is. A legegyszerűbb esetben egyetlen fizikai vonal van a két gép között. Gyakori azonban, hogy több gép osztozik ugyanazon a csatornán. Hagyományosan ezek közül egyik az elsődleges (primary) gép (pl. egy számítógép), míg a többi a másodlagos (secondary) gép (pl. "buta" terminálok). A forgalommenedzselés során az elsődleges állomás egy rövid keretet, egy ún. lekérdezést (poll) küld az első másodlagos állomásnak, hogy van-e elküldendő adata. Ha igen, akkor a terminál elküldi az adatot; ha nem, akkor az elsődleges a következő másodlagos gépet kérdezi le. Más rendszerekben a terminál lekérdezés nélkül is küldhet adatot a számítógépnek. Végül vannak olyan rendszerek is, ilyenek a LAN-ok, amelyekben nincsenek elsődleges, és másodlagos szerepkörű állomások, minden állomásnak egyforma és azonos hozzáférési joga van a csatornához.

1.5.1.2. Hibajelzés és hibajavítás

A telefonvonalon tapasztalt átviteli hibákat különböző fizikai jelenségek okozzák.

Egy jelenség, amely mindig jelen van: a termikus zaj. A rézvezető elektronját nagy sebességgel minden irányba röpdösve széles spektrumú háttérzajt okoznak.

Az adatátviteli zajok egyik legfőbb forrása az impulzuszaj.

A hibák másik fő forrása az, hogy a jelek amplitúdója, terjedési sebessége és fázisa is frekvenciafüggő. A telefonrendszer valójában Fourier-analizálja a jeleket, minden egyes frekvenciakomponenst külön-külön torzít, majd a végén egyesíti azokat.

Még számos más hibaforrás létezik. Két, fizikailag közel levő vezeték között áthallás lehetséges. Amikor a visszhangelnyomók ki vannak kapcsolva, visszhangok keletkeznek.

A mikrohullámú összeköttetésnél gondot okozhat az elhalkulás (fading), de problémát okozhatnak az elvonuló madárrajok is.

Hangátvitel esetén érdemes a jel amplitúdót egy szűk tartományra összenyomni, mivel az erősítők széles tartományban nem lineárisak. Ez az összenyomás, amelyet tömörítésnek (companding) neveznek, hibák forrása lehet.

Végül pedig tökéletes vivőhullámot sem lehet előállítani.

Az első hibakezelést a fizikai rétegben, a bitek és karakterek átvitelénél kell megoldani. A zajok időtartamából következően lehetnek egyedi és csoportos bithibák. A gyakoribb esetben a hibák fennállási ideje általában egy bit átviteli idejének a többszöröse, ezért ezek a hibák csoportosan, hibacsomók formájában jelentkeznek. Mivel az adatátvitel blokkos (keretes) formában történik, ezért az eredmény egy-egy blokk tönkremenetele. A csoportos hibák hátránya, hogy sokkal nehezebb jelezni és kijavítani őket, mint az elkülönült hibákat.

1.5.1.2.1. Hibajavító kódok.

A hálózattervezők két alapvető stratégiát dolgoztak ki a hibák kezelésére.

Az egyik módszerben az adatblokkok elegendő redundanciát tartalmaznak ahhoz, hogy a vevő megállapíthassa, hogy minek kellett volna lennie az átvitt karakternek.

A másik módszer csak annyi redundanciát tartalmaz, hogy a vevő a hiba tényét felfedezhesse, de magát a hibát ne tudja meghatározni. A vevő ekkor újraadást kér.

Az előző stratégia az ún. *hibajavító* kódokat (error correcting codes), míg az utóbbi az ún. *hibajelző* kódokat (error detecting codes) használja.

Általában egy keret m adatbitet (üzenetbitet) és r redundáns bitet (ellenőrzőbitet) tartalmaz. Legyen a teljes hossz n, vagyis

n = m + r.

Az n bites adat- és ellenőrzőbitet tartalmazó egységre gyakran n-bites kódszavakként (code word) hivatkoznak. Két tetszőleges kódszót megadva, mondjuk 10001001 és 10110001 mindig megállapítható, hogy hány bitben különböznek. Ebben az esetben a két szó 3 bitben különbözik egymástól. A különbség megállapítására vegyük a két szó kizáró vagy kapcsolatát, és számoljuk meg, hogy hány 1-es maradt az eredményben!

A két kódszó bitpozíciókban mért távolságát *Hamming-távolságnak* nevezik. Jelentősége az, hogy ha a két kódszó d Hamming-távolságra van egymástól, akkor az egyik a másikba d db egyes hibával konvertálódhat át. A legtöbb adatátviteli alkalmazásban az összes 2^m adatüzenet legális, de az ellenőrzőbitek kiszámításának módja miatt várhatóan nem mind a 2^m lehetséges kódszót használjuk fel. Az ellenőrzőbitek kiszámításának módját ismerve, elkészíthető a legális kódszavak teljes listája, amelyből már kiválasztható az a két kódszó, amelyek Hamming-távolsága a legkisebb. Ez a távolság a teljes kód Hamming-távolsága.

Egy kód hibajavító vagy hibajelző tulajdonsága annak Hamming-távolságától függ.

Ahhoz, hogy d hibát észleljünk d + 1 nagyságú távolság kell, mert egy ilyen kódnál nem fordulhat elő, hogy d számú egybites hiba egy érvényes kódszót egy másik érvényes kódszóvá alakítson át. Amikor a vevő egy érvénytelen kódszót vesz, biztosan állíthatja, hogy hiba volt.

Hasonlóan, d hiba javítására 2d + l távolságra van szükségünk, hiszen ekkor a legális kódszavak olyan távolságra vannak egymástól, hogy még d bitnyi változás után is az eredeti kódszóhoz közelebb állnak, mint bármelyik másik kódszóhoz, így a javítás egyértelműen elvégezhető.

Hibajelző kódra egyszerű példa az a kód, amelyben az adatok végére egyetlen paritásbitet (parity bit) függesztenek. A paritásbit értékét úgy választják meg, hogy az 1-esek száma a kódszóban páros (vagy páratlan) legyen. Ennek a kódnak a Hamming-távolsága 2, mivel minden egyes bithiba rossz paritású kódszót hoz létre. Tehát, csak az egybites hibák észlelésére használható.

Példaként egy egyszerű hibajavító kódra vegyük a következő, mindössze négy érvényes kódszóval rendelkező kódot

000000000, 0000011111, 1111100000, 1111111111

A kód Hamming-távolsága 5, ami azt jelenti, hogy 2 bit hiba javítására képes.
Ha a 0000000111 kódszó érkezik, akkor tudja, hogy az eredeti kódszó a 0000011111 volt.
Ha azonban egy hármas hiba következtében a 0000000000 változik 0000000111-é, akkor a hibát már nem képes helyesen javítani.

Képzeljük el, hogy egy olyan m adatbitből és r ellenőrzőbitből álló kódot akarunk tervezni, amely az összes egyedi hiba kijavítását lehetővé teszi.
Mind a 2^m legális üzenethez külön-külön n illegális kódszó tartozik, amelyek 1 adott távolságra vannak az adott érvényes kódszótól. Úgy képzelhetjük el ezeket, hogy az n bites kódszó bitjeit egyenként invertáljuk. Így tehát a 2^m db üzenet mindegyikéhez n + 1 bitmintát kell hozzárendelni.
Mivel összesen 2^m bitminta van, ezért az
$(n + 1)2^m \leq 2^n$
egyenlőtlenségnek fenn kell állnia. Az n = m + r összefüggést használva ez
$(m + r + 1) \leq 2^r$
alakot veszi fel. Adott m esetén az egyenlőtlenségből kinyerhető az ellenőrzőbitek alsó határa, amely tehát, az egyedi bithibák kijavításához minimálisan szükséges.

Ezt az elméleti határt egy Hammingtől származó módszerrel a gyakorlatban is megkaphatjuk. A kódszavak bitjeit a bal szélső bittől kezdve sorszámmal látjuk el. Azok a bitek, amelyek a kettő hatványainak megfelelő sorszámúak az ellenőrzőbitek. A maradék részen az m adatbit van. Minden ellenőrzőbit, magát is beleértve, valamelyik bitcsoport paritását párossá (vagy páratlanná) teszi.
Annak megállapítását, hogy a k. adatbit melyik ellenőrzőbithez tartozik, úgy végezhetjük el a legkönnyebben, hogy a k-t 2 hatványainak összegeként felírjuk.
Például, 11 = 1 + 2 + 8 és 29 = 1 + 4 + 8 + 16.
Egy bitet csak az ő kiterjesztésében lévő ellenőrző bitek ellenőrzik (pl. a 11. bitet csak az 1., 2., és 8. bitek ellenőrzik).

Amikor egy kódszó megérkezik, a vevő kinulláz egy számlálót. Ezután minden egyes k (k = 1, 2, 4, 8, ...) ellenőrzőbitet megvizsgál, hogy helyes-e a

Karakter	ASCII	Ellenőrzőbitek
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	01111001111
	0100000	10011000000
c	1100011	11111000011
o	1101111	10101011111
d	1100100	11111001100
e	1100101	00111000101

1.5. táblázat. Néhány ASCII karakter Hamming kódja

paritása. Ha nem, akkor k-t ad a számlálóhoz. Ha a számláló minden ellenőrzőbit megvizsgálása után is nulla marad (azaz mindegyik helyes volt), akkor a vevő a kódszót érvényesnek fogadja el. Ha a számláló nem nulla, akkor értéke a hibás bit számát adja meg. Pl. ha az 1., 2. és 8. ellenőrzőbitek helytelenek, akkor az invertált értékű bit a 11., mivel csak ezt az egyetlen bitet ellenőrzi a 1., 2. és 8. bit.

A 1.5 ábrán néhány, egy Hamming-kód segítségével 11-bites kódszóvá kódolt 7-bites ASCII karakter látható. Az adatokat a 3., 5., 6., 7., 9., 10. és 11. bitpozícióban találjuk.

A Hamming kódok csak egyedi hibákat képesek javítani. Csoportos hibák mátrixos alakban kezelhetők, ahol a sorokba a kódszavak kerülnek. Csoportos hibajavításnál a kódszavakat nem sorban egymás után, hanem mátrixoszloponként küldjük el.

1.5.1.2.2. Hibajelző kódok.

Hibajavító kódokat csak ritkán használnak adatátvitelre, pl. szimplex csatornák esetén, amelyeknél újraadás nem kérhető. Leggyakrabban hibajelző kódokat és az azt követő ismétlést alkalmazzák.

Hiba felismerő kódok:

- Átlapolt kód

- Csoportos kód
- Paritás bit
- CRC - ciklus redundancia kód

Egyszerű példaként vegyünk egy csatornát, amelyen a hibák elkülönülve jelentkeznek, és a hibaarány 10^{-6}. Legyen a blokkméret 1000 bit. Hibajavításhoz 1000-bites blokkok esetén 10 ellenőrzőbit szükséges, egy megabitnyi adat 10 000 ellenőrzőbitet igényel. Egyszerű hibaészleléshez blokkonként egyetlen paritásbit elegendő. Minden 1000 blokk után egyetlen külön blokk (1001 bit) elküldése elegendő. Így teljes "overhead" a hibaészlelésre és újraadásra 2001 bit megabitenként, szemben a Hamming kód 10 000 bitjével.

Átlapolt kód

Ha egy blokkhoz egyetlen paritásbitet adunk, és a blokkban hosszú csoportos hiba keletkezik, akkor a hibajelzés valószínűsége csak 0.5, ami aligha fogadható el. A valószínűség jelentősen növelhető azáltal, hogy az egyes blokkokat n oszlopból és k sorból álló mátrixnak tekintjük, és ilyen formátumban küldjük el. A paritásbiteket oszloponként külön számítjuk ki, és a mátrixhoz annak utolsó soraként illesztjük hozzá. A mátrixot ezután sorfolytonosan küldjük el. Amikor a blokk megérkezik, a vevő ellenőrzi az összes paritásbitet. Ha bármelyik is rossz, az egész blokk újraadását kéri. Ez a módszer egyetlen n-bites csoporthibát képes észlelni, mivel oszloponként csak 1 bit változik. Egy n + 1 hosszú csoport azonban már kijelzetlenül átcsúszhat akkor, ha az első bit invertálódik, az utolsó bit invertálódik, a többi bit viszont helyes marad. (A csoporthiba nem azt jelenti, hogy az összes bit rossz lesz, csak legalább az első és utolsó bit rossz lesz.) Ha a blokk súlyosan megsérül egy hosszú csoporthiba vagy több egymást követő rövid csoporthiba következtében, akkor annak a valószínűsége, hogy egy oszlop véletlenül helyes paritással rendelkezik 0.5, így egy hibás blokk elfogadási valószínűsége 2^{-n}.

Cilikus redundancia kód

Bár ez a séma néha megfelelő, de a gyakorlatban egy másik módszer a polinomkód (polynomial code), nevezik még ciklikus redundanciakódnak és CRC kódnak is, használata terjedt el széleskörűen.

A polinomkódok a bitfüzéreket 0 és 1 együtthatójú polinomoknak tekintik. Egy k-bites keret egy olyan polinomnak az együtthatóit adja meg, amelynek k tagja x^{k-1}-től x^0-ig terjed. Az ilyen polinomot (k - 1) fokúnak nevezik. A bal oldali

legszélső bit az x^{k-1} együtthatóját adja meg, a következő az x^{k-2} együtthatóját és így tovább. Például, 110001 hat bitet tartalmaz, ezért egy hattagú polinomot ábrázol, amelynek együtthatói 1, 1, 0, 0, 0, 1, azaz a polinom $x^5 + x^4 + x^0$. A polinom aritmetika, az algebrai mezőelméletnek megfelelően, modulo 2-es aritmetikán alapul. Sem összeadásnál, sem kivonásnál nincs átvitel. Mind az összeadás, mind a kivonás kizáró vagy (XOR) művelettel azonos.

A hosszú osztás ugyanúgy kivitelezhető, mint a bináris, kivéve, hogy a kivonás modulo 2 szerint történik. Akkor mondjuk, hogy az osztó "megvan" az osztandóban, ha az osztandónak ugyanannyi bitje van, mint az osztónak.

Amikor polinomkód módszert alkalmazunk, a küldőnek és a vevőnek előre meg kell egyeznie egy G(x) generátorpolinomban. A generátor magas és alacsony helyértékű bitjeinek 1-nek kell lennie. Egy m-bites keret ellenőrző összegének (checksum) kiszámításához, amely az M(x) polinomra vonatkozik, a keretnek hosszabbnak kell lennie a generátorpolinomnál.

Az alapgondolat az, hogy az ellenőrző összeget úgy kell az elküldendő keret végére függeszteni, hogy az ellenőrzőösszeggel ellátott keret G(x)-el osztható polinomot ábrázoljon. Amikor a vevő megkapja az ellenőrző összeggel ellátott keretet, megpróbálja G(x)-el elosztani. Ha van maradék, akkor hiba keletkezett!

Az ellenőrzőösszeg kiszámításának algoritmusa:

1. Legyen G(x) r-ed fokú! Illesszünk a keret alacsony helyértékű végére r db 0-ás bitet úgy, hogy most már m + r bitet tartalmazzon, és így az $x^r M(x)$ polinomnak feleljen meg!

2. A modulo 2-es osztás szabályai szerint osszuk el az $x^r M(x)$ polinomot reprezentáló bitfüzért a G(x)-nek megfelelő bitfüzérrel!

3. A modulo 2-es kivonás szabályai szerint vonjuk ki a keletkezett maradékot (amely mindig r vagy kevesebb bitet tartalmaz) az $x^r M(x)$ polinomot reprezentáló bitfüzérből! Az így kapott eredmény adja a végső, elküldésre váró, ellenőrzőösszeggel ellátott keretet. Nevezzük ezt T(x)-nek!

Példa: a keret 1001011011, a generátor polinom
$G(x) = x^4 + x + 1$. (r=4)
Tehát a generátor: 10011.
Az üzenet 4 db 0 értékű bit hozzáfűzése után: 11010110110000.

Ezen elvégezzük a moduló 2 osztást.(ha az osztandó rész 1-gyel kezdődik, moduló 2-vel levonjuk az osztót).

```
    1 0 0 1 0 1 1 0 1 1 0 0 0 0
-   1 0 0 1 1
    ---------
      0 0 0 1 1
    -   0 0 0 0 0
        ---------
          0 0 1 1 1
        -   0 0 0 0 0
            ---------
              0 1 1 1 0
            -   0 0 0 0 0
                ---------
                  1 1 1 0 1
                -   1 0 0 1 1
                    ---------
                      1 1 1 0 1
                    -   1 0 0 1 1
                        ---------
                          1 1 1 0 0
                        -   1 0 0 1 1
                            ---------
                              1 1 1 1 0
                            -   1 0 0 1 1
                                ---------
                                  1 1 0 1 0
                                -   1 0 0 1 1
                                    ---------
                                      1 0 0 1 0
                                    -   1 0 0 1 1
                                        ---------
                                          0 0 0 1
```

Az osztás elvégzését a fenti táblázatban követhetjük nyomon. Az osztás maradéka 0001 lett.

Igy az átvitt bitsorozat: 11010110110001 (a keret, és utána fűzve a maradék.) Egyértelmű, hogy T(x)-nek oszthatónak kell lennie G(x)-el, modulo 2 rendszerben. Bármilyen osztásnál a maradékkal csökkentett osztandó már osztható az osztóval. Például 10-es számrendszerben, ha 210278-at elosztjuk 10941-el, akkor a maradék 2399. Ezt 210278-ból levonva 207879-et kapunk, amely már osztható 10941-el.

Elemezzük a módszer hatékonyságát:

Képzeljük el, hogy egy átviteli hiba következett be, melynek eredményeként a vevőhöz T(x) polinom helyett T(x) + E(x) polinom érkezik meg. Az E(x)-ben minden bit egy T(x)-ben invertált bitre vonatkozik. Ha k db 1-es van E(x)-ben, akkor k egybites hiba volt. Egyetlen csoportos hiba előfordulását egy kezdeti és egy lezáró 1-es közé zárt 0-ák és 1-ek keveréke jellemzi. Az E(x) többi bitje 0.

Az ellenőrzőösszeggel kibővített keretet megkapva a vevő elosztja azt G(x)-el, azaz kiszámítja [(T(x) + E(x))/G(x)]-et. A [T(x)/G(x)] mindig 0, így a számítás eredményét [E(x)/G(x)] adja. Csak azok a hibák maradnak észrevétlenek, melyek

olyan polinomoknak felelnek meg, melyeket G(x)-el osztva nincs maradék. Az összes többi hiba detektálható.

Ha egyetlen bithiba fordult elő, akkor $E(x) = x^i$, ahol i a hibás bitet jelöli ki. Ha G(x) két vagy több tagot tartalmaz, akkor E(x) sohasem osztható G(x)-el, azaz az összes egyedi bithiba kiderül.

Ha két különálló egybites hiba keletkezett, akkor $E(x) = x^i + x^j$, ahol i > j. Ez írható még $E(x) = x^j(x^{i-j} + 1)$ alakban is. Ha feltételezzük, hogy G(x) nem osztható x-szel, akkor az összes kettős hiba észlelésének elégséges feltétele az, hogy G(x) ne ossza $x^k + 1$-et semmilyen i-j-nél kisebb k értékre (azaz a keret maximális hossza alatt).

Ha páratlan számú hibás bit van, akkor E(x) páratlan számú tagot tartalmaz (pl. $x^5 + x^2 + 1, dex^2 + 1$ nem lehetséges). Nincs olyan páratlan tagszámú polinom, amelynek x + 1 tényezője lenne a modulo 2-es rendszerben! Emiatt, ha x + 1-et G(x) egyik tényezőjévé tesszük, minden páratlan számú invertált bitet tartalmazó hibát felfedezhetünk.

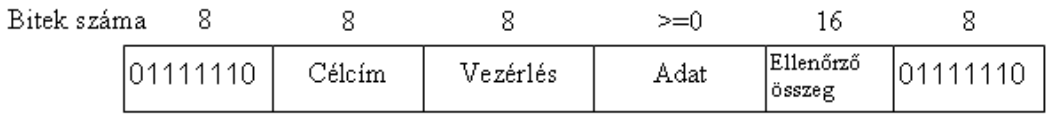
Mint a legfontosabbat megemlítjük, hogy egy r ellenőrzőbites polinomkód az összes r-nél kisebb hosszúságú csoporthibát észleli.
Három polinom vált nemzetközi szabvánnyá:
CRC-12 = $x^{12} + x^{11} + x^3 + x^2 + x + 1$
CRC-16 = $x^{16} + x^{15} + x^2 + 1$
CRC-CCITT = $x^{16} + x^{12} + x^5 + 1$

Az x+1-et mindegyik elsőrendű tényezőként tartalmazza. CRC-12-t 6-bites karakterek esetén, a másik kettőt 8-bites karakterek esetén alkalmazzák. Egy olyan 16-bites ellenőrzőösszeg, mint a CRC-16 és a CRC=CCITT, észleli az összes egyes és kettős hibát, az összes páratlan hibás bitet tartalmazó hibát, az összes 16 vagy annál rövidebb csoportos hibát, a 17-bites csoportos hibák 99,997 %-át, valamint a 18-bites és annál hosszabb csoporthibák 99,998 %-át.

1.5.1.2.3. A HDLC protokoll.

Jelentése: Magas szintű adatkapcsolat-vezérlés (High-level Data Link Control). Keret formátum a bitalapú protokollokhoz a 1.41. ábrán látható.

Minden bitalapú protokoll ezt a keretstruktúrát használja.



1.41. ábra. A HDLC protokoll formátuma

A *célcím mező* a több terminálos rendszereknél a terminál azonosítására szolgál. Pontkapcsolat esetén néha a parancsok és a válaszok megkülönböztetésére használják.

A *vezérlés mezőt* többek közt nyugtázásra és sorszámozásra használják.

Az *adat mező* tartalmazza az információkat, amely tetszőleges hosszúságú lehet, de meg kell jegyezni, hogy a kerethossz növekedésével (a többszörös csoportos hibák nagyobb valószínűsége miatt) az ellenőrző összeg hatékonysága romlik.

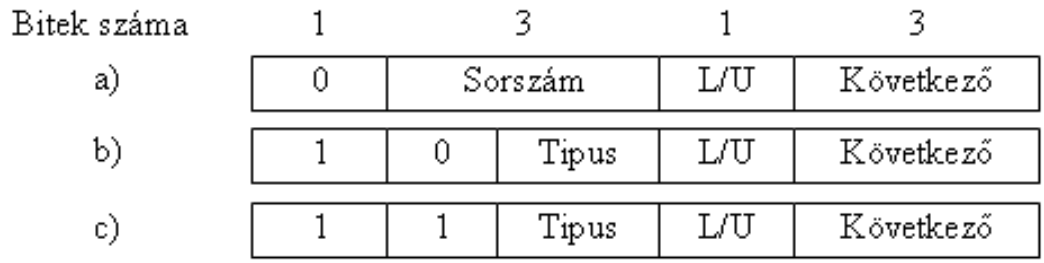
Az *ellenőrző összeg mező* a ciklikus redundancia kód a CRC.

A *keret kezdet és vég mezőben* a 01111110 8 bites jelsorozat található, mely minden keretet határol.

Keretosztályok:

- a) Információs (Information)
- b) Felügyelő (Supervisory)
- c) Számozatlan (Unnumbered)

Legfeljebb hét nyugtázatlan keret lehet a rendszerben egy adott pillanatban.



1.42. ábra. A HDLC vezérlése

1.6. A hálózati réteg

A hálózati réteg feladata a csomagoknak a forráscsomóponttól a célcsomópontig való eljuttatása. Egy csomagnak útja során esetleg több csomópontot is érintenie kell ahhoz, hogy célját elérhesse. Ezen feladathoz képest az adatkapcsolati réteg - amely csak egy vonal két vége közötti keretmozgatást végzi - feladata nyilvánvalóan szerény. A hálózati réteg az a legalacsonyabb réteg, amely két végpont közötti (end-to-end) átvitellel foglalkozik.

A hálózati réteg csak akkor képes a feladatát ellátni, ha ismeri a kommunikációs alhálózat topológiáját, és ki tudja választani az azon keresztül vezető legalkalmasabb útvonalakat. Figyelembe kell venni az útvonalak kiválasztásánál azt is, hogy az egyes kommunikációs vonalak nehogy túlterhelődjenek, míg mások kihasználatlanul maradjanak.

Végül, amikor a forrás és a célállomás különböző hálózatokban vannak, akkor a hálózati rétegnek kell foglalkoznia a hálózatok közti különbségekkel, és meg kell oldania az ezekből adódó problémákat. Ezek között találjuk a szállítási rétegnek nyújtott szolgálatot, a csomagok alhálózaton keresztüli forgalmának irányítását, a torlódásvezérlést és több hálózat összekötését (internetworki1ng).

1.6.1. A szállítási rétegnek nyújtott szolgálat

A hálózati réteg szolgálatokat nyújt a szállítási rétegnek. Mivel néhány hálózatban a hálózati réteg az IMP-kben, a szállítási réteg pedig a hosztokon fut, ezért ezekben a hálózatokban a hálózati és a szállítási réteg közötti határ egyben az alhálózat és a hoszt közötti határt is kijelöli. Ez azt jelenti, hogy a hálózati réteg által nyújtott szolgálat egyben az alhálózat által nyújtott szolgálatot is meghatározza.

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

1) A szolgálatoknak függetleneknek kell lenniük az alhálózat technikájától.

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

3) A szállítási réteg számára hozzáférhető hálózati címeknek egységes számozási rendszert kell alkotniuk. (LAN-okon és WAN-okon keresztül egyaránt.)

Kérdés	Összeköttetésalapú	Összeköttetésmentes
Kezdeti felépítés	Szükséges	Nem lehetséges
Célcím	Csak felépítés alatt kell	Minden csomagban kell
Csomag sorszámozás	Garantált	Nem garantált
Hibakorlátozás	Hálózati réteg végzi	Szállítási réteg végzi
	(pl.: alhálózat)	(pl.: hosztok)
Forgalomszabályozás	Hálózati réteg biztosítja	Hálózati réteg biztosítja
Opcióegyeztetés lehetséges?	Igen	Nem
Van összeköttetés azonosító?	Igen	Nem

1.6. táblázat. Az összeköttetésalapú és az összeköttetésmentes szolgálat közötti különbségek

Az összeköttetésmentes és összeköttetésalapú szolgálatok közötti viszonyt a következő analógia jobban megvilágíthatja.

A nyilvános távbeszélő-hálózatok összeköttetésalapú szolgálatot kínálnak. A telefonáló először egy számot tárcsáz azért, hogy felépülhessen az összeköttetés. A felek ezután beszélnek (adatokat cserélnek). Végül az összeköttetés lebomlik. Jóllehet, ami a távbeszélőrendszeren belül történik az kétségtelenül nagyon bonyolult, a két felhasználónak úgy tűnik, mintha egy dedikált két pontos csatornával rendelkeznének, amely az információkat mindig a küldési sorrendben kézbesíti.

Ezzel szemben a postai levéltovábbító rendszer összeköttetésmentes. Minden egyes levél tartalmazza a címzett teljes címét, és minden más levéltől függetlenül kézbesítik. A levelek sem szükségszerűen ugyanabban a sorrendben érkeznek meg, ahogyan elküldték őket. Ha pl. egy levélkézbesítő véletlenül elveszít egy levelet, a posta időzítése nem jár le, és nem küld egy újabb példányt. Röviden, a hibavédelmet és forgalomszabályozást maguk a felhasználók végzik, függetlenül a postától (alhálózattól).

A 1.6. táblázatban összefoglaltuk az összeköttetésmentes és összeköttetés alapú szolgálatok közötti különbségeket.

Az OSI összeköttetésalapú hálózat szolgálati primitívjei négy kategóriába csoportosíthatók:

összeköttetés létesítése, lebontása, használata és alaphelyzetbe állítás.

A legtöbb primitív paramétereket használ. Az a módszer, ahogyan a paramétereket a primitívekhez eljuttatják implementációnként változhat.

Az N CONNECT.kérés primitív összeköttetés létesítésére használható. Kijelöli a cél, valamint a hívó hálózati címét. Tartalmaz továbbá két, opcionális szolgálatkérésekre használható logikai változót is. A nyugtakérés a küldő számára lehetővé teszi, hogy minden egyes csomagra nyugtát kérjen. Ha a hálózati réteg nem teszi lehetővé a nyugtázást, akkor e változó értékét a kézbesítés során az N CONNECT.bejelentés primitívben hamisba (false) kell állítani. Ha a hálózat réteg biztosít ugyan nyugtázást, de a cél nem akarja használni, akkor N CONNECT.válasz primitívjében hamisba állítja a jelző értékét. Nyugta csak akkor használható, ha mind a szállítási funkcionális elemek, mind a hálózati szolgáltató használni akarják. E tulajdonság egyben az opcióegyeztetésre (option negotiation) is példát adott.

A sürgős-kérés jelző a következő példa az opcióegyeztetésre. Ha mindhárom fél elfogadja, akkor sürgős adatok (expedited data) küldésére nyílik mód, amelyek lényegüket tekintve olyan csomagok, amelyek megsértve a sorban állási sorrendet, azonnal a sor elejére kerülnek. Az, hogy ez ténylegesen vagy csak logikailag megy végbe implementációfüggő. A hívó fél elhelyezhet felhasználói adatokat is az összeköttetés-kérésekben. A hívott fél, a kérés elfogadása vagy visszautasítása előtt, megvizsgálhatja ezeket az adatokat. Az összeköttetés-kéréseket az N-CONNECT.válasz primitívvel lehet elfogadni, és az N DISCONNECT.kérés primitívvel lehet visszautasítani. Amikor egy kérés visszautasításra kerül, akkor a hívott az ok paraméterrel indokolhatja meg, hogy miért nem hajlandó elfogadni a kérést, és hogy ez az állapot tartós vagy átmeneti. A hálózati réteg maga is visszautasíthat egy összeköttetés-létesítési kérést pl. akkor, ha a kívánt szolgálatminőség nem elérhető (állandó állapot), vagy ha az alhálózat jelenleg túlterhelt (átmeneti állapot). A még nem tárgyalt N CONNECT és N DISCONNECT primitívek maguktól értetődőek, kevés megjegyzést kívánnak.

Miután egy összeköttetés felépült, mindkét fél az N DATA.kérés primitívet használhatja adatküldésre. Amikor a csomagok megérkeznek, egy N DATA.bejelentés primitív hívódik meg a vevő oldalon. A sürgős adatok a normál adatok primitívjeivel analóg szolgálat-primitíveket használnak. Ha a felek megállapodtak a nyugtázásban, akkor egy csomag megérkezésekor a vevőnek egy N DATA ACKNOWLEDGE.kérés primitívet kell kiadnia. Ez a primitív nem tartalmaz nyugtát,

így az adatokat küldő félnek egyszerűen számolnia kell a nyugtákat. Ha a nyújtott szolgálat minősége alacsony, és az adatok vagy, a nyugták elveszhetnek, akkor ez a séma nem túl kielégítő, de nem is a hálózati réteg feladata az, hogy hibamentes szolgálatot nyújtson, ezt a szállítási rétegnek kell végeznie. A hálózati rétegbeli nyugták csupán a szolgálat minőségének javítását, de nem a tökéletessé tételét célozzák meg.

Az N RESET primitíveket katasztrófák bejelentésére használják, pl. valamelyik szállítási entitásnak vagy magának a hálózati szolgáltatónak a meghibásodása esetén. Egy N RESET kérése, bejelentésé, megválaszolása és megerősítése után a sorok eredeti, üres alapállapotukba állnak vissza. Az N RESET kiadásakor a sorban levő adatok elvesznek. Ezen a ponton ismét a szállítási réteg feladata az, hogy a rendszer felépüljön az N RESET okozta helyzetből.

Az OSI összeköttetésmentes primitívjei az N UNIDATA primitívek legfeljebb 64512 byte adat elküldésére használhatók. Az N UNIDATA primitív nem nyújt sem hibavédelmet, sem forgalomszabályozást, de semmilyen más vezérlést sem. A küldő egyszerűen csak kirakja a csomagokat az alhálózatra, és a legjobbakat reméli.

Az N FACILITY.kérés primitív arra ad lehetőséget a hálózati szolgáltatás használójának, hogy az átlagos kézbesítési jellemzőkről adatokat szerezhessen egy adott célállomásra vonatkozóan (pl. a kézbesített csomagok százalékát). Az N FACILITY.bejelentés primitív magától a hálózati rétegtől jön, nem egy távoli szállítási entitástól.

Az N REPORT.bejelentés primitív lehetőséget ad a hálózati rétegnek, hogy problémáit a hálózati szolgálat használójának bejelentse. Ha pl. egy adott cél elérhetetlen, akkor e tényt ezzel a primitívvel lehet bejelenteni. A primitív használatának részletei hálózat függőek, a szabványban nem definiáltak.

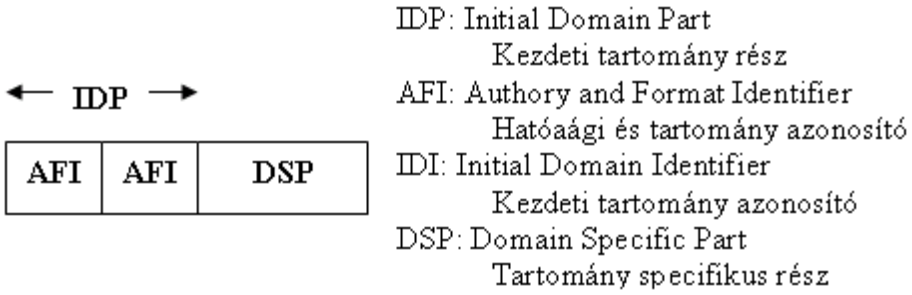
A hálózati réteg egyik funkciója az, hogy lehetővé tegye a szállítási réteg számára az egységes névhasználatot. Ideális az lenne, ha az összes hálózati operátor összegyűlne és egyetlen névtartományban meg egyezne úgy, hogy az alhálózatokból otthonokba stb. futó minden egyes vonal világszerte egyedi címmel rendelkezzen. Sajnos a helyzet nem ilyen irányba változik. Az adott helyzet lehető legjobbá tétele érdekében az OSI hálózati rétegének címzését úgy tervezték meg, hogy magába foglalja a ma létező különböző címzési módszereket.

Az összes hálózati szolgálatprimitív NSAP-címeket használ a forrás- és a célcsomópont azonosításához. Az NSAP-címek formátumát az 1.43. ábrán láthatjuk.

Minden NSAP-cím három mezőt tartalmaz.

Az első az AFI (Authority and Format Identifier - hatósági és, formátumazonosító), amely a harmadik mezőben a DSP-ben (Domain Specific Part - tartomány specifikus rész) levő cím típusát azonosítja. A kiosztott kódok lehetővé teszik, hogy a DSP mező csomagkapcsoló hálózati címeket, telefonszámokat, ISDN-számokat, telexszámokat és hasonló számozási sémákat tartalmazzon akár bináris, akár tömörített decimális formában. Az AFI mező értéke 10 és 99 között lehet, elég helyet hagyva a jövőben megjelenő új számozási sémák számára is.

második, vagyis az IDI (Initial Domain Identifier - kezdeti tartomány azonosító) mező azt a tartományt specifikálja, amelyhez a DSP-ben levő számok tartoznak. Ha pl. a DSP egy telefonszám, akkor az IDI az ország hívószáma lehet. A teljes NSAP-cím változó hosszúságú, de legfeljebb 40 decimális számjegy, vagy 20 byte hosszúságú.



1.43. ábra. Az OSI hálózati címek (NSAP) formátuma

Annak megértésére, hogy az NSAP mire jó, tekintsünk egy telefonrendszerrel kapcsolatos analógiát. A legtöbb modern hivatalban és otthonban manapság már több telefoncsatlakozó van, amelyekbe több készülék csatlakoztatható. Minden csatlakozó aljzatból egy telefonvonal vezet valamelyik telefonközpontba (PBX). Ezeknek a csatlakozóknak világszerte egyedi számuk van, amely egy országkódot, egy területi kódot és egy előfizetői számot tartalmaz. Amikor egy telefonkészüléket egy ilyen csatlakozóba dugunk, akkor az a csatlakozó számának feltárcsázásával érhető el. Vegyük észre, hogy a szám valójában a csatlakozóhoz tartozik, és nem pedig az aktuálisan hozzákapcsolt telefonkészülékhez. Amikor egy készüléket egy

másik hivatalba költöztetnek át, akkor az csak az új hivatali csatlakozó számával hívható. Így tehát, a csatlakozók a telefonrendszer egységes névtartományát határozzák meg, függetlenül attól, hogy aktuálisan milyen telefonkészüléket csatlakoztatunk hozzájuk, vagy hogy ki veszi fel a készüléket, ha az csöng. Ebben az analógiában a csatlakozók a NSAP-k és a telefonszámok az NSAP-címek.

Amikor egy szállítási entitás kéri a hálózatot, hogy hívjon föl egy távoli gépet, akkor megadja a felhívandó NSAP-címet (azaz telefonszámot). A hálózati réteg nem törődik azzal, hogy melyik szállítási entitás (azaz telefon) kapcsolódik az adott NSAP-hoz, azzal meg pláne nem, hogy ki birtokolja a szállítási entitást (azaz ki veszi fel a telefont). A szállítási funkcionális elemek NSAP-hoz kapcsolódása nem a hálózati, hanem a szállítási réteg hatásköre.

1.6.2. Forgalomirányítás

A hálózati réteg igazi feladata az, hogy a csomagok útját a forrástól a célig kijelölje. A legtöbb alhálózatban a csomagok több csomópontátlépést (hop) végeznek útjuk során. Az egyetlen figyelemre méltó kivételt az üzenetszórásos hálózatok jelentik, noha különböző hálózatban levő forrás- és célállomás esetén az útvonalkijelölés ott is probléma marad. A megfelelő útvonalat kiválasztó algoritmusok és az általuk használt adatstruktúrák a hálózati réteg tervezésének legfőbb területe. Ebben a szakaszban csak a probléma felvázolására törekszünk, a fejezetben később részletesen is megvizsgálunk több algoritmusjavaslatot.

A forgalomirányítási algoritmus (routing algorithm) a hálózati réteg szoftverének azon része, amely azért a döntésért felelős, hogy egy bemenő csomagot melyik kimenő vonalon kell továbbítani. Ha az alhálózat belsőleg datagrammokat használ, akkor e döntést minden egyes beérkező csomagnál meg kell ismételni. Ha azonban az alhálózat virtuális áramköröket használ belsőleg, akkor forgalomirányítási döntést csak egy új virtuális áramkör felépítésekor kell hozni. Ez utóbbi esetet gyakran viszony-forgalomirányításnak (session routing) nevezik, mert a kijelölt útvonal a teljes felhasználói viszony idejére érvényben marad (pl. egy terminálról végrehajtott bejelentkezés vagy egy állomány továbbítása esetén).

Vannak bizonyos tulajdonságok, amelyek egy forgalomirányítási algoritmusban mindenképpen kívánatosak :

helyesség, egyszerűség, robosztusság, stabilitás, korrektség és optimalitás.

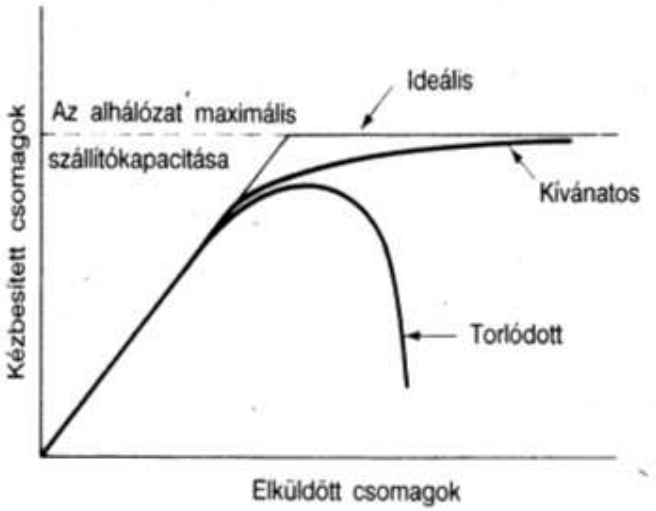
A helyesség és az egyszerűség nem kíván magyarázatot, de a robosztusság igénye talán kevésbé nyilvánvaló. Amikor, egy új hálózat életre kel, mindenki azt várja, hogy évekig rendszerszintű hibáktól mentesen működjön. Ez alatt az idő alatt mindenfajta hardver- és szoftverhiba előfordulhat. Hosztokat, IMP-ket és vonalakat kapcsolnak be-ki, cserélnek le és a topológia is rendszeresen változhat. A forgalomirányítási algoritmusnak mindezekkel együtt kell tudnia működni úgy, hogy néhány IMP meghibásodása miatt ne kelljen a hálózat hosztjain futó munkákat leállítani, és a teljes rendszert újraindítani.

A forgalomirányítási algoritmusokat két fő osztályba sorolhatjuk: *adaptív és nem adaptív* algoritmusok.

A nem adaptív algoritmusok az adaptívokkal ellentétben nem támaszkodnak döntéseikben a pillanatnyi forgalom és topológia mért, vagy becsült értékeire. Ha egy adaptív algoritmus jól alkalmazkodik az aktuális forgalomhoz, akkor nem kell előre felkészülnie bizonyos helyzetekre, hanyagul viselkedhet a hálózat jövőjét illetően. Jól alkalmazható algoritmusról azonban könnyebb beszélni, mint elkészíteni azt. Az adaptív algoritmusokat tovább bonthatjuk centralizált, elszigetelt, valamint elosztott (McQuillan, 1974) algoritmusokra, amelyekkel részletesen nem foglalkozunk.

1.6.3. Torlódás

Ha az alhálózatban (vagy annak egy részében) túl sok csomag van jelen, akkor a teljesítmény erősen lecsökken. A jelenséget torlódásnak (congestion) nevezik. A 1.44. ábrán e jelenséget mutatjuk be. Amikor a hosztok által az alhálózatba juttatott csomagok száma még az alhálózat kapacitásán belülre esik, akkor a csomagok kézbesítődnek (kivéve - néhány átviteli hiba miatt sérült csomagot), és a kézbesített csomagok száma arányos az elküldöttek számával. Ha azonban a forgalom túlságosan megnő, akkor az IMP-k már nem győzik a továbbítást, és elkezdenek csomagokat veszíteni. Ez a tendencia a forgalom növekedésével csak rosszabbodik, és egész magas forgalomnál a teljesítmény teljesen lezuhan, szinte egyetlen csomag sem kerül kézbesítésre.



1.44. ábra. Torlódás

Túl nagy forgalomnál, torlódás lép fel, és a teljesítmény erősen csökken. Torlódás fellépését több tényező okozhatja. Ha az IMP-k túl lassan végzik el adminisztrációs feladataikat (pufferek sorkezelése, táblák frissítése stb.), akkor sorok keletkezhetnek még akkor is, ha egyébként a vonalak fölös kapacitással rendelkeznek. Másfelől viszont, még ha az IMP CPU-ja végtelenül gyors is, de a kimeneti vonalak kapacitása kisebb, mint a bemeneti vonalaké, akkor ugyancsak sorok keletkeznek. Ilyen helyzet léphet fel akkor is, ha a három bemeneti vonalra teljes sebességgel érkező csomagokat csak egyetlen azonos kapacitású kimeneti vonalon lehet továbbítani. Bárhogyan is áll elő, a probléma mindenképpen az IMP elégtelen puffer-kapacitásában keresendő. Egy végtelen számú pufferrel ellátott IMP mindig elsimíthatná az ilyen szűk keresztmetszetből adódó problémákat azzal, hogy addig várakoztatná a csomagokat, amíg sor nem kerül azokra.

Természetesen, stabil működés esetén, a hosztok nem pumpálhatják a csomagokat nagyobb sebességgel az alhálózatba, mint ahogyan az alhálózat elnyelni képes azokat. A torlódásvezérlésnek azt kell biztosítania, hogy az alhálózat képes legyen a jelentkező forgalom lebonyolítására. Ez egy átfogó kérdés, amely magába foglalja az összes hoszt, IMP viselkedését, beleértve az IMP-ken belüli tároló és továbbító folyamatokat, és minden egyéb más tényezőt is, amelyek az alhálózat kapacitásának csökkenését idézhetik elő.

1.6.4. Hálózatközi együttműködés

A LAN-ok számos személyi számítógépet, munkaállomást és miniszámítógépet kapcsolnak össze. A számokat bogarászó (pl. fizikusok), vagy a szavakat morzsolgató emberek (pl. költők) inkább a számítógépközpontokat keresik fel, mivel az előbbiek a nagy számítási kapacitást keresik, az utóbbiak pedig tartózkodnak mindenféle hardverbonyodalomtól. Mind a tanszéki LAN-ok, mind a nagyszámítógépek gyakran nemzeti vagy nemzetközi WAN-okba is bekapcsolódnak, ill. egymással is összeköttetést teremtenek. A következő események könnyen elképzelhetők:

1) LAN-LAN. Egy számítástechnikus egy állományt ér el, és tölt le saját gépébe egy másik LAN-ról, valamilyen mérnöki tevékenységhez.

2) LAN-WAN. Egy számítástechnikus (elektronikus) levelet küld egy távoli fizikusnak.

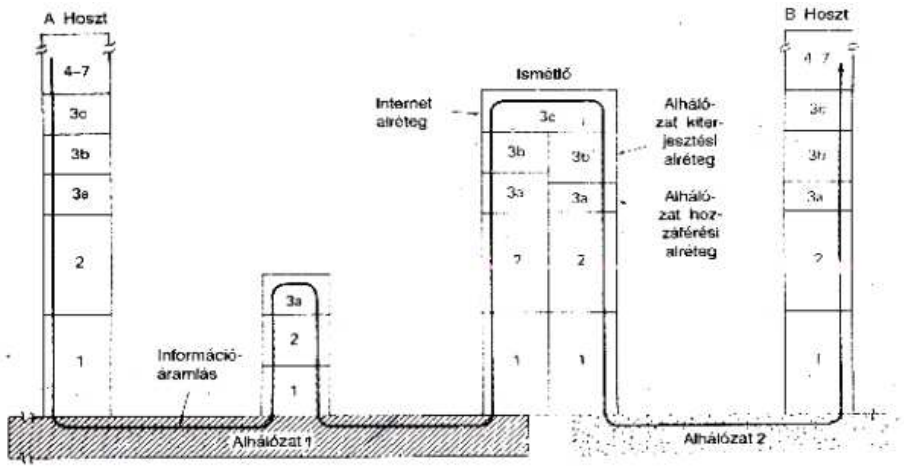
3) WAN-WAN. Két költő szonetteket cserél egymással.

4) LAN-WAN-LAN. Különböző egyetemeken levő mérnökök kommunikálnak egymással.

1.6.4.1. Az OSI és a hálózatközi együttműködés

A hálózati réteg három alrétegre bontható: alhálózat-hozzáférési alréteg (subnet access sublayer), alhálózat-kiterjesztési alréteg (subnet enhancement sublayer), internet alréteg (internet sublayer), ahogy a 1.45. ábrán látható. Az alhálózat-hozzáférési alréteg célja az, hogy a használt, egyedi alhálózat hálózati réteg protokollját kezelje. Adat- és vezérlőcsomagokat állít elő és vesz, valamint végrehajtja a hálózati réteg szokásos funkcióit. Nem garantálható, hogy más alhálózatokkal szintén működni fog. Az alhálózat-kiterjesztési alréteg céljául a különböző szolgálatokat kínáló alhálózatok összehangolását tűzték ki. A 1.45. ábrán látható ismétlőben a 3a és a 3a' felső határai különböznek. A 3b és a 3b felső határai azonban már ugyanazok, így a 3c már mindegyik alhálózattal működni tud.

Az internet alréteg elvileg bármilyen típusú alhálózat szolgálattal működhet. Ha datagramm szolgálatot választunk, akkor az alhálózat-kiterjesztési alréteg feladata az, hogy eltakarja a virtuális áramköröket, és csak datagramm szolgálatot nyújtson az internet alrétegnek. Ha jobbra nem képes, akkor minden egyes felülről hozzáérkező datagrammra egy virtuális áramkört létesít, elküldi rajta a datagram-



1.45. ábra. A hálózati réteg belső szerkezete

mot, majd lebontja az áramkört. A gyakorlatban valószínű, hogy az alhálózat-kiterjesztési alréteg a virtuális áramkört a tétlenné válása után még néhány percig fenntartja, mivel nagy esély van arra, hogy újabb datagrammok esetén ismét használnia kell.

Az internet alréteg alapvető feladata a vég-vég hatáskörű forgalomirányítás. Amikor egy csomag érkezik egy ismétlőbe, a rétegeken felfelé haladva eljut az internet alrétegbe, amely megvizsgálja, és eldönti, hogy továbbítsa-e, és ha igen, akkor melyik alhálózatot vegye igénybe erre a célra (a többoldalú ismétlők több alhálózat közül választhatnak). Első megközelítésben a több alhálózaton keresztüli forgalomirányítás hasonló az egyetlen alhálózatban alkalmazott forgalomirányításhoz, és az eddig tanulmányozott technikák itt is érvényesek. Nagy kiterjedésű internetre nyilvánvalóan a hierarchikus forgalomirányítást célszerű használni, mivel nem igényli, hogy az ismétlők ismerjék és számon tartsák a távoli alhálózatok struktúráját.

A 1.45. ábrán látható ismétlő a 3. rétegig terjed, és a hálózatok között a csomagokat ebben a rétegben mozgatja. Általános (nem OSI) esetben az ismétlés bármelyik rétegben végbemehet.

Az ismétlők négy közismert típusa a következő:

- 1. Réteg. Jelismétlő (repeater), kábelszegmensek közötti bitmásolást végez.

- 2. Réteg. Híd (bridge), LAN-ok között áramló keretek tárolását és továbbítását végzi.
- 3. Réteg. Átjáró (gateway), különböző hálózatok között áramló csomagok tárolását és továbbítását végzi.
- 4. Réteg. Protokollátalakító (protocol converter) az illesztést végzi a magasabb rétegekben.

A hidak tároló- és továbbító eszközök. Egy híd teljes kereteket vesz, és átadja az adatkapcsolati rétegnek, amely az ellenőrzőösszegét kontrollálja. Ezután a keret egy másik alhálózaton való, továbbításra lekerül a fizikai rétegbe. A hidak végezhetnek apróbb változtatásokat a kereteken, a továbbításuk előtt. Így pl. mezőket törölhetnek és adhatnak a keret fejrészéhez. Mivel ezek adatkapcsolati rétegbeli készülékek, nem foglalkozhatnak a 3. és az afölötti rétegek fejrészeivel, nem változtathatják meg azokat, és nem is hozhatnak azok tartalmától független döntéseket.

Az átjárók fogalmilag hasonlóak a hidakhoz, de a hálózati rétegben helyezkednek el. Az 1.45. ábrán látható ismétlő egy átjáró. Néhányan az átjáró fogalmát általános értelemben, minden rétegben alkalmazhatóan használják, és a forgalomirányító (router) kifejezést használják a hálózati rétegbeli átjáró megjelölésére. Ebben a fejezeten az "átjáró" a hálózati rétegre vonatkozik.

A szállítási rétegben és fölötte, az ismétlőket rendszerint protokoll átalakítóknak nevezik, bár ahogy már korábban említettük, néhányan az "átjáró" kifejezést használják, a megnevezésükre. Egy protokoll átalakító feladata sokkal összetettebb, mint egy átjáróé. A protokoll átalakítónak úgy kell egy protokollt egy másikba alakítania, hogy a folyamat során a protokollok lehetőleg a legkevesebbet veszítsenek jelentésükből. Protokollá alakítóra jó példa egy olyan ismétlő, amely az OSI szállítási réteg protokollját az ARPA Internetben használt protokollba (TCP) alakítja. Egy másik protokoll átalakítási példa egy OSI elektronikus levél üzeneteinek (MOTIS) ARPA Internetbeli formátumúvá (RFC 822) alakítása.

1.6.4.2. Hidak

A legtöbb híd 802 LAN-okat köt össze, ezért elsősorban a 802 hidakra koncentrálunk. Mielőtt a hidak technológiáját vizsgálnánk, érdemes egy pillantást vetnünk néhány olyan helyzetre, amelyben a hidak alkalmazása szükségessé válhat. Hat

olyan okot említünk meg, amiért akár egyetlen szervezet is több LAN használatára kényszerül.

Először is, sok egyetemi és vállalati részlegnek saját LAN fái vannak, amelyekhez személyi számítógépeket, munkaállomásokat és miniszámítógépeket kapcsolnak. Mivel az egyes részlegek céljai különbözne egymástól, ezért a különböző részlegek különböző LAN-okat használhatnak tekintet nélkül arra, hogy a többi részleg mit csinál. Előbb-utóbb szükségessé válik az együttműködés. Ebben az esetben tehát, a tulajdonosok autonómiája okozza a különböző típusú LAN-ok megjelenését.

Másodszor, a szervezetek egymástól földrajzilag is jelentős távolságra levő épületekben helyezkedhetnek el. Olcsóbb lehet, ha az egyes épületekben különálló LAN-okat használnak, és azokat hidakkal és infravörös összeköttetéssel kötik össze, és nem egyetlen koaxiális kábelt vezetnek végig az egész egyetem területén.

Harmadszor, a terhelési viszonyok kedvezőbbé tétele érdekében egy logikailag összetartozó LAN-t, különálló LAN-ok halmazára vághatunk szét. Például a Carnegie-Mellon Egyetemen sok ezer munkaállomás áll a diákok és a tanszéki dolgozók számítástechnikai igényeinek kielégítésére (Morris, 1988; Morris és mások; 1986). Az állományok rendszerint állományszolgáltatókon (fele server) helyezkednek el, a diákok onnan töltik le saját gépeikre. A rendszer hatalmas mérete eleve kizárja, hogy az összes munkaállomást egyetlen LAN-ba lehessen integrálni - a szükséges sávszélesség elérhetetlenül nagy. Ehelyett hidakkal összekötött LAN-okat alkalmaznak. Minden LAN egy saját állományszolgáltatót és egy csoport munkaállomást tartalmaz úgy, hogy a keletkező forgalom nagy része lehetőleg az adott LAN-on belül bonyolódjon le.

Negyedszer, néhány helyzetben egy LAN a terhelést tekintve alkalmas lenne az adott célra, de a legtávolabbi gépek közötti (pl. a 802.3 esetén 2,5 km-nél nagyobb) távolság már áthidalhatatlan számára. Még ha a kábel lefektetése könnyű is lenne, a fellépő nagy terjedési késleltetési idő megakadályozza a hálózat működését. Az egyetlen megoldás az, hogy a LAN-t részekre kell bontani, és a szegmensek között hidakat kell alkalmazni.

Ötödször, itt van a megbízhatóság kérdése is. Egyetlen LAN esetén egy hibásan működő csomópont, amely folyamatosan szeméttel telíti a hálózatot, megbénít-

hatja a LAN teljes működését. A hidakat hasonlóan az épületekben levő vészkijáratokhoz a kritikus helyekre tehetjük, így megóvhatjuk a teljes rendszert egyetlen "őrültté" vált csomóponttól. Az ismétlőktől eltérően a hidak programozhatóak, így felkészíthetők bizonyos megkülönböztetésre, vagyis mely kereteket továbbítson és melyeket ne.

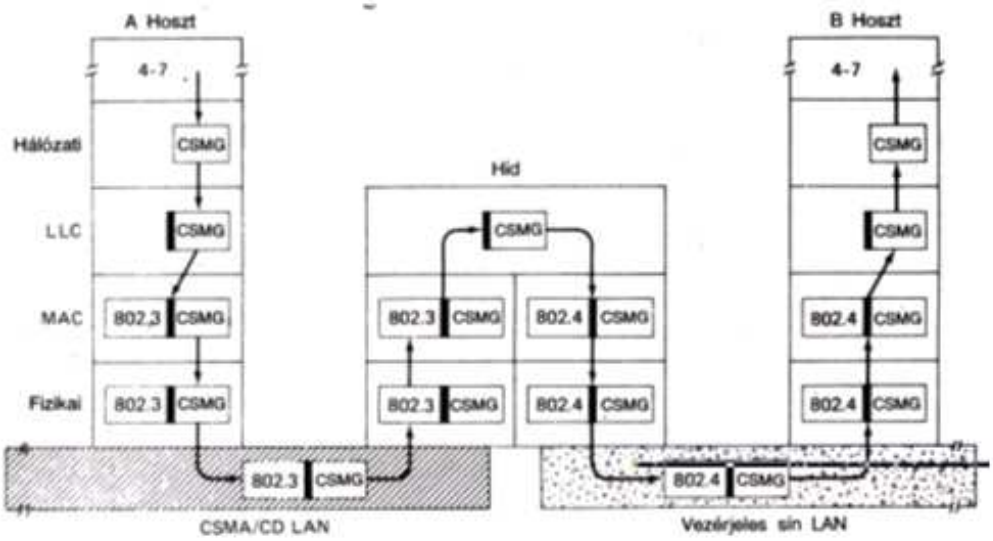
Hatodszorra, és egyben utolsóként említjük a hidak azon tulajdonságát, amely a szervezetek biztonságát növeli. A legtöbb LAN-interfésznek van egy ún. válogatás nélküli üzemmódja, amelyben az összes csomagot elküldi a számítógépnek, nemcsak azokat, amelyeknek címzettje. A kémek, és a minden lében kanál emberek kifejezetten szeretik ezt a tulajdonságot. Hidaknak megfelelő helyekre való telepítésével, valamint az érzékeny forgalom kiszűrésével a hálózat egyes részeit el lehet különíteni úgy, hogy annak forgalma más hálózati részekről nem hozzáférhető.

Miután már láttuk, hogy a hidak miért is szükségesek, fordítsuk figyelmünket arra a kérdésre, hogy miként is működnek. A 1.46. ábrán egy egyszerű kétoldalas hidat láthatunk. Az A hosztnak van egy elküldendő csomagja. A csomag lefelé haladva eléri az LLC alréteget, és ott egy LLC fejrészt kap. Ezután a MAC alréteghez kerül, és ott egy 802.3 fejrész illesztődik elé (és egy az ábrán fel nem tüntetett utótag is a csomag végére). Az egység ezután a kábelen keresztül eljut a híd MAC alrétegébe, ahol leveti 802.3 fejrészét. A csupasz csomag (az LLC fejrésszel) ezután a híd LLC alrétegéhez jut. Ebben a példában a csomag címzettje a hídhoz kötött 802.4 alhálózatban van, így a csomag útja a híd, 802.4 oldalán lefelé vezet. Vegyük észre, hogy egy k különböző LAN-t összekötő híd különböző MAC alréteggel, és k különböző fizikai réteggel rendelkezik, amelyek az egyes LAN-típusoknak megfelelőek !

1.6.4.3. Összeköttetésalapú és összeköttetésmentes átjárók összehasonlítása

A konkatenált virtuális áramkörnek és a datagramm megközelítésnek különböző előnyei és hátrányai vannak.

A konkatenált virtuális áramkör modellnek alapvetően ugyanazok az előnyei, mint egy virtuális áramkört használó alhálózatnak: a torlódás enyhíthető (az átjárókban) a pufferek előre-allokálásával, biztosítható a sorrendhelyes továbbítás,



1.46. ábra. Egy 802.3 és egy 802.4 LAN-t összekötő híd működése

rövid fejrészek használhatók, és elkerülhetők a késleltetett, kettőzött csomagokból származó problémák. Van azonban néhány hátránya is: minden nyitott összeköttetés számára táblahelyet kell fenntartani az átjárókban függetlenül attól, hogy van-e forgalom vagy nincs, a torlódás elkerülésére nem lehet alternatív utakat kijelölni, valamint rendkívül érzékeny az út mentén levő átjáró meghibásodásra. További hátránya az is, hogy implementálása nagyon nehéz, majdhogynem lehetetlen akkor, ha a hálózatok közül az egyik egy megbízhatatlan datagramm hálózat.

A hálózatközi datagramm szolgálat tulajdonságai ugyanazok, mint a datagramm alhálózatoké: nagyobb az eshetőség a torlódás kialakulására, ugyanakkor nagyobb az adaptálódási képessége a torlódáshoz, nagyobb az ellenálló képessége az átjárók meghibásodása esetén, valamint hosszabb a csomagfejrész igénye. Különböző adaptív forgalomirányítási algoritmusok alkalmazására ad lehetőséget az interneten belül, ugyanúgy, ahogy egyetlen datagramm hálózaton belül is.

A datagramm modell legfőbb előnye az, hogy olyan alhálózatok együttműködését is lehetővé teszi, amelyek belsőleg nem virtuális áramköröket használnak. Sok LAN, mozgó hálózat (pl. légi vagy tengeri flották), sőt még néhány WAN is ebbe a kategóriába esik. Amikor az együttműködő hálózatok közül az egyik ilyen,

akkor súlyos problémák adódhatnak, ha a hálózatközi együttműködés a virtuális áramkör stratégiára épül.

1.7. A szállítási réteg

1.7.1. Bevezetés

A szállítási réteg (Transport layer) alapvető feladata az, hogy adatokat fogadjon a viszonyrétegtől, kisebb darabokra vágja szét azokat (ha szükséges), majd adja tovább a hálózati rétegnek, és biztosítsa, hogy minden darab hibátlanul érkezzék meg a másik oldalra. E cél eléréséhez a hálózati réteg által nyújtott szolgálatokra támaszkodik. Mindezeket a feladatokat anélkül kell hatékonyan végrehajtania, hogy a viszonyréteg a hardvertechnikában elkerülhetetlenül bekövetkező változásokat észlelné. A hálózati réteg és a szállítási réteg nem különíthető el teljesen egymástól, a szállítási réteg tulajdonképpen feljavítja a hálózati réteg által nyújtott szolgáltatás minőségét. Egyik legfontosabb feladata a hosszabb átvitelek közben esetlegesen felbomló összeköttetés újbóli felépítése. Ilyenkor a távoli szállítási réteggel új kapcsolat épül ki, amelyen keresztül folytatódhat az adattovábbítás.

1.7.1.1. Szolgálatminőségi jellemzők

A szállítási réteg feladata az is, hogy megvizsgálja a hálózati réteg szolgálatait, és ezek alapján eldöntse, hogy tudja-e biztosítani a viszonyréteg által igényelt szolgálatot.

A szállítási réteg maga is különböző szolgáltatásokat biztosít a viszonyréteg felé. Ezeket a szolgáltatásokat különböző szolgálatminőségi jellemzőkkel jól definiálhatjuk:

Összeköttetés-létesítési késleltetés: azt az időmennyiséget adja meg, mely a szállítási réteg felhasználójának összeköttetés-kérése és az arra kapott megerősítés között eltelik. Ebbe az időmennyiségbe a távoli szállítási réteg késleltetése is beletartozik.

Összeköttetés-létesítési hibavalószínűség: azt definiálja, hogy az összeköttetés-létesítési késleltetésre megadott időn belül milyen valószínűséggel nem jön létre az összeköttetés.

Áteresztőképesség: a csatorna átviteli képessége bit/sec mértékegységben megadva.

Átviteli késleltetés: megadja, hogy egy adat a forrás szállítási réteg üzenetküldése után minimum mennyi idő alatt érkezik meg a vevő félhez (műholdas kapcsolatnál ez legalább 270 ms).

Megmaradó hibaarány: a szállítási réteg által a viszonyréteg felé továbbított adatok hibaszázaléka (bár a szállítási réteg feladata éppen a hibák elrejtése a felsőbb rétegek elől, a valóságban ez az érték mégsem nulla).

Szállítási hibavalószínűség: azt adja meg, hogy milyen arányban tartja be a rendszer a megmaradó hibaarányt, illetve azt, hogy az átvitel milyen arányban felel meg a két szállítási réteg által egyeztetett paramétereknek.

Összeköttetés-lebontási késleltetés: megadja, hogy az összeköttetés lebontását kérő jelzés (disconnect) után legfeljebb mennyi idő múlva szabadul fel az átviteli csatorna.

Összeköttetés-lebontási hibavalószínűség: megadja, hogy a rendszer milyen valószínűséggel nem képes a kapcsolatot az előbb definiált időn belül lebontani.

Védelem: meghatározza az adatátvitel biztonságát, azaz hogy egy külső fél milyen nehezen tud hozzáférni az adatokhoz.

Prioritás: meghatározza a szállítási réteg által létrehozott összeköttetések fontosságát, és ezáltal biztosítja, hogy torlódás esetén a magasabb prioritású összeköttetéseket a rendszer előbb szolgálja ki. Így a magasabb prioritású adat megelőzheti az alacsonyabb prioritásút.

Rugalmasság: definiálja, hogy milyen valószínűséggel áll fenn a kapcsolat a két oldal között (a szállítási réteg is bonthatja a kapcsolatot).

Az összeköttetést kérő felhasználó az összeköttetés kérés során megadja azokat az szolgálatminőségi paramétereket, amelyeket az összeköttetésnek teljesítenie kell. Előfordulhat, hogy a szállítási réteg rögtön felismeri, hogy az összes kívánt paraméter nem teljesíthető, és értesíti az összeköttetést kérő felet a kísérlet eredménytelenségéről anélkül, hogy megpróbálná az összeköttetést felépíteni. Az összeköttetés létesítését kérő fél megadja az összeköttetésre a paraméterek kívánt értékét, valamint a számára még elfogadható értékeket. Amennyiben a távoli fél a kívánt paramétereket nem tudja teljesíteni, de a minimális paramétereket igen, a két fél között paraméter-egyeztetés történhet. Ezt a folyamatot, mely során a két fél egyezteti az összeköttetés paramétereit, *opcióegyeztetésnek* nevezzük. Az egyeztetett opciók az összeköttetés teljes időtartama alatt érvényben maradnak.

1.7.2. Szállítási primitívek

T-CONNECT primitívvel jelzi a szállítási réteg az összeköttetés-létesítési szándékát. A primitívnek négy változata van:

- T-CONNECT.kérés
- T-CONNECT.bejelentés
- T-CONNECT.válasz
- T-CONNECT.megerősítés

T-DISCONNECT primitívvel jelzi az összeköttetésben részt vevő felek egyike, hogy meg kívánja szüntetni az összeköttetést. A T-DISCONNECT kérés primitívre az összeköttetés bontását kérő szállítási réteg nem vár választ. Változatai:

- T-DISCONNECT.kérés
- T-DISCONNECT.bejelentés

T-DATA primitív egyszerű adatküldésre használható. Változatai:

- T-DATA.kérés
- T-DATA.bejelentés

T-EXPEDITED-DATA primitív olyan adatok küldésére használható, amelyeknek nagy a prioritása. Az így küldött adatok átugorják a sorban előttük állókat. Ezt a primitívet általában valamilyen folyamatot megszakító parancs átvitelére használják. Változatai:

- T-EXPEDITED-DATA.kérés
- T-EXPEDITED-DATA.bejelentés

T-UNITDATA primitívet az előbbiekkel ellentétben összeköttetés nélküli kapcsolatnál használják adatküldésre. Ilyenkor a küldő fél nem vár választ vagy nyugtát az elküldött adatokra. Változatai:

- T-UNITDATA.kérés
- T-UNITDATA.bejelentés

1.7.3. Hálózati kommunikáció

A szállítási réteg egy összeköttetés felépítése, használata majd lebontása során általában négy primitívet használ. Amikor a szállítási réteg felé egy összeköttetéslétesítési kérelem érkezik a viszonyréteg felől, akkor azt a szállítási réteg egy T-CONNECT.kérés primitívvel jelzi a rendszer további része felé. Ebben a primitívben egyben megnevezi azt a címet, amellyel az összeköttetést létre kívánja hozni. Ez egy T-CONNECT.bejelentés primitív formájában jelenik meg a címben megadott másik félnél. A megszólított fél meg tud felelni a kezdeményező paramétereinek, akkor egy T-CONNECT.válasz primitívvel elfogadja az összeköttetési igényt. Amennyiben a megszólított fél a kívánt paramétereket nem tudja teljesíteni, egy T-DISCONNECT.kérés primitívvel válaszol a kezdeményezőnek. (A folyamat illusztrációja a 1.47. ábrán megtekinthető.)

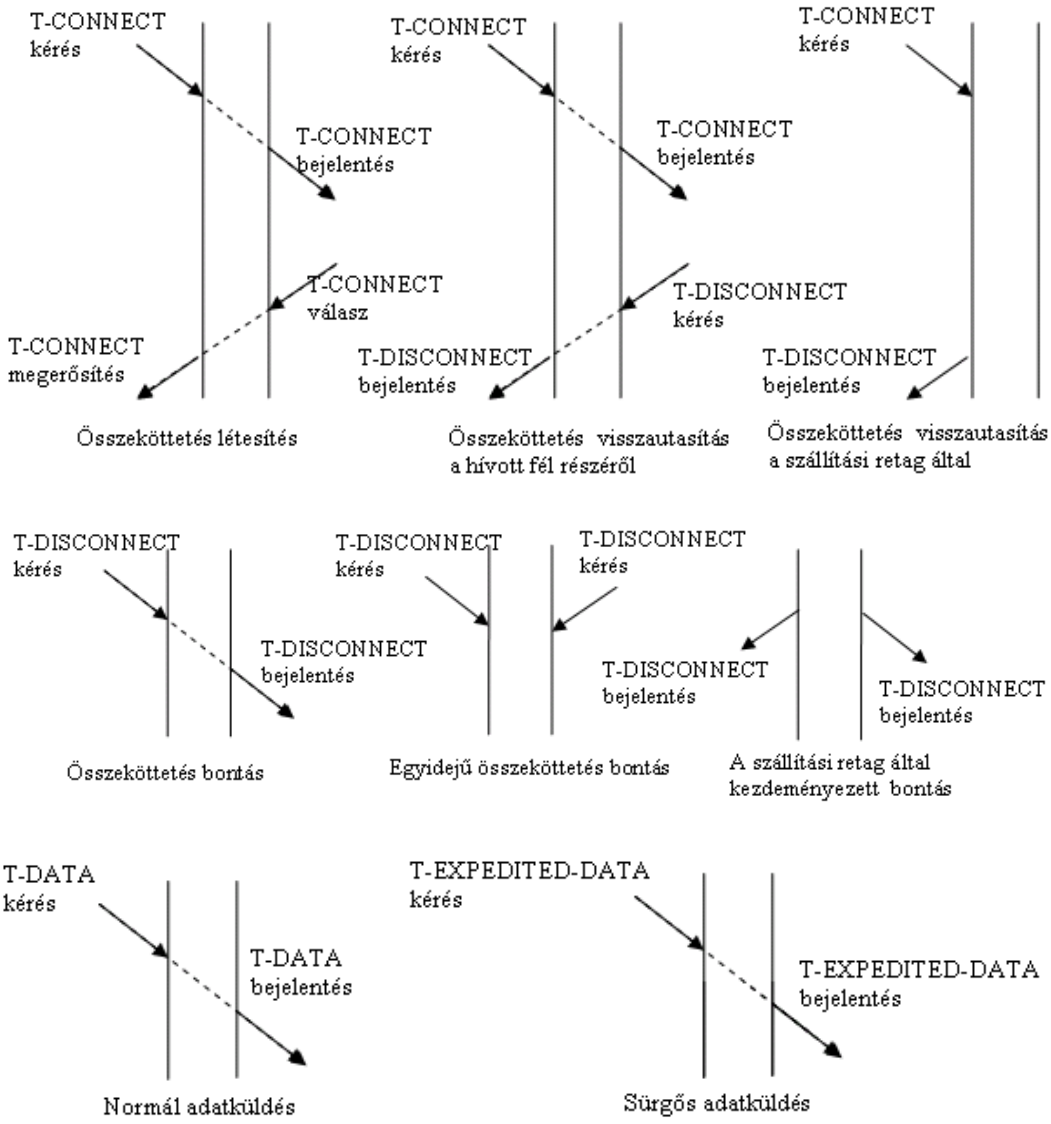
Előfordulhat, hogy az összeköttetést kezdeményező fél által elküldött T-CONNECT.kérés primitív nem jut el a címzetthez, mert maga a szállítási szolgáltató utasítja el az összeköttetést. Ez akkor lehetséges, ha például hibás a szállítási szolgáltató (fizikai réteg), vagy ha a kérés primitívben rossz paramétert vagy címet adott meg a kezdeményező fél. Ilyenkor a szállítási réteg egy T-DISCONNECT.bejelentés primitívvel jelzi a kapcsolatot kezdeményező fél felé az összeköttetés lebontását. Ilyenkor a címzett fél tudni sem fog a hibás összeköttetéslétesítési kísérletről.

Az összeköttetés lebontása háromféleképpen mehet végbe. Az a szokványos, hogy a kapcsolatban résztvevő egyik fél T-DISCONNECT.kérés primitívet ad ki, mely a másik oldalon T-DISCONNECT.bejelentés primitívként jelenik meg. Az összeköttetés lebontását mindkét fél kezdeményezheti.

Amennyiben a két fél egy időben jelenti be kapcsolatlebontási szándékát T-DISCONNECT.kérés primitívvel, akkor az összeköttetés úgy szűnik meg, hogy a T-DISCONNECT.bejelentés primitív egyik félhez sem jut el.

Lehetséges az is, hogy a szállítási réteg zárja le az összeköttetést. Ilyenkor mindkét fél számára T-DISCONNECT.bejelentés primitívet küld. Ez akkor fordulhat elő, ha például a szolgáltató hálózat működésképtelenné válik.

A sürgős adattovábbítást a T-EXPEDITED-DATA primitív használatával valósíthatjuk meg. Ilyenkor a küldő fél nem kap nyugtát vagy választ. A sürgős adat



1.47. ábra. Szállítási primitívek

továbbítását az adatot küldő fél a T-EXPEDITED-DATA.kérés primitívvel jelzi, mely a másik félnél a többi primitívet megelőzve (a hálózat késleltetését figyelembe véve azonnal) T-EXPEDITED-DATA.bejelentés primitívként jelentkezik.

1.7.4. Hálózatok osztályozása

Különböző hálózati szolgáltatások különböző minőségű szolgálatot biztosítanak. A hálózati réteg azonban részben eltakarhatja a fizikai hálózat rossz tulajdonságait

és így jobb interfészt biztosíthat. Ennek alapján háromféle hálózati osztályt különböztetünk meg.

A osztály: hibátlan, N-RESET nélküli adatátvitelt biztosít. Ezeknél a hálózatoknál az elveszett, megsérült vagy kettőzött adatcsomagok száma is elhanyagolható. Ilyen tulajdonságokkal rendelkeznek általában a LAN-ok .

B osztály: összeköttetés-mentes szolgáltatással tökéletes csomagkézbesítést biztosít (N-RESET lehetséges). Az ilyen típusú szolgálatnál csomagok csak ritkán vesznek el, de a hálózati réteg időnként kiad N-RESET-eket, például torlódás vagy hardverhiba esetén. Ilyenkor a szállítási protokoll feladata, hogy új összeköttetést hozzon létre és folytassa a megszakadt feladatot. Általában ebbe az osztályba tartoznak a WAN-ok.

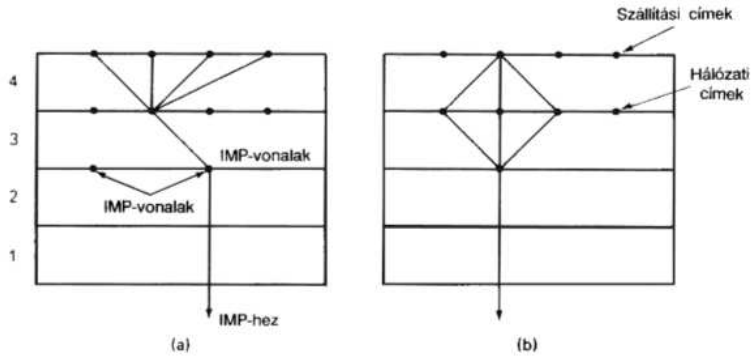
C osztály: megbízhatatlan csomagküldéses szolgálat (ezek összeköttetés-mentes szolgálatot biztosító hálózatok).

A hálózati szolgáltató típusától függően más és más szállítási protokollok szükségesek. Minél rosszabb a hálózati szolgálat, annál összetettebb szállítási protokollt kell megvalósítani. Ennek a problémának a kezelésére a szállítási réteg öt hálózati osztályt különböztet meg.

A hálózati osztályok ismertetése előtt meg kell határoznunk két fogalmat, melyek szükségesek a hálózati osztályok megkülönböztetéséhez. Ezek a nyalábolás és a hibákból való felépülés.

Nyalábolás: a nyalábolás fogalma azt jelenti, hogy a rendszer a különböző szállítási összeköttetéseket ugyanahhoz a hálózati összeköttetéshez rendeli hozzá, azaz egy fizikai csatornán keresztül több összeköttetést is megvalósíthat (pl. a kapcsoló számítógépek felé). A nyalábolásnak két fajtája létezik. Felfelé nyalábolásnak (upward multiplexing) nevezik, amikor a szállítási réteg a szállítási összeköttetéseket céljuk szerint csoportosítja, és azokat minimális számú hálózati összeköttetésen keresztül valósítja meg. Lefelé nyalábolásnak (downward multiplexing) nevezik azt a nyalábolási módot, amikor a szállítási réteg több hálózati összeköttetést használ, s az adatforgalmat ciklikusan megosztja közöttük. A két különböző típusú nyalábolási módszer illusztrációja a következő 1.48. ábrán megtekinthető.

A hibákból való felépülés: a hibák javítása és az összeomlás utáni helyreállítás a szállítási réteg egyik legfőbb feladata. Ez a feladat akkor válik igazán



1.48. ábra. Nyalábolási módszerek

lényegessé, ha a hálózat megbízhatatlan vagy a hosztok az összeomlás veszélyének vannak kitéve. Ha például a hálózati réteg egy N-RESET-et ad ki, akkor a szállítási rétegeknek egyeztetniük kell, hogy mely adatcsomagok érkeztek meg a vevő oldalra és melyek nem. Másik lehetőség az, amikor hoszt-összeomlás történik. Ilyenkor a hosztnak az összeomlás után fel kell derítenie az összeomlás előtti státuszát, amelyet a többi hosztnak küldött üzenetszórásos lekérdezéssel valósíthat meg. Ezen a problémák a megoldására általános, minden helyzetben jól működő algoritmus nem létezik. Ennek ellenére a szállítási rétegnek mégis kezelnie kell ezeket a hibákat, és a hálózattól függően meg kell oldania a javítást.

Ezek után az öt hálózati osztály a következő:

Egyszerű osztály: az A osztályra épülő szolgáltatás. Akkor alkalmazható, amikor a hálózati réteg megbízható (a hálózat hibamentes adatátvitelt biztosít). Ekkor a szállítási réteg nem végez semmilyen hibajavítást, csak továbbítja az adatokat a viszonyréteg felé. Ez a protokoll minden kért szállítási összeköttetésre egy új hálózati kapcsolatot létesít.

Alaphibákból felépülő osztály: a B osztályra alapuló szolgáltatás, mely az N-RESET-et kiküszöböli. Egy N-RESET utasítás után a két szállítási réteg újra szinkronizálódik és onnan folytatják a feladatot, ahol az félbeszakadt. Az N-RESET-ekből való felépüléseken túl semmilyen hibavédelmet vagy forgalomszabályozást ez a protokoll nem biztosít.

Nyaláboló osztály: az A osztályra épülő szolgáltatás, mely a csatorna jobb kihasználtsága érdekében több adatátviteli feladatot képes összefűzni, mikor a

csatornának kicsi a terheltsége. Ez azt jelenti, hogy egy fizikai kapcsolatra több szállítási összeköttetést tud multiplexelni.

Hibákból felépülő nyaláboló osztály: a B osztályra épülő szolgáltatás, mely az N-RESET-et kiküszöböli, az átvitel során fellépő hibákat ki tudja javítani és - amennyiben a csatorna kihasználtsága megengedi - nyalábolást végez. Ugyanakkor képes az N-RESET-ekből való felépülésre is.

Hibajelző és hibákból felépülő osztály: C osztályra épülő szolgáltatás, mely az adatátvitel során fellépő összes hiba javítását elvégzi. Előnye, hogy csak primitív hálózati szolgáltatást igényel, így maga a fizikai hálózat egyszerű és olcsó. A szállítási rétegnek tehát ezt az öt fajta hálózatot kell tudnia megkülönböztetnie, és ezen hálózatok tulajdonságainak megfelelően úgy végrehajtania az adatküldést, illetve -fogadást, hogy a felsőbb rétegek elől rejtve maradjanak az átvitel során fellépő esetleges hibák és az N-RESET-ek kiadásával járó rendszerhibák.

A szállítási réteg ezt a feladatot a megfelelő szolgálati primitívek alkalmazásával valósítja meg.

1.7.5. Összefoglalás

A szállítási réteg feladata tehát az, hogy megszüntesse a kínált hálózati szolgálat és a szállítást használó által megkívánt szolgálat közötti különbséget. Ezzel egy időben a szabványosított szolgálat definícióval a hálózati technológiát elfedi a felsőbb rétegek elől. A szabványosított definíció azt is jelenti, hogy a fizikai összeköttetésben bekövetkező változtatások nem teszik szükségessé a felsőbb rétegek szoftverének módosítását.

Az OSI-modell szerinti szállítási szolgálat három fázist különböztet meg, az összeköttetés-létesítést, -használatot és -lebontást. Az OSI-szabvány ezek mindegyikéhez definiálja a megfelelő szolgálati primitíveket.

1.8. A viszony réteg

1.8.1. Bevezetés

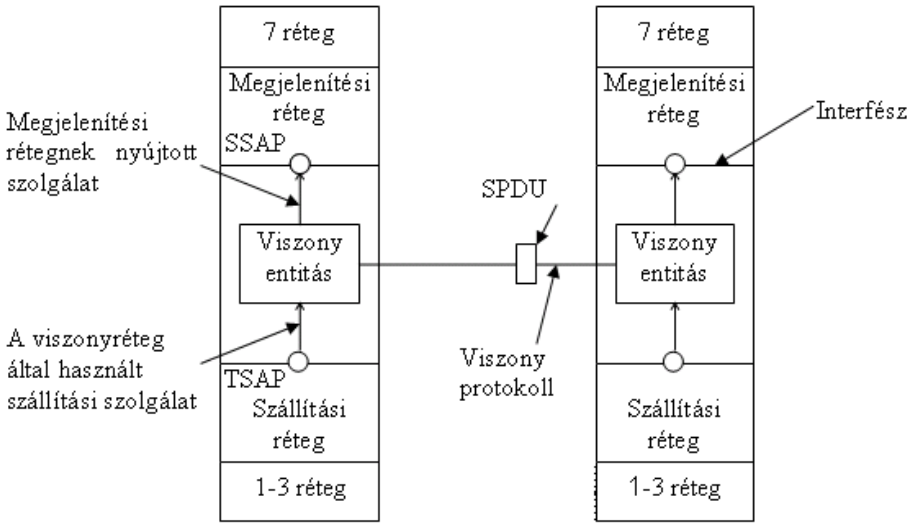
Míg az OSI-modell alsó négy rétege két végpont közötti hibátlan kommunikációt biztosít, addig a viszonyrétegtől kezdődően a felső három réteg felhasználó-orientált szolgálatokat nyújt.

A viszonyréteg lehetővé teszi, hogy különböző gépek felhasználói viszonyt (session) létesítsenek egymással. A viszonyréteg, akárcsak a szállítási réteg közönséges adatátvitelt tesz lehetővé, mindezt néhány olyan szolgáltatással kiegészítve, amelyek egyes alkalmazásokhoz hasznosak lehetnek.

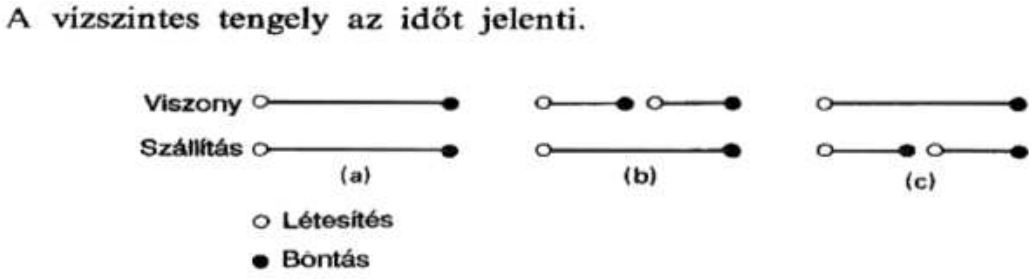
Egy viszony pl. arra alkalmas, hogy egy felhasználó bejelentkezzen egy távoli időosztásos rendszerbe, vagy hogy állományokat cseréljen két gép között.

A viszonyréteg egyik szolgáltatása a párbeszéd szervezése. A viszonyok egy időben egy- és kétirányú adatáramlást is lehetővé tehetnek. Amennyiben a forgalom csak egyirányú lehet (hasonlóan az egyvágányos vonatforgalomhoz), a viszonyréteg feladata a soron következő irány nyomon követése. A viszony és a szállítási összeköttetés sok mindenben hasonlít egymásra, mégsem azonos. (A viszonyréteg kapcsolata az őt körülvevő rétegekhez a 1.49. ábrán figyelhető meg.) A szállítási réteg szolgálatokat nyújt a felette elhelyezkedő viszonyrétegnek. Egy viszony létesítéséhez egy szállítási összeköttetés kiépítésére van szükség.

Amikor egy viszony befejeződik, akkor általában a hozzá tartozó szállítási összeköttetés is lebomlik. A helyzet azonban nem mindig ilyen egyszerű. Elképzelhető olyan eset, amikor a viszony lebontása után nincs szükség a szállítási összeköttetés megszüntetésére, mert rövid időn belül újra szükség lesz rá. Ilyenkor jobb kihasználtságot biztosít, ha az egymás után következő viszonyok ugyanazt a szállítási összeköttetést használják. Egy másik lehetőség, amikor egy viszony alatt a szállítási összeköttetés valamilyen oknál fogva megszakad. Ilyenkor a viszonyrétegnek egy új szállítási összeköttetést kell létesítenie, és a megkezdett adatátvitelt ezen új összeköttetésen keresztül folytatnia. (Ezt szemlélteti a 1.50. ábra.) Ez akkor lényeges, amikor a szállítási rétegek hoszton kívüliek, mert ilyenkor a hálózati hibákból való felépülés a viszonyréteg feladata. Ebben az esetben egy viszony több szállítási összeköttetést használ időben egymás után.



1.49. ábra. A viszonyréteg kapcsolatai



1.50. ábra. Szállítási és viszony összeköttetések

Ugyanakkor lényeges eltérés a szállítási réteg működésétől, hogy több viszony egyidejű megvalósítása egyetlen szállítási összeköttetésen keresztüli nem lehetséges (azaz a viszonyok nyalábolása nem engedélyezett).

1.8.1.1. Szolgálati primitívek

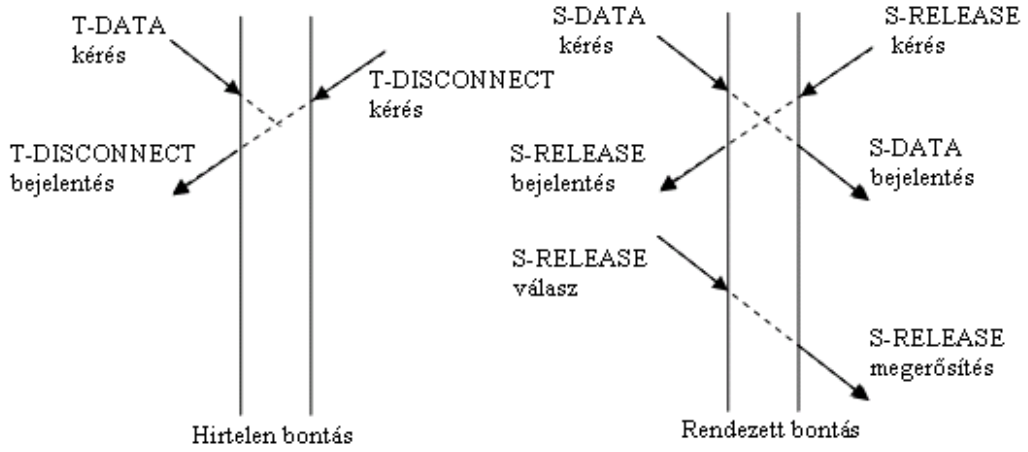
A viszonyréteg szolgálati primitívei lényegesen különböznek az alatta lévő entitások primitíveitől, hiszen többek között ezen a szinten jelenik meg először a nyugtázott bontás, a szinkronizáció stb. A viszonyréteg szolgálati primitíveit és azok feladatát a következő táblázatban összegeztük.

	K	B	V	M	
Primitívek					Funkció
Összeköttetés alapú					
S-CONNECT	x	x	x	x	Összeköttetés létesítés
S-RELEASE	x	x	x	x	A viszony rendezett bontása
S-U-ABORT	x	x			A felhasználó általi hirtelen bontás
S-P-ABORT		x			A szolgáltató általi hirtelen bontás
S-DATA	x	x			Normál adattovábbítás
S-EXPEDITED-DATA	x	x			Sürg?s adat továbbítás
S-TYPED-DATA	x	x			Sávon kívüli adattovábbítás
S-CAPABILITY-DATA	x	x	x	x	Vezérl? adat továbbítás
S-TOKEN-GIVE	x	x			Vezérjel-átadás
S-TOKEN-PLEASE	x	x			Vezérjrl kérés
S-CONTROL-GIVE	x	x			Az összes vezérjel átadása
S-SYNC-MAJOR	x	x	x	x	Fő szinkronizációs pont beszúrása
S-SYNC-MINOR	x	x	x	x	Mellék szinkronizációs pont beszúrása
S-RESYNCHRONIZE	x	x	x	x	Vissza az előző szinkronizációs ponthoz
S-ACTIVITY-START	x	x			Tevékenység indítása
S-ACTIVITY-END	x	x	x	x	Tevékenység vége
S-ACTIVITY-DISCARD	x	x	x	x	Tevékenység eldobása
S-ACTIVITY-INITERRUPT	x	x	x	x	Tevékenység felfüggesztése
S-ACTIVITY-RESUME	x	x			Tevékenység folytatása
S-U-EXCEPTION-REPORT	x	x			Felhasználói kivétel bejelentés
S-P-EXCEPTION-REPORT		x			Szolgáltatói kivétel bejelentés
Összeköttetés mentes					
S-UNITDATA	x	x			Összeköttetés mentes adat továbbítás

Egy új viszony létrehozásakor a viszonyréteg meghívja a szállítási réteg megfelelő primitívjét. Egy viszony létrehozásakor ugyanúgy lehetséges a társviszonyok közötti paraméter-egyeztetés, ahogy ezt a szállítási rétegnél már láttuk. Ezen paraméterek közül néhány a másik viszonyrétegre, néhány pedig a szállítási rétegre vonatkozik.

A legszembetűnőbb különbség a viszony-összeköttetés és a szállítási összeköttetés lebontása között van, hiszen a szállítási réteg erre szolgáló primitívje *hirtelen bontást* eredményez, melynek során adatvesztés léphet fel, míg a viszony-összeköttetés lebontására szolgáló primitív *rendezett bontást* valósít meg, melynek során adatvesztés nem lehetséges. Azonban egy külön erre a célra szolgáló primitív

(S-U-ABORT kérés) használatával a viszonyréteg is kérhet hirtelen bontást. A két különböző kapcsoltbontási eljárás a 1.51. ábrán látható.



1.51. ábra. Az összeköttetések különböző bontási sorrendje

A viszony szinten megjelenő nyugtákról az alsóbb rétegek gondoskodnak. A viszonyentitáshoz tartozó 58 összeköttetés alapú szolgálati primitívet 7 csoportba oszthajuk:

Összeköttetés-létesítési primitívek: Ez a csoport négy primitívet tartalmaz S-CONNECT xxx formában (a táblázatból kiolvasható). Az S-CONNECT kérés primitív kijelöli a viszonyazonosítót, a hívó és a hívott fél címét, a szolgálatminőséget, a kezdeti szinkronizációs pont számát, a kezdeti vezérjel-hozzárendeléseket, a felhasználói adatokat (amennyiben vannak) és az egyéb lehetséges opciókat. Az opciók a viszony fennállása során használni kívánt szolgálatokat határozzák meg, illetve adatátviteli típusok engedélyezésére és tiltására használhatók.

Viszonybontási primitívek: Ez a csoport hét primitívet tartalmaz. Az S-RELEASE kérés primitív rendezett bontást tesz lehetővé. Egy másik lehetőség az egyeztetett bontáshoz bontási vezérjelet (release token) használunk. Ilyenkor bontást csak az a felhasználó kezdeményezhet, aki a bontási vezérjelet birtokolja. Ezt a megvalósítást akkor alkalmazzák általában, ha a két fél mester-szolga viszonyban van. Ezeken kívül létezik még két hirtelen lebontási forma is: az S-U-ABORT primitívvel a felhasználó kezdeményezhet

hirtelen bontást, az S-P-ABORT primitívvel a szolgáltató teheti meg ugyanezt. A szolgáltató azonban csak akkor kezdeményezhet hirtelen bontást, ha a viszonyréteg valamilyen fatális hibát észlel.

Adattovábbítási primitívek: A négy különböző típusú adatfolyamhoz (normál adat, sürgős adat, minősített adat, képességadat) szintén négy különböző primitív tartozik. Most nézzük meg az utolsó kettőre vonatkozó primitíveket (a másik kettőt lásd később). Minősített adat érkezését S-TYPED-DATA bejelentés primitív jelzi. Ezt az adattípust a felhasználó leginkább vezérlőüzenetek küldésére használhatja (de bármilyen más egyéb adat küldésére is használható). Ezek a vezérlőüzenetek a hálózatmenedzselés számára soron kívül biztosíthatnak adatot. A minősített adat küldése nem függ vezérjeltől, ezért ilyen adat bármikor továbbítható. A képességadatok (capability data) kifejezetten a viszonyréteg számára küldött üzenetek küldésére szolgálnak. Általuk lehetővé válik a viszony fennállása alatt is a viszonyopciók és paraméterek megváltoztatása. A képességadatok teljesen nyugtázottak és csak két tevékenység között küldhetők, abban az esetben, ha a küldő az adat-, a szinkronizációs- és a tevékenység-vezérjelek birtokában van.

Vezérjel-menedzselési primitívek: A viszonyrétegben három vezérjellel találkozunk. Az S-TOKEN-GIVE kérés primitívvel vezérjelet lehet átadni a társentitásnak. Az átadandó vezérjelet a primitív paraméterei jelölik ki. Az S-TOKEN-PLEASE kérés primitívvel annak kibocsátója a paraméterek által kijelölt vezérjel iránti igényét jelenti be. Az S-CONTROL-GIVE kérés primitívvel az összes vezérjelet lehet egyidejűleg átadni a társentitásnak. Ez a primitív is csak tevékenységen kívül használható.

Szinkronizációs primitívek: Minden szinkronizációs eljáráshoz létezik külön primitív. Az egyes primitívek megadják a visszaállítás helyét vagy a beállítandó hely szinkronizációs sorszámát (0...999999 tartományban). Minden szinkronizációs primitív kiadása megköveteli a megfelelő vezérjel birtoklását.

Tevékenység-menedzseléssel kapcsolatos primitívek: A tevékenységek indításának, megszakításának, újraindításának stb. menedzselése is vezérjelek által biztosított. (A részletesebb leírást lásd később.)

Kivételes helyzetet jelző primitívek: Ezeket a primitíveket (S-P-EXCEPTION-REPORT és S-P-ABORT) a felhasználó nem, csak a szolgáltató adhatja ki, amennyiben szükség van rá.

1.8.1.2. Kölcsönhatás-menedzselés

Az OSI-modell a felső szinten full-duplex kommunikációt ír elő. Előfordulhat azonban olyan helyzet, hogy a felsőbb rétegek szoftvere csak a fél-duplex kommunikációt támogatja, ilyen lehet pl. egy adatbázis-kezelő rendszer, amely távoli terminálokról is elérhető. Fél-duplex üzemmódban vagy a felhasználónak vagy az adatbázist biztosítónak van adási joga, de a kettőnek egyszerre sohasem. Azt a mechanizmust, amelynek feladata eldönteni, hogy éppen melyik félnek van adási joga, kölcsönhatás-menedzselésnek (dialog management) nevezik. Ezt a feladatot a viszonyrétegnek kell megvalósítania.

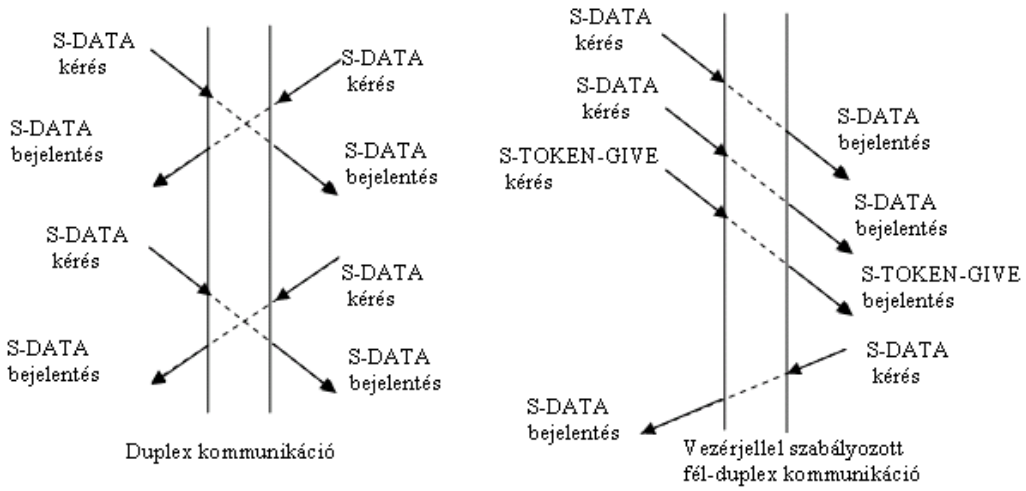
A kölcsönhatás-menedzselést egy adat-vezérjel (data token) segítségével valósítják meg. Amikor egy új viszonyt hozunk létre, akkor opcionálisan választható ki a fél-duplex működés. Ilyenkor a paraméter-egyeztetés során az is eldől, hogy melyik fél kapja meg először a vezérjelet. Csak a vezérjelet birtokló küldhet adatot.

Amikor a vezérjelet birtokló befejezi az adást, akkor átadhatja az adatküldés jogát a másik félnek az S-TOKEN-GIVE.kérés primitívvel. Amennyiben az a fél szeretne adatot küldeni, aki nem birtokolja a vezérjelet, akkor egy S-TOKEN-PLEASE.kérés primitívvel elkérheti az adattovábbítás jogát a vezérjelet birtokló féltől. Ha a vezérjelet birtokló fél elfogadja ezt a kérést, akkor S-TOKEN-GIVE.kérés primitívvel válaszol vagy elutasítja a kérést. A duplex és fél-duplex kommunikáció protokolljának megvalósítását a 1.52. ábra szemlélteti.

Ha viszony létesítésekor full-duplex üzemmódot választunk, akkor az adattovábbításra való jogosultság kijelöléséhez nincs szükség vezérjelre, hiszen mindkét fél küldhet adatot egy időben.

1.8.1.3. Szinkronizálás

A viszonyréteg szintjén az adatküldés már nagyméretű adategységek formájában is lehetséges. Egy nagyobb méretű adatállomány átküldése során nagyobb eséllyel léphet fel valamilyen hiba az adattovábbításban, mint az alsóbb szinteken továbbított kisebb méretű csomagok esetén. Annak elkerülése érdekében, hogy egy



1.52. ábra. Összeköttetési módok

hiba fellépése esetén az egész adatállományt újra kelljen küldeni, a társentitásokat szinkronizálni kell.

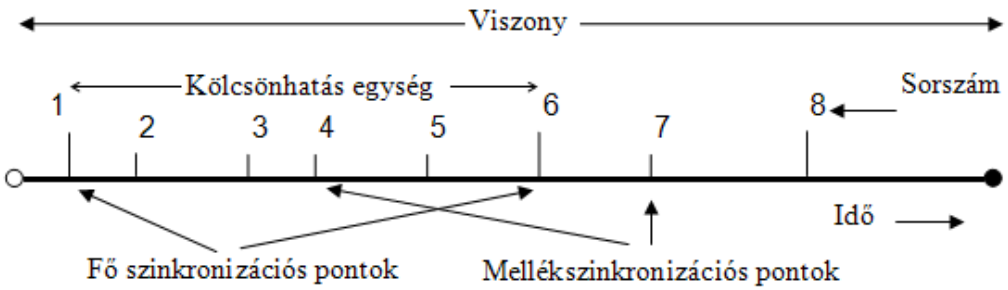
A szinkronizáció lehetőséget nyújt arra, hogy hiba esetén a viszonyrétegek egy korábbi ismert állapothoz térjenek vissza és onnan folytassák az adattovábbítást. Ez látszólag a szállítási réteg feladata lenne, azonban a szállítási réteg csupán a kommunikációs hibák javítására alkalmas.

Tény, hogy nemcsak a két fél közötti kommunikáció során jelentkezhet hiba vagy adatvesztés. Erre lehet példa, amikor egy adatbázis-kezelő rendszerben a terminál felé küldött adatállományt a terminál nem tudja valamilyen lemezes tárolóegységen rögzíteni, ezért azt azonnal pl. nyomtatóra írja ki. Ilyenkor, ha a terminál már vette az adatot és ezt nyugtázta a küldő fél felé és ezután a nyomtatásban valamilyen hiba jelentkezik (például begyűrődik a lap), akkor adatvesztés léphet fel.

Ennek megoldására a viszonyréteg meghatározott nagyságú darabokra (lapokra) osztja fel a teljes adatállományt, és ezen darabok közé szinkronizációs pontokat (synchronisation point) szúr be. Így, ha valamilyen hiba lép fel, akkor a viszonyréteg egy korábbi szinkronizációs pontnak megfelelő állapotból folytathatja a munkát. Ez akkor lehetséges, ha a küldő viszonyentitás mindaddig tárolja az adatokat, ameddig szükség lehet rájuk. Hiba fellépése esetén természetesen a két viszonyréteg között egyeztetésre van szükség, hogy kiderüljön, a vevő oldalán mely

szinkronizációs pontig kerültek az adatok hibátlanul megjelenítésre. Ezt a folyamatot újraszinkronizálásnak (resynchronisation) nevezzük. Minden szinkronizációs pontnak van egy sorszáma, amely alapján a két fél meg tudja azokat különböztetni.

A rendszer kétféle szinkronizációs pontot különböztet meg, a mellék- és a főszinkronizációs pontokat. (A szinkronizációs pontok helyzetére egy lehetséges példát mutat be a 1.53. ábra.)



1.53. ábra. Szinkronizálás

A főszinkronizációs pontokkal határolt egységet kölcsönhatás-egységnek (dialog unit) nevezzük. A főszinkronizációs pontok a folyamat logikailag jelentősebb részeit jelölik. Újraszinkronizáció esetén a két félnek a legutolsó főszinkronizációs pontig lehet visszamennie, és az adattovábbítást onnan lehet folytatnia. A küldő tudja, hogy amit a legutolsó főszinkronizációs pont előtt küldött, az biztonságban megérkezett a túloldalra és feldolgozásra került, így az oda tartozó adatokat kitörölheti.

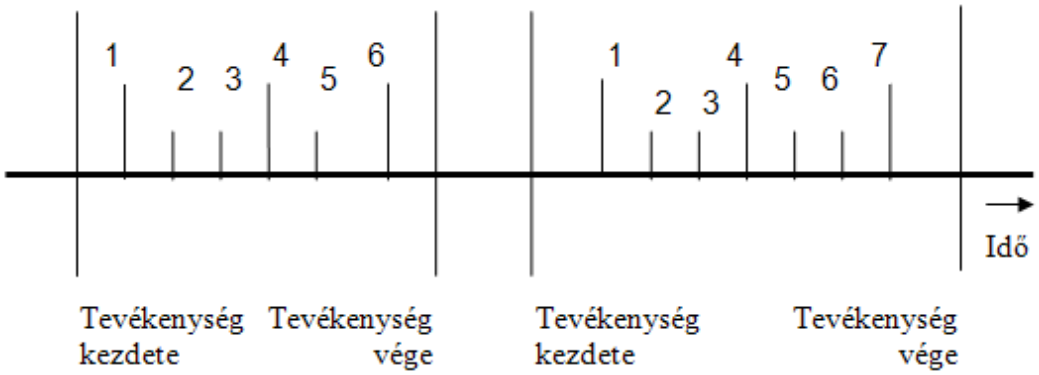
Adatátvitel közben a főszinkronizásiós pontok közé mellékszinkronizációs pontok ékelődnek. Hiba esetén az előző mellékszinkronizációs ponttól is újra kezdhető az adattovábbítás, azonban - eltérően főszinkronizációs pontoktól - a mellékszinkronizációs pont előtt küldött adatok nem kerülnek törlésre a mellékszinkronizációs pont után sem. Ebből következően a mellékszinkronizációs pontra a küldő fél nem vár nyugtázást.

A szinkronizációs pontok beállításához is vezérjelre van szükség. A két különböző szinkronizációs ponthoz különböző vezérjelek tartoznak. Amikor a rendszer újra-szinkronizációt hajt végre, az összes vezérjel abba a pozícióba kerül vissza, amelyben a szinkronizációs pont beállításakor volt. A szinkronizációt és az újra-

szinkronizációt a viszonyréteg bonyolítja le, ám az üzenetek tárolása és a soron következő esetleges újraadása már a viszonyréteg feletti entitások feladata.

1.8.1.4. Tevékenységmenedzselés

A viszonyréteg egy további, a szinkronizálással kapcsolatos feladata a tevékenységmenedzselés. Ez a szolgálat lehetővé teszi a felhasználónak, hogy az adatfolyamot logikai egységekre, azaz tevékenységekre (activity) bontsa. A tevékenységek egymástól függetlenek. Például egy olyan viszony esetén, amikor több adatállományt akarunk átvinni két számítógép között, meg kell jelölnünk az állományok kezdetét és végét. Ezt egy olyan címke beszúrásával érhetjük el, amely különbözik az adatüzenettől. Az ilyen címkét nevezik sávon kívüli (out of band) információnak. A feladat illusztrációja a 1.54. ábrán látható.



1.54. ábra. Tevékenység menedzselés

A rendszer ezt úgy valósítja meg, hogy minden állománytovábbítás előtt kiad egy S-ACTIVITY.kérés primitívet, mely a másik fél számára S-ACTIVITY.bejelentés primitív formájában jelenik meg. A küldő fél ezzel jelzi az adatállomány kezdetét. Az állomány végét az adó fél az S-ACTIVITY-END primitív segítségével hasonló módon jelez a másik fél felé.

A tevékenység meghatározása a felhasználó feladata, nem a viszonyrétegé. A viszonyentitás csak azt biztosítja, hogy az egyik oldalon kiadott S-ACTIVITY.kérés eljusson a másik félhez, illetve a vevő oldal válaszát eljuttassa a felhasználónak. Ebből következik, hogy a viszonyréteg nem értelmezi az S-ACTIVITY primití-

vet, csak továbbítja azt. A tevékenység mindkét irányú adatforgalmat magában foglalhatja.

A viszonyréteg egyéb feladatokat is ellát. Ezen feladatok közül az egyik a karanténba zárásnak (quarantining) nevezett szolgáltatás. Ez a technika azt eredményezi, hogy a vevő fél az üzenetek feldolgozását csak azután kezdi meg, miután az összes üzenetet egy bemeneti pufferben eltárolta.

A tevékenységmenedzselés fontos tulajdonsága, hogy a tevékenységek megszakíthatók, és később információvesztés nélkül folytathatók. Az adatküldés közben bármikor kiadható az S-ACTIVITY-INTERRUPT.kérés primitív, melynek hatására az állománytovábbítás ideiglenesen szünetel. A tevékenység felfüggesztése után új indítható, majd annak befejeződése után a megszakított tevékenység folytatható.

A tevékenységmenedzselés akkor vihető véghez, ha a két fél egyezteti a tevékenység struktúráját. Akkor azonban, amikor mindkét fél egyszerre akar elindítani egy tevékenységet, hiba léphet fel. Ennek elkerülése végett egy vezérjelet alkalmaznak (mely ugyanaz a vezérjel, amelyet a főszinkronizációs pontoknál is használnak). Csak az a fél indíthatja el a tevékenységet, amelyik a tevékenység vezérjelét birtokolja. Az azonban még nem elegendő, hogy a küldő fél a tevékenység vezérjelét birtokolja, hiszen ha az egyik felhasználó akkor indít el egy tevékenységet, amikor a másik éppen mellékszinkronizációt végez, akkor hiba léphet fel. Ezért egy tevékenység vagy egy szinkronizációs művelet megkezdéséhez a felhasználónak a szinkronizációs mellékjelet és az adatvezérjelet is birtokolnia kell (amennyiben az adatvezérjel használatban van). Ekkor azonban előfordulhat, hogy az egyik felhasználó a tevékenység-vezérjelet, a másik pedig a mellékszinkronizációs vezérjelet birtokolja, és mindketten tevékenységet szeretnének indítani. Ebben az esetben mindketten folyamatosan S-TOKEN-PLEASE primitíveket adnak ki, és végetlen ciklusban a másikra várnak, azaz holtpontban vannak. Ezt a problémát csak az alkalmazások szintjén tudjuk megoldani.

A tevékenységek szoros kapcsolatban vannak a szinkronizációs pontokkal, hiszen új tevékenység indításakor a szinkronizációs sorszámok 1-től indulnak. A tevékenység kezdetét jelző címke egyben főszinkronizációs pont is. Ebből követke-

zik, hogy a tevékenység megkezdése után nem lehet a tevékenység kezdete előtti szinkronizációs pontra visszaszinkronizálni.

1.8.2. Összefoglalás

A viszonyréteg az első olyan szolgálati entitás, amely nem csupán a kommunikációval foglalkozik. A viszonyréteg által biztosított legfontosabb szolgálatok:

- kölcsönhatás-menedzselés, amely a fél-duplex összeköttetés esetén eldönti, hogy melyik félnek van adási joga
- szinkronizálás, mely a hibákból történő felépülést segíti
- tevékenység-menedzselés, mely logikai egységekre bontja az adathalmazt.

1.9. A megjelentési réteg

1.9.1. Bevezetés

A megjelenítési réteg (presentation layer) olyan feladatok végrehajtásáért felelős, amelyek elég gyakoriak ahhoz, hogy általános megoldásúak legyenek ahelyett, hogy a felhasználók esetenként külön-külön oldják meg azokat.

Az alsó rétegektől eltérően, amelyek csak a bitek megbízható mozgatásával foglalkoznak, a megjelenítési réteg az átviendő információ szintaktikájával és szemantikájával foglalkozik. A megjelenítési entitás alatti rétegek az adatokat csak bitmintaként kezelik, viszont az alkalmazási rétegnek már olyan adatokra van szüksége, melyek közvetlenül megjeleníthetőek. Ezért a megjelenítési réteg feladata az adatok struktúrájának értelmezése, konverziója (például az ASCII és az EBCDIC kódrendszerek között).

A megjelenítési entitás feladatai közé tartozik továbbá az adattömörítés (amennyiben érdemes) és az adatvédelem biztosítása.

A megjelenítési réteg feladata tehát az, hogy a küldő gép belső adatábrázolási formája szerint struktúrált adatokat egy átvitelre alkalmas bitfolyammá alakítsa, majd ezt az adatfolyamot a célnál igényelt ábrázolási formájúvá alakítsa vissza.

A megjelenítési rétegnek négy alapvető feladata van:

- Viszonyszolgálati primitívek végrehajtása.
- Összetett adatstruktúrák definiálása.
- Az aktuálisan szükséges adatstruktúrák menedzselése.
- Adatok külső és belső formája közötti átalakítás.

1.9.2. Adattömörítés

Az adatábrázolás az adatátviteli rendszerek egyik legnagyobb problémája, hiszen nincsen univerzális szabvány az adatábrázolás miként jére. Az adatátvitelt azonban nem csak a kapcsolatban lévő gépek adattárolásának inkompatibilitása és annak kiküszöbölése lassíthatja, hanem a fizikai csatorna sávszélessége is. Amennyiben kicsi a csatorna sávszélessége, és így lassú az átvitel, azzal is csökkenthetjük az átvitel idejét, ha a tényleges fizikai átvitel előtt tömörítjük az adatainkat, ezáltal az átküldendő adatblokk nagyságát lecsökkentjük. A tömörítés azért is gazdaságos, mert a számtógép-hálózatokat működtető szerverek használatáért általában

az átvitt adatok mennyiségének függvényében kell fizetni, így adattömörítés alkalmazása költségcsökkenéssel jár. Megjegyzendő, hogy tömörítés nem mindig gazdaságos. Előfordulhat, hogy átviteli csatornánk sávszélessége roppant nagy, s a tömörítés-kitömörítés több időt igényel, mint a tömörítetlen adat átvitele.

Az adattömörítés szoros összefüggésben van az adatábrázolással, mert más-más struktúrájú adatokat más-más algoritmussal lehet a leghatékonyabban tömöríteni. A csatornákon elküldött adatokat szimbólumok sorozataként is felfoghatjuk, mely szimbólumok egy olyan szimbólumkészletbe tartoznak, amelynek minden elemét ismerjük. Adattömörítést három általános módszerrel érhetünk el. Ezek egyenként a következőkre építenek:

- A szimbólumkészletek véges volta.
- A használt szimbólumok relatív gyakorisága.
- Jellemző szimbólumsorozatok.

1.9.2.1. Egyformán valószínű szimbólumok véges készletének kódolása

Sok alkalmazásban az üzeneteket egy véges halmazból nyerik. Ilyenkor, ha a halmaz elemszáma relatíve kicsi, a szimbólumok pedig viszonylag hosszúak, akkor előfordulhat, hogy jobban járunk, ha a szimbólumokat egyszerűen sorszámozzuk vagy valamilyen más egyszerű kódolást alkalmazunk.

1.9.2.2. Gyakoriságfüggő kódolás

Egyes adathalmazokban (például szövegben) néhány szimbólum jóval nagyobb gyakorisággal tűnik fel, mint a többi. Ilyen esetekben előnyös, hogyha a gyakran előforduló szimbólumoknak rövid kódot választunk, a ritkábban előfordulóak pedig hosszabb kódot is kaphatnak.

Ennél a módszernél fontos kérdés, hogy mekkora a kódoláshoz szükséges kódonkénti minimális bitszám. Az elméleti minimális kódolási határt azonban függetlenül kódolt szimbólumokkal nem lehet elérni, mert azok szinte mindig törtszámú bitet tartalmaznak.

A Huffman-kódolással azonban nagyon jól megközelíthető az elméleti minimum határ, a kód matematikailag bizonyíthatóan optimális. A kódolás lényege az, hogy a különböző gyakoriságú karakterekhez különböző hosszúságú bitmintát rendelünk. Így például előfordulhat, hogy az "e" betűt csak két biten, míg az "x"-et tizenkét

biten tároljuk. Az algoritmus garantálja az optimalitást, azaz a tizenkét biten kódolt "x" nem okoz akkora veszteséget, mint amennyit nyerünk az "e" két biten történő kódolásával.

Kódolás előtt természetesen először meg kell állapítanunk a karakter-előfordulási gyakoriságokat, így a tömörítendő állományt a kódolás során kétszer kell végigolvasnunk (gyakoriságok meghatározása, majd kódolás). Maga az algoritmus nem túlságosan bonyolult, de jelen jegyzet kereteibe nem fér bele.

A Huffman-kódoláson kívül még számos tömörítési algoritmus létezik, közöttük olyan is, amelyik csupán egyszer olvassa végig a tömörítendő adathalmazt, azonnal kódol. Az efféle algoritmusok ismertetése azonban nem tartozik e tárgy témakörébe.

1.9.2.3. Környezetfüggő kódolás

A valós adathalmazok sajátossága, hogy az adatsorozatban egymás után következő szimbólumok nem függetlenek egymástól, más szóval egy meghatározott szimbólumot más-más valószínűséggel követhetnek a különböző szimbólumok. A tömörítési algoritmus során minden egyes szimbólumra készítenünk kell egy táblázatot, amely azt tartalmazza, hogy aktuális szimbólumunkat milyen valószínűséggel követik a különböző szimbólumok. Ha szimbólumok és követőik között szoros korreláció áll fenn, akkor ez a módszer jelentős megtakarítást eredményezhet. Ilyen környezetfüggő kódolás például a Baudot-féle kódolás, melyet nem ismertetünk.

1.9.3. Adatbiztonság és adatvédelem

Biztonsági szolgálatok a megjelenítési rétegben:

- Az adatok illetéktelen személyek általi olvasásának megakadályozása.
- Annak megakadályozása, hogy illetéktelenek az adatfolyamba adatokat szúrhassanak be, vagy töröljenek ki.
- Minden egyes üzenet küldőjének ellenőrzése és azonosítása.
- A felhasználók számára digitálisan hitelesített üzenetküldés biztosítása.

Ezeket a célokat titkosítással lehet megvalósítani. A titkosítást az OSI-modell három rétegében lehet elvégezni.

Amennyiben a titkosítást a fizikai rétegben végzik, akkor titkosítási hardver egységet építenek be a számítógép és a fizikai közeg közé. Ez az egység végzi a kimenő adatbitek kódolását, illetve a beérkező adatok dekódolását. Ez a kapcsolattitkosításnak (link encryption) nevezett módszer egyszerű, villámgyors, de rugalmatlan, a kódoló algoritmus megváltoztatása nehézkes és költséges. A megoldás előnye viszont az, hogy a nemcsak a tényleges adatok, hanem a fejrészek és a farokrész is titkosításra kerül, ezáltal az adatblokk mérete is titkosított. Ilyenfajta titkosításra azonban az átlagos felhasználók között nincs szükség, ezért a titkosítás általában a felsőbb rétegekben kerül megvalósításra.

Ha a titkosítási funkciót a szállítási rétegbe építjük bele, akkor a teljes viszony titkosítva lesz.

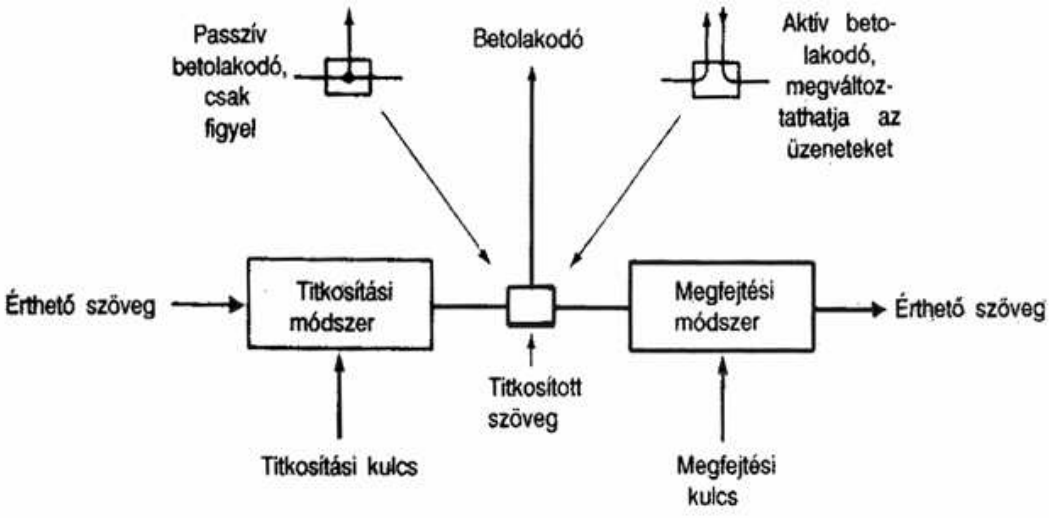
Ennél kifinomultabb megoldás az, amikor a titkosítást a megjelenítési rétegben valósítjuk meg, mert ilyenkor csak az adatstruktúrák kerülnek titkosításra, s így a titkosításra fordított többletkapacitás minimalizálható. Ezzel egyidőben a titkosítás sebessége is növekszik valamelyest.

1.9.3.1. Titkosítás

Az adatok titkosításának célja, hogy arra illetéktelen személy ne olvashassa el vagy változtathassa meg a továbbított adatokat.

A legtöbb titkosítási algoritmus alapgondolatát a következő. A titkosítandó üzenetet, melyet nyílt szövegnek (plain text) neveznek, egy kulccsal (key) parametrizálható függvény segítségével átalakítják. Az így kapott titkosított szöveget (kriptogramot) továbbítjuk a másik félnek, melyet az ugyanazon kulcs segítségével vissza tud fejteni. A kulcs általában egy rövid szimbólumfüzér, mely kiválasztja a több lehetséges titkosítási eljárás közül az aktuálisat. A kulcsot sokkal gyakrabban cserélhetjük, mint az általános módszert, így a kulcs alapvető fontosságú a titkosításban. Ennek az alapötletnek a vázlatát mutatja be a 1.55. ábra.

Az egyszeri kulcs (one time key) titkosítás lényege, hogy kulcsnak egy véletlen, a titkosítandó adathalmazzal megegyező hosszúságú bitmintát választunk, majd azt bitenként valamilyen logikai kapcsolatba hozzuk a titkosítani kívánt üzenettel. Ez a titkosítási módszer semmiféle információt nem hordoz az azt feltörni szándékozó számára, így tulajdonképpen megfejthetetlen, de ugyanakkor sok hátránya is van.



1.55. ábra. A titkosítási modell

Például mindkét félnek rendelkeznie kell a kulcs egy példányával, a visszafejtés nagyon érzékeny az adatküldés közben képződő esetleges hibákra, illetve kulcsok szinkronizálására a két oldalon.

A titkosítási módszerek két kategóriába sorolhatók.

A *helyettesítéses rejtjelezés* lényege, hogy az egyes szimbólumokat vagy szimbólum-csoportokat más szimbólumokkal vagy szimbólum-csoportokkal helyettesítjük, de sorrendjüket nem változtatjuk meg. A helyettesítéses rejtjelezést szokás egy úgynevezett S-dobozzal (substitution) modellezni.

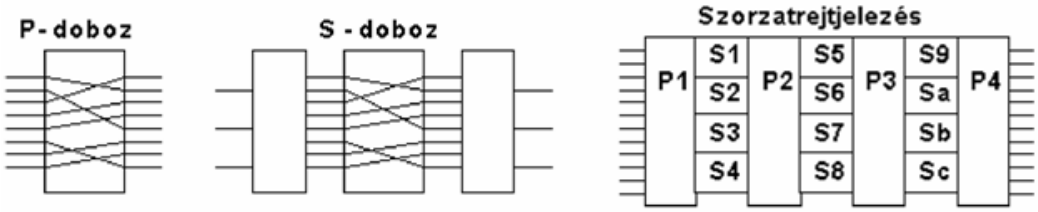
A *felcseréléses rejtjelezésnél* viszont megváltoztatják a szimbólumok sorrendjét, de nem álcázzák azokat. A felcseréléses rejtjelezés gyakran használt modellje a P-doboz (permutation).

A doboz-modellek a hardveres úton történő megvalósításra utalnak. A *szorzat-rejtjelezések* csoportjába soroljuk azokat a titkosítási módszereket, amelyek során a helyettesítéses és felcseréléses módszereket együttesen alkalmazzuk.

A két talán legismertebb titkosítási algoritmus a DES és az RSA.

1.9.3.1.1. A DES titkosítási algoritmus.

A DES (Data Encrypting Standard, adattitkosítási szabvány) az IBM által 1977 januárjában megalkotott szorzat típusú kódolási szabvány, amely eredeti



1.56. ábra. A DES titkosítási modell

formájában ma már nem biztonságos, de kis módosítással biztonságossá tehető. Az algoritmus szimmetrikus, azaz a kódolás és a dekódolás ugyanazzal a kulccsal végezhető. A kódolás 19 különálló fokozatból épül fel. Ennek során a kódolandó szöveget 64 bites blokkokra bontjuk, majd minden egységet egy 56 bites kulccsal kódolunk. A kódolt blokkok mérete ugyancsak 64 bit. Az 56 bites kulcs határozza meg, hogy az egyes lépésekben milyen S- és P-dobozokon keresztül haladjon a kódolandó információ. Az S- és P-dobozok természetesen leírhatók logikai függvényekkel, ezáltal a kódolás és a dekódolás is megoldható szoftver eszközökkel. Maga a kódolás roppant gyors, ezért nagy előszeretettel alkalmazták. Az 56 bites kulcs ma már nem elegendő megfelelő biztonsági szint eléréséhez (a megfejtéshez kellően nagy gépkapacitás mellett elegendő néhány óra), de például egy 112 bites kulcsot alkalmazva - mai gépkapacitások mellett - a dekódolás több millió évig tartana.

1.9.3.1.2. Az RSA titkosítási algoritmus.

Az RSA (a kitalálók vezetékneve: Rivest, Shamir, Adleman) egy prímszám-generáláson alapuló nem szimmetrikus algoritmus, amelynél a kulcs egyben a feladó személyét is azonosítja. Az algoritmus röviden a következő:

1. Válasszunk 2 nagy prímszámot:
   p-t és q-t (lehetőleg 10^{100}-nál nagyobbak legyenek).
2. Számoljuk ki az n = p*q és z=(p-1)*(q-1) számokat.
3. Válasszunk egy z-hez relatív prímet, jelöljük d-vel.
4. Keressünk egy olyan e számot, melyre e*d=1 mod z.
   A fenti paraméterek előzetes meghatározása után megkezdhetjük a titkosí-

tást. A kódoláshoz az e és n számokra van szükség, míg a dekódoláshoz a d-re és n-re.

5. A titkosítando szöveg legyen P, a titkosított pedig C, akkor

   $C = P^e \ mod \ n$

6. mig a dekodolás

   $P = C^d \ mod \ n$

A módszer biztonsága a nagy számok faktorizálásának nehézségén alapszik. Ha a támadó képes lenne szorzatalakra hozni a mindenki által ismert n számot, megkapná a p-t és q-t, ami alapján z-t kiszámolhatná. A z érték ismeretében pedig az e és d meghatározása euklideszi algoritmussal elvégezhető. Szerencsére azonban az utóbbi 300 év próbálkozásai alapján a matematikusok többsége azon a véleményen van, hogy a nagy számok prímtényezős felbontása rendkívül nehéz feladat. Egy 200 bites szám számítógépes faktorializálása körülbelül 4 milliárd évet venne igénybe, egy 500 bitesé pedig hozzávetőlegesen 10^{25} évig tartana. Az RSA-kódolás (és dekódolás) szoftveres úton történik, a kódolási idő valamivel hoszszabb, mint DES esetén. A módszer olyannyira biztonságos és hatékony, hogy napjainkban ez a legelterjedtebb titkosítási algoritmus.

1.9.4. Szolgálati primitívek

A megjelenítési réteg viszonya a szomszédos entitásokhoz nagyon hasonló a viszonyrétegéhez, így az általuk használt szolgálati primitívek sem különböznek sokban. A viszony entitás és a megjelenítési réteg közötti hasonlóság következtében szinte az összes megjelenítési szolgálati primitív közvetlenül a viszonyréteg hasonló primitívjét használja fel. A megjelenítési réteg szolgálati primitíveit és azok feladatait az alábbi táblázatban foglaltuk össze.

	K	B	V	M	
Primitívek					Funkció
Összeköttetés alapú					
P-CONNECT	x	x	x	x	Összeköttetés létesítés
P-U-ABORT	x	x			A felhasználó általi hirtelen bontás
P-P-ABORT		x			A szolgáltató általi hirtelen bontás
P-DATA	x	x			Normál adattovábbítás
P-EXPEDITED-DATA	x	x			Sürgős adat továbbítás
P-TYPED-DATA	x	x			Sávon kívüli adattovábbítás
P-CAPABILITY-DATA	x	x	x	x	Vezérlőinformáció adat továbbítás
P-TOKEN-GIVE	x	x			Vezérjel-átadás
P-TOKEN-PLEASE	x	x			Vezérjrl kérés
P-CONTROL-GIVE	x	x			Az összes vezérjel átadása
P-SYNC-MAJOR	x	x	x	x	Fő szinkronizációs pont beszúrása
P-SYNC-MINOR	x	x	x	x	Mellék szinkronizációs pont beszúrása
P-RESYNCHRONIZE	x	x	x	x	Vissza az előző szinkronizációs ponthoz
P-ACTIVITY-START	x	x			Tevékenység indítása
P-ACTIVITY-END	x	x	x	x	Tevékenység vége
P-ACTIVITY-DISCARD	x	x	x	x	Tevékenység eldobása
P-ACTIVITY-INITERRUPT	x	x	x	x	Tevékenység felfüggesztése
P-ACTIVITY-RESUME	x	x			Tevékenység folytatása
P-U-EXCEPTION-REPORT	x	x			Felhasználói kivétel bejelentés
P-P-EXCEPTION-REPORT		x			Szolgáltatói kivétel bejelentés
Összeköttetés mentes					
P-UNITDATA	x	x			Összeköttetés mentes adat továbbítás

A P-U-EXCEPTION-REPORT, a P-P-EXCEPTION-REPORT és a P-ALTER-CONTEXT primitívek lehetővé teszik a felhasználó számára, hogy bármilyen összetett adatstruktúrát kezelhessen.

Az alkalmazások által igényelt adatstruktúrákat csoportokba, úgynevezett kontextusokba lehet osztani. Egy viszonyon belül több csoportra is szükség lehet, az említett primitívek a viszonyon belüli kontextuscserét teszik lehetővé. Kontextusmenedzselésnek nevezzük azt a folyamatot, melynek során a felhasználók megegyeznek, hogy mely adatstruktúrák mely kontextusba kerüljenek. A szabványos eljárásnál az egyik felhasználói folyamat az általa használni kívánt adatstruktúrakönyvtárakat (az összes adatstruktúra definícióját) átadja a másiknak. A másik fél vagy elfogadja vagy elutasítja ezeket. Az aktuális kontextust bármelyik fél

bármikor megváltoztathatja. A megjelenítési réteg egy időben több kontextust is fenntarthat, hogy gyorsítsa a kontextusok közötti váltást.

1.9.5. Összefoglalás

A megjelenítési réteg az adatstruktúráknak és azok külső reprezentációinak kezelésével foglalkozik. Lehetővé teszi, hogy különböző belső felépítéssel rendelkező számítógépeket vagy alhálózatokat kössünk össze. Ezzel egyidőben elvégzi az esetleges adattömörítést, illetve gondoskodik a hálózati védelemmel és biztonsággal kapcsolatos kérdésekről. Ennek megfelelően támogat különböző titkosítási módokat.

1.10. Az alkalmazási réteg

1.10.1. Bevezetés

Az alkalmazási rétegben vannak azok a felhasználói (alkalmazási) programok, amelyek a tényleges munkát végzik. Ezek a programok használják a megjelenítési réteg által nyújtott szolgálatokat.

Az alkalmazási réteg számos kommunikációs protokollt támogat. Például több száz inkompatibilis terminéltípus létezik ma a világon. Elképzelhetjük, hogy milyen is lehet egy olyan teljes képernyős, hálózaton működő szövegszerkesztő megírása, amelynek akár több tucat különböző képernyő-elrendezéssel, szövegbeírásra, törlésre, kurzormozgatásra való escape-szekvenciával stb. rendelkező terminéltípus kezelésére kell felkészülnie.

Bizonyos alkalmazások azonban (például az állománytovábbítás) annyira elterjedté váltak, hogy működésük definiálására szabványokat hoztak létre.

Az alkalmazási réteg legfőbb feladatai közé tartozik az állománytovábbítás, a virtuális terminál, az elektronikus levelezés, valamint az érvényesítés, konkurencia és visszaállítás megvalósítása.

1.10.2. Állománytovábbítás

Az állománytovábbítás (File Transfer Access and Management, FTAM) protokoll segítségével lehetőség nyílik számítógépek közti állománymozgatásra. Lehetővé válik például az, hogy adatainkat egy távoli szerveren tároljuk, s azt egy FTAM-alkalmazás segítségével bármikor egy másik számítógépre másoljuk. Elképzelhető például, hogy a magyarországi számítógépen tárolt információkat Japánban lehívjuk, s legújabb termékeink katalógusát kissé módosítva átnyújtsuk japán üzletfeleinknek.

1.10.3. Virtuális terminál

A virtuális terminál a fentebb vázolt szövegszerkesztő-probléma megoldására született.

Terminálnak nevezzük a kijelzőből és adatbeviteli egységből álló berendezést, amely adattárolásra alkalmatlan, csupán egy központi számítógép felől érkező információkat jeleníti meg, illetve továbbítja a bevitt adatokat a központi egység felé.

Az OSI szabványosított terminálkezelő protokollja a Virutal Terminal (VT). A protokoll úgynevezett escape-szekvenciákat (escape sequences) definiál a kurzor mozgatására, a karakterek beszúrására, törlésére stb. Ezen egységes protokoll segítségével lehetőség nyílik egy rendszeren belül többféle terminál kezelésére.

A VT protokoll segítségével távoli állományokhoz férhetünk hozzá, azokat módosíthatjuk, törölhetjük, hozzáférhetünk a távoli gép erőforrásaihoz. Előző példánknál maradva VT segítségével az imént módosított termékkatalógusunkat egy Japánban kiadott paranccsal azonnal kinyomtathatjuk tízezer példányban magyarországi székhelyünk nagyteljesítményű nyomtatóján.

Lényeges különbség az FTAM és a VT között, hogy míg FTAM-mel az állományokat fizikailag is saját számítógépünkre másoljuk (vagy éppen fordítva: saját állományainkat másoljuk át a központi gépre), addig VT-vel az állományokat a távoli gépen módosítjuk, azok tényleges fizikai valójukban mindvégig a központi gépen maradnak.

1.10.4. Érvényesítés, konkurencia és visszaállítás

A számítógép hálózatoknak lehetővé kell tenniük több felhasználó egyidejű együttműködését. Ennek megvalósítását szolgálja a CCR (Commitment, Concurrency and Recovery), amely több résztvevő üzembiztos működését hangolja össze, még ismétlődő rendszerösszeomlások esetén is.

CCR jellegű probléma például az elektronikus úton történő banki átutalás egyik számláról a másikra. Ilyen tranzakciónál hiba léphet fel akkor, ha a központból kiadott utasításokat csak az egyik bankban tudja végrehajtani. Ebben az esetben a bankok adatállománya inkonzisztenssé válna. Ennek megakadályozásához olyan protokollra van szükség, mely vagy sikeresen elvégzi mindkét műveletet, vagy egyáltalán nem csinál semmit.

A CCR ezt a problémát úgynevezett elemi tevékenységek (atomic action) kezelésével oldja meg. Egy elemi tevékenység üzenetek és műveletek olyan halmaza,

amelyek vagy sikeresen végrehajtódnak vagy az eredeti állapotukba visszaállíthatóak, mintha egyetlen tevékenység sem következett volna be.

Az elemi tevékenységeket kétfázisú érvényesítéssel (two-phase commit) valósítják meg.

Az első fázisban a mester közli a szolgákkal a teendőket, az egyes szolgák pedig ellenőrzik, hogy eleget tudnak-e tenni a kérésnek. Ha igen, akkor a kérést feljegyzik egy fokozott biztonságú tárolóba (stable storage), végül a sikeres végrehajtást nyugtázzák a mester felé. Ha a szolga nem képes végrehajtani a kérést, akkor hibát jelez vissza és a továbbiakban nem csinál semmit.

Amikor az összes válasz megérkezik a mesterhez, akkor az ellenőrzi, hogy minden szolga végre tudja-e hajtani a feladatát. Ha igen, akkor - második fázisként - mindegyiknek érvényesítés (commit) parancsot küld, amellyel megadja a jelet a tényleges végrehajtásra.

Mivel az adatvesztési hibákat az alsóbb rétegek lekezelik, ezért a CCR-nek csupán a szolga esetleges összeomlásából keletkező hibát kell tudnia kezelni, és helyreállítania. A hiba javítása azért lehetséges, mert a fokozott biztonságú tárolón feljegyzett kezdeti állapot és végrehajtás után elvárt állapot összehasonlításából detektálható, az esetleges összeomlás.

Ha bármelyik szolga hibát jelent az első fázisban, akkor mester megszakítja a tevékenységet és utasítja az összes szolgát, hogy szabadítsák fel zárolt adataikat és állítsák vissza kezdeti állapotukat.

A CCR a kétfázisú érvényesítés minden egyes alapvető tevékenységéhez primitíveket biztosít. A C-RESET primitívre akkor van szükség, amikor egy összeomlás után a mester vagy valamelyik szolga újraindul. Célja az, hogy a kezdeti állapotba állítsa vissza a tevékenységet, s így lehetőséget biztosítson az újrakezdéshez.

Az alkalmazási réteg szolgálati primitíveit és azok feladatait a következő táblázatban foglaljuk össze.

Művelet	Állomány-művelet	Rekord-művelet	Bittérkép	Leírás
Create-file	x		x	Állomány-létrehozása
Delete-file	x			Állomány-törlése
Select-file	x			Állomány-kijelölése
Deselect-file	x			Kijelölés-megszüntetése
Open-file	x			Állomány-megnyitása-olvasásra-vagy-módosításra
Close-file	x			Állomány-lezárása
Read-attribute	x		x	Állományattribútum-kiolvasása
Change-attribute	x		x	Állományattribútum-módosítása
Locate		x		Egy-rekord-lokalizálása
Read		x	x	Adat-olvasása-az-állományból
Insert		x	x	Adat-beillesztése-az-állományba
Replace		x	x	Adat-felülíráse
Extend		x	x	Rekord-bővítése-adattal
Erase		x	x	Rekord-türlése

1.10.5. Elektronikus levelezés

Az ARPANET megjelenése után a várakozásokkal ellentétben nem a folyamatok közötti kommunikáció tette ki az adatforgalom legnagyobb részét, hanem a felhasználók közt zajló elektronikus levelezés.

Bár az elektronikus levelezés az állománytovábbítás egy speciális formájának tekinthető, speciális jellemzői megkülönböztetik az állománytovábbítástól. Emiatt az elektronikus levelezési rendszereket két részből építik fel. Az egyik rész a felhasználói interfészt biztosítja az üzenetek megszerkesztésére és olvasására, a másik rész pedig az üzenetek továbbításáért felelős.

Az elektronikus levélüzenetek abban is különböznek az adatállományoktól, hogy mindig tartalmaznak az üzeneten kívül más mezőket is, mint például a küldő neve és címe, a fogadó neve és címe stb.

Az alkalmazási réteg szabványa (Message Oriented Text Interchange Standard, MOTIS) azonban az eddig említetteknél sokkal több problémával foglalkozik. Ezen problémák közé tartozik például a katalógusszolgálat (mely a telefonkönyv elektronikus megfelelője), a távoli munkabevitel, a grafika megvalósítása, a társulásvezérlés a konkurenciavezérlés stb.

1.10.6. Összefoglalás

Az alkalmazási réteg a felhasználói szintű problémák megoldásáért felelős. Például kezelnie kell minden olyan problémát, mely akkor léphet fel, ha ugyanahhoz az állományhoz egy időben több felhasználónak kell hozzáférnie. Hasonló fontosságú feladata az elektronikus levelezés biztosítása. Ezenkívül virtuális terminálok biztosításával lehetővé teszi a terminháltól független felhasználói programok megírását.

1.11. A TCP/IP hivatkozási model

1.11.1. Az ARPANET

A múlt század hatvanas éveinek közepén, a hidegháborús időszak csúcsán az amerikai Védelmi Minisztérium (DoD) egy atomháborút is túlélni képes információs hálózatot szeretett volna. Az ARPA-t a Szovjetunió által 1957-ben fellőtt Szputnyik műholdra tett válaszlépésként hozták létre, a célja az volt, hogy ösztönözze a hadiiparban hasznosítható technológiák fejlődését. Tehát mint sok más fejlesztés, ez is eredetileg katonai célokat szolgált.

Az ARPANET egy hálózatokból és hosztokból álló csomagkapcsolt hálózat volt. Az alhálózat átviteli vonalakkal összekapcsolt miniszámítógépekből, ún. csomóponti gépekből (Interface Message Processors, IMP) épült föl. A nagyobb megbízhatóság érdekében a csomóponti gépeknek legalább két másik géphez kellett csatlakoznia.

Kezdetben négy csomópontja volt, majd egyre több egyetem is csatlakozott. Kidolgoztak egy protokollt, hogy egy csomóponti számítógéphez (Terminal Interface Processor, TIP) csatlakozva terminálok is csatlakozhassanak a hálózathoz. Ahogy bővült a hálózat, egyre világosabbá vált, hogy az ARPANET protokolljai nem igazán megfelelőek több hálózatból álló rendszerek esetén. Ez az észrevétel a protokollok további fejlesztéséhez vezetett, aminek csúcspontja a TCP/IP modell és a TCP/IP protokollok kifejlesztése volt.

1983-ra a több mint 200 csomóponti gépet és hosztokat tartalmazó ARPANET már stabil és igencsak sikeres volt. Ekkor leválasztották róla a katonai részt, amivel egy külön alhálózat jött létre MILNET néven. A MILNET és a megmaradt kutatói hálózat közé szigorúan ellenőrzött átjárókat építettek be. Az 1980-as években további hálózatokkal, elsősorban LAN-okkal bővült az ARPANET.

Ahogy a gépek száma nőtt, egyre költségesebbé vált egy bizonyos hoszt megkeresése, ezért létrehozták a DNS (Domain Name System) rendszert, amelynek célja, hogy a gépeket doménekbe szervezze, és a hosztok neveit leképezze IP címekre. 1990-re az ARPANET-et megelőzték újabb hálózatok, olyanok, amelyek pont belőle fejlődtek ki. Így az ARPANET elhalt, szétdarabolódott és részben betagozódott az Internetbe.

1.11.2. Az NSFNET és a nagy sebességű hálózatok

Az 1970-es évek végére az amerikai egyesült államokbeli Nemzeti Kutatási Alap (U. S. National Science Found, NSF) felismerte, hogy az ARPANET-nek óriási hatása van az egyetemi kutatásokra, és ez lehetővé tette, hogy a kutatók országszerte hozzáférhessenek bármilyen adathoz, és közös projekteken dolgozhassanak. 1984-ben az NSF az ARPANET-ből kiindulva elkezdett kifejleszteni egy olyan nagy sebességű hálózatot, amely minden egyetemi kutatócsoport számára nyitott volt. A gerinchálózat gépeit kezdetben 56 kb/s-os bérelt telefonvonalak kötötték össze, 1990-re már 45 Mbit/s sebességet is el tudtak érni, de a cél az volt, hogy az ezredfordulóig a 3 Gb/s sebességet elérjék. Később minden európai ország is létrehozott legalább egy olyan országos hálózatot, amely az NFS regionális hálózatokhoz hasonlít.

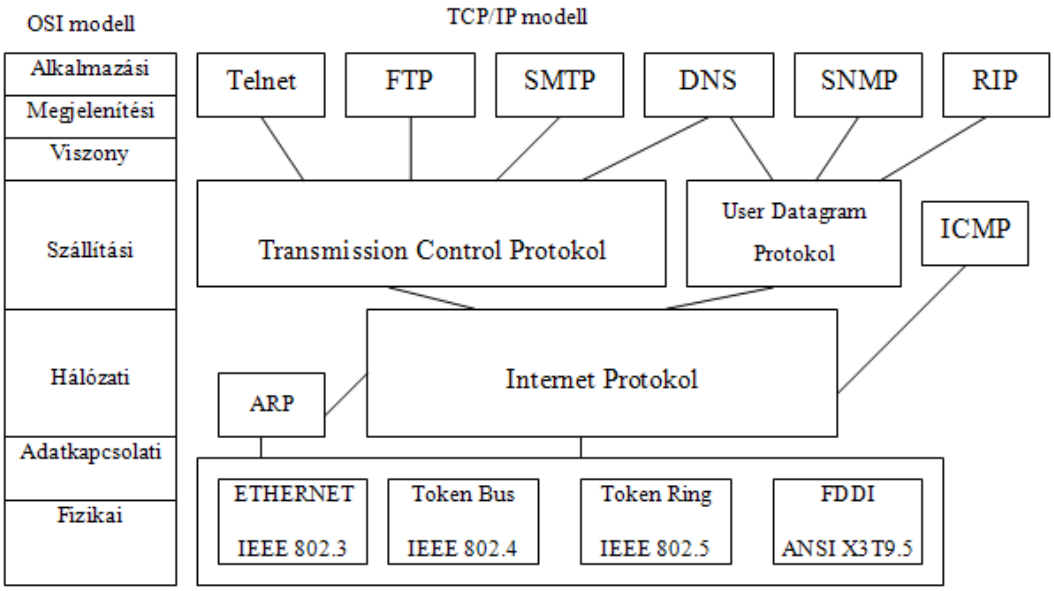
1.11.3. Az Internet

Miután 1983. január 1-jétől az ARPANET hivatalos protokollja a TCP/IP lett, a kapcsolódó hálózatok, gépek és felhasználók száma ugrásszerűen megnőtt. Valamikor az 1980-as évek közepe táján a sok egymáshoz kapcsolódó hálózatot egy Internetnek kezdték tekinteni, így született meg az Internet.

A fejlődés exponenciális jelleggel folyt tovább, 1990-re már közel 3000 hálózatot és 200000 számítógépet foglalt magába. 1992-re a hosztok száma elérte az egymilliót. 1995-re számos gerinchálózat, több száz középszintű hálózat, több tízezer LAN, több millió hoszt és több tízmillió felhasználó alkotta az Internetet. Ez a szám napjainkra a '95-ös érték többszörösét is meghaladja és a fejlődés természetesen nem áll meg.

Az Internetet a TCP/IP hivatkozási modell és a TCP/IP protokollkészlet tartja össze. A TCP/IP egy olyan egységes szolgáltatást tesz lehetővé világszerte, amelyet leginkább a telefonhálózathoz vagy a 19. században egységesített vasúti nyomtávhoz lehetne hasonlítani. A definíció alapján egy gép akkor kapcsolódik az Internetre, ha ismeri a TCP/IP protokollt, van IP címe, illetve az Interneten lévő bármely másik gépnek képes IP csomagot küldeni. A valóságban ez nem ennyire egyszerű, ugyanis nagyon sok személyi számítógép képes arra, hogy modem segítségével felhívjon egy Internet-szolgáltatót, amelytől kap egy ideiglenes IP címet, és

más internetes hosztoknak máris küldheti ez IP csomagokat. Amíg ezek a gépek az Internet szolgáltató útválasztójához (routeréhez) kapcsolódnak, joggal tekinthetjük őket is úgy, mintha az Interneten lennének.



1.57. ábra. Az osi modell és a TCP/IP protokoll készlet összehasonlitása

1.11.3.1. A TCP/IP protokoll architektúra áttekintése

FELHASZNÁLÓI RÉTEG	HTTP, FTP, Telnet, Finger SSH, DNS, POP3/IMAP SMTP, Gopher, Time/NTP BGP, Whois, TACACS+, SSL	DNS, SNMP RIP, RADIUS Archie, tftp Traceroute	Ping		
SZÁLLÍTÁSI RÉTEG	TCP	UDP	ICMP	OSPF	
INTERNET RÉTEG	IP				ARP
HÁLÓZATI-FIZIKAI ESZKÖZ RÉTEG	Ethernet/802.3, Token Ring (802.5), SNAP/802.2, X.25, FDDI, ISDN Frame Relay, SMDS, ATM, Wireless (WAP, CDPD, 802.11) Fibre Channel, DDS/DS0/T-carrier/E-carrier, SONET/SDH, DWDM PPP, HDLC, SLIP/CSLIP, xDSL, Cable Modem (DOCSIS)				

1.11.4. Ethernet

A korai 1970-es évek végén kifejlesztett Ethernet bizonyult a valaha tervezett egyik legegyszerűbb, legmegbízhatóbb és leghosszabb életű protokollnak. Habár az Et-

hernet nagyon sokféle eszközön használható a fizikai rétegben, a három legelterjedtebb a 10BaseT, 10Base2, és a 10BaseF vagyis UTP, koaxiális és üvegszálas kábelek.

Az UTP-t csillagpontos elrendezésben használjuk, melyben az összes csomópont egy központi hub-ra csatlakozik. A 10Base2 egyetlen koaxiális kábelen keresztül -sorosan- kapcsolja össze a munkaállomásokat, és nincs szükség hub-ra. A 10BaseF üvegszálas kábel amellett hogy drága, képes nagy távolságokban lévő és/vagy elektromosan zajos területeken keresztül összekapcsolni csomópontokat.

Érdekes különbség a koaxiális Ethernet és más Ethernet típusú hálózatok között, hogy a koaxiális tényleges point-to-multipoint kapcsolat, az UTP vagy az üvegszálas viszont - a fizikai réteg szempontjából - csak point-to-point, és szüksége van egy további hálózati eszközre (repeater, hub) ami összekapcsolja a munkaállomásokat.

Ha az Ethernetet jellemezni kéne: sok ember hangoskodik egy kis tárgyalóteremben, az emberek mindig csak akkor beszélhetnek, ha éppen csend van a szobában. Ha két ember feláll és elkezd kiabálni egy időben, akkor úgymond "ütközést" idéznek elő, egymást mintegy semlegesítve.

Ütközés esetén a két szembenálló, egyidőben kiabáló ember szépen visszaül egy időre, hogy annak letelte után valamelyik újból próbálkozzon. Ezeknek az ütközéseknek a valószínűsége mértani sorozat szerint növekszik, a beszélők és a beszéd mennyiségétől függően. Valójában azok a hálózatok már túlterheltek, amelyeken a szegmens kihasználtság túllépi 30-40%-ot.

Az Ethernet hub-okat 10BaseT hálózatokban használják. Egy szabványos hub tulajdonképpen csak egy buta repeater (jelerősítő), minden kapott adatot megismétel a többi portján.

A 100 Mbit Ethernet (100BaseTX) ugyanúgy működik mint a 10 Mbit Ethernet, csak 10-szer gyorsabb. A csúcsminőségű rézen (avagy Category 5 illetve CAT 5 UTP néven is ismert), 100BaseTX ugyanazt a két pár rézvezetéket használja kommunikációra, ellentétben a 10BaseTX-el ahol egy párat adásra és egy párat vételre használ (Ethernet kártya adás közben figyeli, hogy nem ad e valaki más is közben - Carrier Sense Multiple Access, Collision Detection).

A gigabit Ethernet a 100 Mbit Ethernetnél 10-szer gyorsabb, működésében ugyanaz, szélessávú internetszolgáltatók gerinchálózatainál használatos.

Ha a társalgószobánk túl hangos lesz, feloszthatjuk a társaságot két szobára, melyek közé egy olyan személyt teszünk, aki mindkét szobát figyeli, meghallgatja, megjegyzi ki-mit mond (adatok és Ethernet kártyacímek), majd továbbítja az üzeneteket a másik szobába. Ezt az eszközt "transparent bridge" -nek nevezzük, intelligens eszköz, működését a munkaállomások nem észlelik.

Az Ethernet switchek kicsit többek egy nagysebességű, többportos hídnál. Megjegyzik az összes rájuk kapcsolódó Ethernet kártya címét, és ezek alapján eldöntik, hogy melyik adatot merre továbbítják. A 10 - 100Mbites hálózatok közötti kommunikációhoz szükséges az un. buffering (adatok ideiglenes tárolása, mivel a 10Mbites hálózat lassabb), melyhez szintén switcheket érdemes használni. Általában az olcsóbb swicheken is találunk 1-2 100Mbites portot a többi 10Mbites mellett, amely előbbikre a szervereket, utóbbiakra pedig a munkaállomásokat csatlakoztatjuk.

1.11.5. Internet Protocol (IP)

Az IP egy csomagkapcsolt, datagramm jellegű, megbízhatatlan hálózati protokoll. A szállított csomagok, a datagrammok, amely a forrás hoszttól a cél hosztig kerülnek továbbításra, akár több hálózaton is keresztül. A hálózati réteg megbízhatatlan, összeköttetés-mentes szolgálatot biztosít, azaz a csomagok elveszhetnek, többszöröződhetnek, hibás sorrendben érkezhetnek; így az összes megbízhatósági mechanizmust a szállítási rétegben kell megvalósítani.

Az IP csomag fejléce a következő mezőket tartalmazza:

Verzió a használt IP verziószámát adja (Általánosan 4, de létezik 6-os verzió is, ez az IPv6. 6-os formátuma nem egyezik meg a 4-es verzióval, és még nincs is bevezetve)

IHL (Header Length) megadja a fejléc hosszát 32 bites szavakban, ez minimum 5, azaz 20 byte.

Szolgálattípus (Type of Service, TOS) a továbbítás típusát határozzák meg. A mező két részből áll. 3 bit a csomag fontosságát határozza meg, azonban ez

<table>
<tr><td colspan="32"><——— 32 bit ———></td></tr>
<tr><td colspan="32">1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3</td></tr>
<tr><td colspan="32">0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1</td></tr>
<tr><td colspan="4">Verzió</td><td colspan="4">IHL</td><td colspan="8">Szolgálat tipus</td><td colspan="16">Teljes hossz</td></tr>
<tr><td colspan="16">Azonosítás</td><td colspan="3">Flag</td><td colspan="13">Tördelési eltolás</td></tr>
<tr><td colspan="8">Élettartam</td><td colspan="8">Protokoll</td><td colspan="16">Fejresz ellenőrző összeg</td></tr>
<tr><td colspan="32">Forráscím</td></tr>
<tr><td colspan="32">Célcím</td></tr>
<tr><td colspan="32">opciók</td></tr>
<tr><td colspan="32">TCP fejrész, adatok</td></tr>
</table>

1.7. táblázat. Az IP protokoll felépítése

csak a lokálisan értelmezendő. Két bit foglalt, a fennmaradó 3 bit valamelyikének (vagy mindegyikének) beállításával kérheti a csomag feladója, hogy azt rendre kisebb késleltetésű és/vagy nagyobb sávszélességű és/vagy nagyobb megbízhatóságú útvonalon keresztül továbbítsa a hálózat, amennyiben választási lehetőség adódik. (Lásd lent) Minthogy a specifikáció nem követeli meg ezeknek a biteknek a figyelését, a jelenlegi Internetben nem sokat számít a TOS bitek beállítása.

Teljes hossz a csomag hosszát byteokban, így a datatgram maximális hossza 64 kilo oktet (Kbajt)lehet, azonban ilyen nagy csomagokat nem szoktak használni, mert nem praktikus.

Azonosítás egy egyedi számot tartalmaz, ami a fragmentáció során azonosítja a csomagot.

Flagek és

Tördelési-eltolás vezérli a fragmentálást és az összegyűjtést

Élettartam , **TTL** (Time To Live) mező egy másodpercben megadott érték és a csomag élettartamát jelöli. Minden hálózati berendezés köteles másodpercenként csökkenteni ezen az értéken, és ha eléri a nullát, a csomagot el kell dobni. Ezzel érjük el, hogy egy csomag ne keringjen az örökkévalóságig a hálózatban. A router-eknek akkor is csökkenteniük kell egyel ezt a mezőt, ha egy másodpercnél rövidebb idő alatt továbbítják a csomagot. Minthogy az esetek többségében ez történik, a TTL mező gyakorlatilag minden router-en

való áthaladáskor csökken egyet. (Egy ilyen áthaladást hop-nak nevezünk.) Az IP következő verziójában éppen ezért ezt a mezőt hop-count-nak (hop-számláló) nevezik (kezdő értéke operációs rendszer függő, újabban javasolt a 255 érték).

Protokoll annak a felsőbb szintű protokollnak a számát adja, amelyik számára a csomag szól.

Fejrész ellenőrzőösszeg (Header checksum)

Forráscím (Source IP address)

Célcím (Destination IP address)

Opciók (IP options) esetleges opciók (lásd lejjebb)

PRECEDENCE	D	T	R	0	0

Bits 0-2: Fontosság (Precedence)

Bit 3: 0 = Normál késleltetés 1 = Kis késleltetés

Bit 4: 0 = Normal sávszélesség 1 = Nagy sávszélesség

Bit 5: 0 = Normal megbízhatóság 1 = Nagy megbízhatóság

Bit 6-7: Fentartva későbbi használatra

TOS bitek (Type Of Services)		Késleltetés (Delay)	Sávszélesség (Throughput)	Megbízhatóság (Reliability)
Telnet		1	1	1
FTP	Vezérlés (Control)	1	0	0
	Adat (Data)	0	1	0
TFTP		1	0	0
SMTP	Parancsok (CMD)	1	0	0
	Adat (Data)	0	1	0
DNS	UDP Query	1	0	0
	TCP Query	0	0	0
	Zone Transfer	0	1	0

Az **opciók** olyan ritka, IP szintű funkciók megvalósítására szolgálnak, melyeknek nem volt érdemes a minden csomagban jelen lévő fejlécben helyet fenntartani. Az opciókat minden állomás köteles megérteni és feldolgozni, nem implementációjuk, csupán jelenlétük választható. Az RFC791 a következő opciókat definiálja:

- Security. A csomag hitelesítéséhez szükséges információk.
- Source routing. A feladó által megadott útvonalon, állomások megadott listáján halad végig a csomag. Két vállfaja van, a szigorú (strict) és a laza (loose). Az első esetben csak a listán felsorolt állomásokon haladhat végig a csomag és ha két szomszédosnak felsorolt állomás nem szomszédos, akkor

a csomag elvész és egy "Source routing failed" ICMP csomag küldődik a feladóhoz. A második esetben ha a listán két szomszédosnak feltüntetett állomás a valóságban nem szomszédos, akkor is továbbítódik a csomag a lista következő eleméhez, de a router-ek által kijelölt útvonalon.

- Útvonalrögzítés. A csomag által érintett állomások IP címe rögzül a csomagban.
- Időbélyeg
- Stream ID. Egy 16 bites azonosító, főként más, folyam (kapcsolat) orientált hálózatokkal való együttműködés segítése miatt.

1.11.5.1. Datagrammok fragmentálása és összerakása

A fragmentáció akkor következik be, ha haladási útvonalon a következő link-en az MTU kisebb, mint a csomagméret.

Ekkor a router (vagy maga a feladó) több darabra tördeli a csomagot, mindegyik darabba beleírja, hogy ez az eredeti csomag hányadik byte-jától kezdődő információkat tartalmazza és ad a csomagnak egy egyedi azonosítót. Az azonosító fragmentbe belekerül és jelzi a vevőnek, hogy mely darabok alkotják az eredeti csomagot. A router a kapott darabokat egyesével feladja és a hálózatra bízza őket. A nagyobb fragmentek esetleg később még tovább darabolódhatnak egy még kisebb MTU-jú link-en.

A vevő feladata a csomag összevárása és összerakása.

A feladó egy bit (DF=Don't Fragment bit) beállításával kérheti, hogy csomagját ne darabolják, ez esetben ha az MTU túl kicsi a csomag továbbításához, a csomagot eldobják és erről ICMP üzenetben tájékoztatják a feladót (lásd lejjebb).

Az IP csupán azt követeli meg, hogy az alatta lévő hálózat képes legyen minimum 68 byte-os IP csomagok továbbítására, ez tehát a minimálisan szükséges MTU.

A TCP/IP implementációk különböznek egymástól a datagramm méretének megválasztásában, azonban a szabvány szerint legalább 576 oktet nagyságú datagrammokat választanak, ha nem biztosak abban, hogy a nagyobb méretet útközben mindenhol megértik. Ez az eléggé konzervatív megközelítés abból fakad, hogy az összerakást megvalósító kódok sokszor hibásak. A tervezők kerülni igyekeznek a fragmentálást. Mindegyikük másként gondolkodik arról, hogy mikor biztonságos a

nagyobb méret. Néhányan csak a lokális hálózatra esküsznek, de vannak olyanok is, akik az egész hálózatra kiengednek ilyen datagrammokat. Az 576 oktet eléggé biztonságos ahhoz, hogy mindenki támogassa.

1.11.5.2. Az IP címzési rendszere

Az IP 32 bites címeket használ, a címeket byte-onkét, egymástól ponttal (".") elválasztva, tízes számrendszerben írjuk. (Például 152.66.208.40).

Minden cím egy hálózati és egy állomásszámból áll. A hálózat száma azonosítja azt a hálózatot, melyen a cél található, az állomásszám pedig azon belül magát az állomást.

Kezdetben a hálózat száma 7 bites, az állomás száma pedig 24 bites volt, minthogy kevés, de népes hálózatokra számítottak. Később ezt a címformátumot A osztályú címnek nevezve még két címosztályt vezettek be, a B és C osztályt. Ezekben nagyobb hely volt a hálózat száma részére (14 és 22 bit) és kisebb az állomás számára (16 és 8 bit), több, de kisebb hálózat számára címteret biztosítva. Sajnálatos módon a legtöbb jelenlegi hálózat számára a B osztály sok, a C osztály kevés állomás bekapcsolását teszi lehetővé, ami miatt a B osztály kimerülni látszik. Ezzel a problémával később még foglalkozni fogunk.

	<— 32 bit —>																															
											1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	3	3
Class	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
A	0	Hál. cím									Hoszt cím (HOST_ID)																					
B	1	0	Hálózati cím (NET_ID)														Hoszt cím (HOST_ID)																
C	1	1	0	Hálózati cím(NET_ID)																					Hoszt cím								
D	1	1	1	0	Többesküldés (MULTICAST_ID)																													
E	1	1	1	1	jövőbeli felhasználásra (EXPERIMENTAL_ID)																													

1.8. táblázat. Az IP cimosztályok

Az Internet Multicast számára van fenntartva a D osztály, az ebben levő 28 hasznos bit strukturálatlan és egy cím egy multicast csoportot jelöl.

Az E osztályú címek későbbi felhasználásra vannak fenntartva.

Az IP címtartományok osztályok szerint csoportosítva:

Class A ⇒ 1.0.0.0 - 126.0.0.0 7-bites NET_ID és 24-bites HOST_ID. Az A osztályú címek nagy méretű hálózatok számára vannak fenntartva, és hálózatonként 16777216 (224) hosztot tudnak megcímezni.

Class B ⇒ 128.xxx.0.0 - 191.xxx.0.0 14-bit NET_ID, 16-bit HOST_ID, ez 65536 megcímezhető hosztot jelent.

Class C ⇒ 192.xxx.yyy.0 - 223.xxx.yyy.0 21-bit NET_ID, 8 bit HOST_ID, hálózatonként 254 címet jelent.

Class D ⇒ 224.xxx.yyy.zzz - 239.xxx.yyy.zzz (multicast IP)

Az IP címekben a 0 és a 255 speciális jelentéssel bír. A 0 az olyan gépek számára van fenntartva, amelyek nem tudják a hálózati címüket. Bizonyos helyzetekben lehetséges, hogy egy számítógép nem tudja, melyik hálózatra csatlakoztatták. A 0.0.0.23 például egy olyan számítógép címe, amelynek hosztszáma 23, de nem tudni, hogy melyik hálózaton. A 255-t üzenetszórásra (broadcast) használják. Az üzenetszórás lényegében egy olyan üzenet, amelyet az adott hálózaton minden számítógép lát. Olyankor használatos, amikor a "címzett ismeretlen".

Tegyük fel például, hogy egy, a hálózatra kapcsolt számítógép nevére van szükségünk, mert az Internet címét szeretnénk tudni. Mondjuk, hogy a legközelebbi névszolgáltatónak nem tudjuk a címét. Ilyenkor segít az üzenetszórás. Előfordulhat az is, hogy egy információt több rendszerrel meg szeretnénk osztani. Ilyenkor hatásosabb az üzenetszórás, mintha az érdekelt rendszerekhez külön küldenénk datagrammokat. Az üzenetszórás megvalósításához egy olyan IP címet kell formálni, amelyben a hálózatot jelölő részbe a küldő hálózat címét, a gépet jelölő részbe pedig csupa egyes bitet (azaz 255-t) írunk. A 193.6.4 hálózaton ez így nézne ki: 193.6.4.255.

Az üzenetszórás tényleges megvalósítása az adott közegtől függ. Az Arpanet-en és két gép közötti hálózatokon nem lehet üzenetszórást alkalmazni, ellentétben az Ethernet alapú hálózatokkal, ahol a csupa egyes bitből álló Ethernet című üzenetet az azon a hálózaton lévő minden számítógép veszi. A 255.255.255.255 gyakorlatilag az egész Internet megcímzésére szolgál, de ilyen IP csomagokat a routerek nem engednek át.

Az állomások számára fenntartott mezőt gyakran két részre osztják, egyik rész az alhálózat (subnet) azonosítja, másik pedig azon belül magát az állomást.

A 127-tel kezdődő címek a visszairányítás(loopback) címei, a hálózat belső tesztelésére használhatók csak.

A 10-zel és 192-vel kezdődő IP címek mindig az aktuális alhálózatra mutatnak, és a hálózat irányító eszközök az ezekre a címekre érkezett csomagokat mindig a belsőhálózat felé küldik.

1.11.5.3. Az alhálózatok címzése (Subnet masking)

Mivel egy B típusú osztály esetén 65534 cím kiosztását jelenti egy hálózatban, ez nehezen kezelhető hálózat kialakulásához vezetne. Emiatt a hálózatokat alhálózatokra (subnet) bontják. Azt, hogy az állomás számára rendelkezésre álló biteket (B osztály esetén 16) hogyan osztják fel a hálózat és a tényleges állomás azonosítója között, a rendszer szempontjából teljesen mindegy, mindössze egy alhálózati mask (subnet mask, netmask) megadása szükséges, amely azonosítja az alhálózatot.

1. PÉLDA: Egy adott gép címe alapján, hálózatának címét keressük. A gép a balu.sch.bme.hu (152.66.208.40, B osztályú cím) és a hozzá tartozó netmask 255.255.248.0

Az IP cím és a netmask bitenkénti ÉS kapcsolatát elvégezve kapjuk a hálózat címét:

10011000 01000010 11010000 00101000	-gép címe 152.66.208.40
11111111 11111111 11111000 00000000	-netmask 255.255.248.0
_____	AND művelet
10011000 01000010 11010000 00000000	-hálózat címe 152.66.208.0

Tehát a hálózatának címe: (152.66.208.0). Ez az alhálózatban összesen 6x254=2032 hosztból állhat.

(Megjegyzés): Az IP címosztályok is meghatároznak egy alapértelmezett netmaskot: Class A - 255.0.0.0 Class B - 255.255.0.0 Class C - 255.255.255.0

Újabban létezhetnek változó hosszúságú alhálózatok is, tehát egy hálózatban lehet olyan alhálózat, melynek azonosítására az állomásszám felső 8 bitjét használjuk, és lehetnek olyanok, ahol a felső 12 bitet. Természetesen az alhálózatok számait úgy kell kiosztani, hogy a felső 8 bitből el lehessen dönteni, hogy ez most egy 8 vagy 12 bites alhálózati szám. Ez akkor lehet hasznos, ha sok eltérő méretű alhálózatunk van és fix beosztás esetén nem elegendő a címtér. A módszer

viszont egy fokkal bonyolultabb routing protokollt és adminisztrációt igényel. Egy alhálózaton levő állomások szomszédosnak számítanak, tehát router közbeiktatása nélkül tudnak kommunikálni, egy közös link-en vannak. Lehetséges azonban, hogy egy link-en több alhálózat is legyen, azonban egy alhálózat nem tartalmazhat több link-et, hiszen azok között az átjárás csak router-en át lehetséges. Ha egy link-en több alhálózat van, akkor a különböző alhálózaton lévő állomások, bár közvetlenül is kommunikálhatnának, mégis router-t vesznek igénybe.

1.11.5.4. Az IP útvonal-keresési rendszere (IP routing)

Egy IP csomag rendeltetési helyére juttatásának miként jét az útvonal-választás (routing) kifejezés jelöli. A részletek nagymértékben függenek az adott implementációtól, viszont egy-két dolgot általánosságban el lehet mondani.

Az IP alapállapotban azzal a feltevéssel él, hogy a rendszerek valamilyen lokális hálózatra kapcsolódnak. Feltesszük, hogy a rendszer a saját hálózatán keresztül csomagokat tud küldeni egy másik rendszernek. (Ethernet alapú hálózat esetén egyszerűen a célállomás Ethernet címét kell megkeresnie, majd a csomagot ki kell adnia a hálózatra.)

A probléma akkor jelentkezik, amikor egy másik hálózaton lévő rendszerhez kell küldeni a csomagot. Itt lépnek be az átjárók (gateway). Az átjáró egy olyan hálózati eszköz, amely egy hálózatot két vagy több másikkal köt össze. Ez a gyakorlatban legtöbbször egy olyan számítógépet jelent, amelynek több hálózati interfésze van. Ez a számítógép a két hálózat között átjáróként üzemelhet. A hálózati szoftvert úgy kell beállítani, hogy az átjáró a két hálózat között dcsomagokat tudjon küldeni. Ha egy gép egy hálózatról olyan csomagot küld az átjáró felé, amely egy másik hálózaton lévő gépek egyikének szól, akkor azt az átjáró továbbítja a célállomás felé.

A főbb kommunikációs központokban több átjáró is található, amelyek különböző hálózatokat kötnek össze egymással. (A legtöbbször speciálisan erre a feladatra készített átjárókat alkalmaznak, amelyek megbízhatóbban, és sokkal hatásosabban működnek az általános célú átjáróknál.)

Az IP szerinti útvonal-választás teljes mértékben a célállomás hálózati számán alapszik. A hálózatba kötött minden egyes számítógép rendelkezik egy táblázattal,

amelyben a hálózati számokat tárolják. Minden hálózatszámhoz tartozik egy átjáró, amelyen keresztül az adott hálózathoz eljuthatunk. Azt észre kell venni, hogy az átjáró nincs feltétlenül arra a hálózatra kötve, egyszerűen csak az a legjobb út, amelyen keresztül az adott hálózathoz el lehet jutni.

Amikor egy számítógép csoamgot akar küldeni egy másiknak, akkor először azt ellenőrzi, hogy a fogadó nincs-e a saját hálózatán. Ha ott van, akkor a csomagot közvetlenül neki küldi el. Ha nincs ott, akkor a rendszer keresni kezdi a táblázatban a célállomás hálózati számát, és a csoamagot annak a hálózatnak az átjárója felé küldi. A hálózati számokat és átjárókat felsoroló táblázat esetenként igen nagy terjedelemre tehet szert. Az Internet például több száz hálózatot foglal magába.

Különböző stratégiákat dolgoztak ki annak érdekében, hogy az útvonal-választási táblák méretét a lehető legkisebb értéken tartsák. Az egyik ilyen módszer az alapértelmezett útvonalak használata. Gyakran fellép az az eset, hogy egy hálózatból csak egyetlen átjárón keresztül lehet kijutni. Egy ilyen átjáró például egy Ethernet alapú lokális hálózat és egy gerinchálózat között létesíthet kapcsolatot. Ilyenkor persze nincs szükség arra, hogy az útvonal-választási táblában az összes külső hálózat szerepeljen. Az átjárót egyszerűen alapértelmezettnek definiáljuk, és így a választott útvonallal nem rendelkező csomagok egyenesen az átjáróhoz kerülnek. Egy így beállított átjáró akkor is használható, ha egy hálózaton több is működik belőle.

Az átjárókat úgy tervezték, hogy a "Nem ez a legjobb átjáró – használd inkább ezt és ezt." üzenetet generálni tudják. A hálózati szoftverek többsége ezt az üzenetet használja arra, hogy az útvonal-választási táblájába bejegyzéseket helyezzen el.

Az IP szakértők többsége azon a véleményen van, hogy a hálózati számítógépek ne próbálják meg az egész hálózat forgalmát nyomon követni. Ehelyett azt ajánlják, hogy alapértelmezett átjárókat használjanak, és rájuk támaszkodjanak az útvonalak megállapításánál, ahogy azt a fentiekben is leírtuk.

Arról nem volt szó, hogy az átjárók hogyan határozzák meg az útvonalakat. Az esetükben a fenti stratégia nem használható, hiszen az útvonal-választási táblázatuknak megfelelően teljesnek kell lennie. Ezért valamiféle útvonal-választási protokoll jelenléte szükséges, amely azt írja le, hogy az átjárók hogyan találhatják

meg egymást, és hogyan frissíthetik az adatbázisukat a különböző hálózatokhoz vezető (legjobb) útvonalakról.

1.11.5.5. Az útkeresési tábla használata

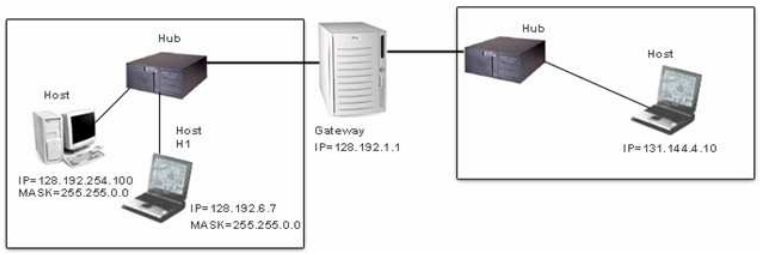
1. Vegyük ki a célgép IP címét a csoamgból, nevezzük ezt *ipdest*-nek

2. Induljunk az útvonal kiválasztási tábla elejéről (majd folytassuk a többi bejegyzéssel). Számítsuk ki a hálózati címrészt az *ipdest*-ből:
ipnet = AND (netmask, ipdest)

– Ha az *ipnet* azonos a hálózati címmel (network address), küldjük a csoamgot a következő router-re (next-hop).

– Ha nem azonosak, akkor ismételjük meg az első két lépést.

– Ha a táblázat egyetlen bejegyzésével sem egyezik meg, akkor adjunk hibaüzenetet



1.58. ábra. Hálózat hoszt útkereséséhez

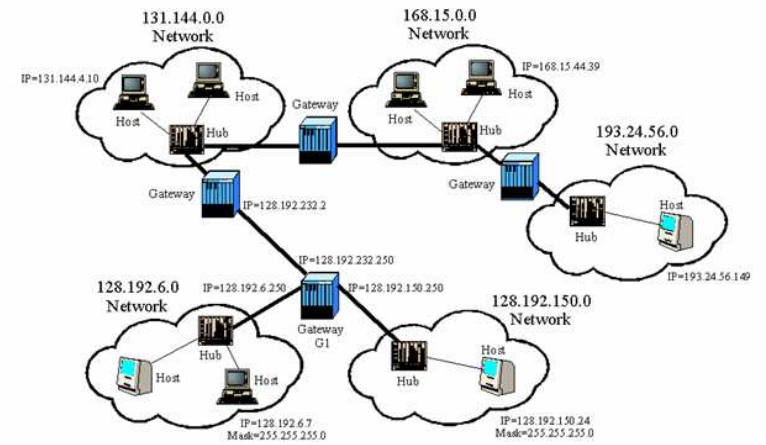
Útkeresési tábla a H1-es hoszt-hoz:

entry	netmask	net-address	next-hop	hop-count	comments
1	255.255.0.0	128.192.0.0	128.192.6.7	0	DCN
2	0.0.0.0	0.0.0.0	128.192.1.1	1	DR

1.Példa: A csomag ipdest=128.192.254.100
A hálózat címrész kiszámítása az ipdest-ből és a táblázat első sorában szereplő netmask-ból.

ipdest =	10000000 11000000 11111110 01100100	(128.192.254.100)
netmask =	11111111 11111111 00000000 00000000	(255.255.0.0)
AND	_____	
ipnet =	10000000 11000000 00000000 00000000	(128.192.0.0)

Az ipnet és a net-address összehasonlítása 128.192.0.0 és 128.192.0.0 azonosak Mivel azonosak, tovább küldjük a következő router-re: next-hop=128.192.6.7 Hálózat átjáró útkeresése



1.59. ábra. Hálózati átjáró útkeresése

A G1-es átjáró útkeresési táblája:

entry	netmask	net-address	next-hop	hop-count	comments
1	255.255.255.0	128.192.6.0	128.192.6.250	0	DCN
2	255.255.255.0	128.192.7.0	128.192.7.250	0	DCN
3	255.255.255.0	128.192.150.0	128.192.150.250	0	DCN
4	255.255.255.0	128.192.232.0	128.192.232.250	0	DCN
5	255.255.255.255	131.144.4.10	128.192.232.2	1	HSR
6	255.255.0.0	168.15.0.0	128.192.232.2	2	NSR
7	0.0.0.0	0.0.0.0	128.192.232.2	1	DR

2. példa: A csomag ipdest=128.192.150.24

Olyan bejegyzést keresünk a táblában, amely netmask-jából és az ipdest-ből generált címrész azonos a sor net-address bejegyzésével, majd elküldjük a csoamgot a megfelelő címre (next-hop).

bejegyzés	netmask	AND(128.192.150.24)	net-address	eredmény
1	255.255.255.0	128.192.150.	128.192.6.0	nem azonos
2	255.255.255.0	128.192.150.	128.192.7.0	nem azonos
3	255.255.255.0	128.192.150.	128.192.150.0	direkt küldés

3. példa: A csomag ipdest=168.15.44.39 2.
Olyan bejegyzést keresünk a táblában, amely netmask-jából és az ipdest-ből generált címrész azonos a sor net-address bejegyzésével, majd elküldjük a csomagot a megfelelő címre (next-hop).

bejegyzés	netmask	AND(168.15.44.39)	net-address	eredmény
1	255.255.255.0	168.15.44.0	128.192.6.0	nem azonos
2	255.255.255.0	168.15.44.0	128.192.7.0	nem azonos
3	255.255.255.0	168.15.44.0	128.192.150.0	nem azonos
4	255.255.255.0	168.15.44.0	128.192.232.0	nem azonos
5	255.255.255.255	168.15.44.39	131.144.4.10	nem azonos
6	255.255.0.0	168.15.0.0	168.15.0.0	küldés: 128.192.232.2

4. példa: A csomag ipdest=193.24.56.149
Olyan bejegyzést keresünk a táblában, amely netmask-jából és az ipdest-ből generált címrész azonos a sor net-address bejegyzésével, majd elküldjük a csomagot a megfelelő címre (next-hop).

bejegyzés	netmask	AND(168.15.44.39)	net-address	eredmény
1	255.255.255.0	168.15.44.0	128.192.6.0	nem azonos
2	255.255.255.0	168.15.44.0	128.192.7.0	nem azonos
3	255.255.255.0	168.15.44.0	128.192.150.0	nem azonos
4	255.255.255.0	168.15.44.0	128.192.232.0	nem azonos
5	255.255.255.255	168.15.44.39	131.144.4.10	nem azonos
6	255.255.0.0	168.15.0.0	168.15.0.0	küldés: 128.192.232.2

1.11.5.6. Network Address Translation

A fentebb már említett 192-vel vagy 10-zel kezdődő IP címek egy olyan közös címkészlet részei, amit több hoszt tud egyidejűleg használni a NAT segítségével. Ez a funkció már előre be van építve a proxy szolgáltatókba, routerekbe, vagy számítógépek operációs rendszereibe. Azért tudja több hoszt egyidejűleg használni ezeket a címeket, mert ezeket a címeket csak a felhasználó belső hálózatán keresztül lehet elérni, kívülről nem láthatóak, csak az az egy valós IP cím, amelyen keresztül a belső hálózat az Internetre csatlakozik.

Például vállalatunk egyetlen Internet-csatlakozással rendelkezik, legyen ennek a címe 152.66.213.80. Viszont 150 számítógéppel rendelkezünk, melyek mindegyikén

Internet elérést akarunk lehetővé tenni. Ekkor 2 hálózati interfészt és egy NAT-ot lehetővé tevő szoftvert kell telepítenünk a direkt-Internet kapcsolattal rendelkező eszközünkre (pl. egy UNIX/WINDOWS alapú PC).

Az egyik hálózati kártya megkapja az Internet eléréshez szükséges hálózati paramétereket, viszont a másik a 192.168.0.1-es címet kapja, amire a többi, belső-hálózati számítógépet csatlakoztatjuk.

A belső-hálózati eszközök mind 192.168.0.2-151 címet kapják, és átjárójuk pedig a 192.168.0.1 cím lesz. A belső-hálózatban lévő számítógépeket nem lehet látni és elérni a külső hálózartól, ugyanis fizikai kapcsolat csak a NAT szerver és az Internet között létezik.

Tehát amikor a belső hálózatról el akarunk majd érni pl. egy www.sch.bme.hu címet, akkor a NAT kiszolgáló a 192.168.0.x címet lefordítja a saját címére (192.168.0.1) amellyel a www.sch.bme.hu szerver fog kommunikálni.

1.11.6. Címfeloldás lokális hálózatokon

1.11.6.1. ARP

Az IP cím nem definiálja fizikailag a hosztokat. Ezeket fizikai címük azonosítja, lokális hálózatokon ez 48 bites cím, amit szokás MAC (Media Access Control) nevezni.

inden hálózati csatoló saját MAC címmel rendelkezik. Ez olyan akár egy 6 byteos rendszám. Az egyes hálózati kártyagyártók saját MAC tartománnyal rendelkeznek és ezeket a címeket égetik csatolóikba.

A lokális hálózat gépei kommunikálásra a fizikai címeket használják. A korábban kifejlesztett IP címzési rendszer változatlan alkalmazását az ARP (Address Resolution Protocol) teszi lehetővé.

A protokoll feladata az IP cím és a hardware cím közti konvertálás. Ennek módja természetesen függ az alkalmazott link-től, így a metódus nem az IP része, hanem minden link-hez külön definiálta az IETF. A legáltalánosabban elterjedt módszer (Ethernet, Token Ring és minden broadcast jellegű link fölötti) a következő.

1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

Hardware tipus		protokoll tipus (0800)
HLEN	PLEN	Művelet
Forrás MAC(1-4byte)		
Forrás MAC(5-6 byte)		Forrás IP (1-2 byte)
Forrás IP (3-4 byte)		Cél MAC (1-2 byte)
Cél MAC (3-6 byte)		
Cél IP (1-4 byte)		

Tegyük fel, hogy a kapcsolatot kereső gép ismeri a partner IP címét. Az előzőleg felderített fizikai címeket átmenetileg egy ARP chaceben tárolja. (Ez a hálózat tehermentesítése céljából fontos.) Először itt vizsgálja meg a keresett gép IP címét, ha már ismert, akkor a datalink headerbe illeszti.

Ha nincs meg a fizikai cím, akkor az ARP protokoll generál egy ARP request csomagot, melyben a címzett fizikai címe helyett a FF-FF-FF-FF-FF-FF cím van. Ez fizikai broadcast cím és minden állomás veszi, de csak az válaszol, amelyiknek az IP címe a keretben szerepel.

Az az állomás, amelyik a megadott IP címről magára ismer, beleírja a csomagba saját MAC címét és visszaküldi a csomagot (ARP response). Előtte azonban megjegyzi a küldő IP és MAC címét, hiszen valószínűleg csomagot fog kapni attól az állomástól, arra valószínűleg válaszolni is fog. Ez a folyamat a felhasználó elöl rejtve zajlik.

Ez a módszer olyan link-ek fölött, melyek nem broadcast jellegűek, azaz nem lehet egy kerettel minden állomást megszólítani (például X.25) nem működik, ott ennél sokkal bonyolultabb eljárások kellenek.

5. Példa: Tegyük fel, hogy a 128.6.4.194 rendszerről a 128.6.4.7 rendszerrel szeretnénk kapcsolatba lépni.

A kezdeményező rendszer első lépésként azt találja, hogy a 128.6.4.7 is ugyanazon az Ethernet alapú hálózaton található.

Második lépésként a 128.6.4.194 megnézi, hogy szerepel-e a saját ARP táblázatában a 128.6.4.7 címen bejegyzés (a 128.6.4.7 Ethernet címe).

Ha igen, akkor a csomaghoz egy Ethernet fejlécet csatol, és elküldi. Tegyük fel azonban, hogy nincs ilyen bejegyzés az ARP táblázatban. Így a csomagot nem lehet elküldeni, hiszen nincs meg az Ethernet cím.

Itt jön be az ARP. A 128.6.4.169 rendszer egy

"Kérem a 128.6.4.7 Ethernet címét"

tartalmú ARP kérést ad ki az Ethernet hálózatra. (Az adott hálózaton minden rendszer figyeli az ARP kéréseket.) 128.6.4.7 hallja a kérést, és egy ARP üzenetet küld a 128.6.4.169-nek, amelynek tartalma:

"A 128.6.4.7 Ethernet címe 8:0:20:1:56:34".

A kérést adó rendszer a kapott információt bejegyzi az ARP táblázatába.

1.11.6.2. RARP

Rendszerint a hosztok a saját IP-címüket a háttértárolón őrzik. De mi történjen a lemez nélküli gépekkel, ha TCP/IP protokollt akarják használni? (pl.: X-terminálok saját állomány kiszolgálójukról töltődnek be.) A megoldást a RARP (Reverse Address Resolution Protocol) adja, ami nagyon hasonló az ARP-hez és a formátuma is megegyezik vele. A RARP protokoll egy szerverhez fordul saját IP címének kiderítése céljából, amiről később operációs rendszert is letölthet.

1.11.6.3. Internet Control Message Protocol (ICMP)

Az ICMP tulajdonképpen az IP felügyeleti protokollja, úgy viselkedik, mintha magasabb szintű protokoll lenne, de az IP integráns része.

Egy ICMP csomag valójában egy IP csomag, melyben a protokoll azonosítója 1.

ICMP üzenetet a következő szituációkban küldenek:

- *A címzett elérhetetlen.* A router-ek küldik a feladónak, ha a cél nem létezik, vagy a hálózata végtelen távolságban van, esetleg beállított DF bit mellett fragmentációra lett volna szükség. A címzett is küldheti, ha például nem fut a jelölt protokollt támogató processz.
- *Lejárt a csomag élettartama.* A router-ek küldhetik, ha a TTL nullára csökkent, vagy a címzett, ha a fragmentek összevárására kijelölt idő letelt és még nem érkezett meg az összes darab. " Hibás IP csomagot adtunk fel.

- *Túl gyorsan küldjük a csomagokat.* Ezt az üzenetet router-ek vagy a címzett küldheti, tipikusan még mielőtt kimeríti erőforrásait, így az a csomag, amire válaszként jön, még célba érhetett.
- *Átirányítás (Redirect).* Más irányba küldjük inkább az ehhez a címzetthez küldendő csomagjainkat, mert arra rövidebb az út. Ezt router-ek küldhetik az állomásoknak a hálózat működésének javítása érdekében.
- *Echo és Echo reply.* Ezzel a két üzenettel a címzett elérhetőségét és működését tesztelhetjük. Egy Echo üzenetre minden állomásnak kötelező Echo reply-jal válaszolni.
- *Időbélyeg kérés és válasz.* Ez az állomások óráinak szinkronizálására használatos.
- *Saját hálózat számának lekérdezése és megválaszolása.* Arra szolgálnak, hogy egy állomás megszólítson valakit a saját hálózatán (a hálózat száma kitöltetlenül hagyható) és attól elkérje a hálózat számát. A válaszoló egy teljesen kitöltött címmel válaszol, így a lekérdező állomás is birtokába jut a hálózat számának.

Az eredeti ICMP funkciók mellé az RFC1256 megjelenésével az ICMP router discovery mechanizmusa társult. A router-ek periodikusan hirdetményeket tesznek közzé a link-en (Router Advertisement), melyekben számos paraméterüket közlik az állomásokkal (többek között MAC címüket). Az állomások így megismerik, milyen router-ek is vannak a link-en és könnyen továbbíthatják nekik csomagjaikat, melyek nem a link-re szólnak. Az állomásoknak nem kell megvárniuk a hirdetmények periodikus közzétételét, hanem felszólításokkal (Router Solicitation) soron kívül is kiválthatják azokat.

1.11.6.4. Információ- és névszervezés (Domain rendszer)

Ahogyan már korábban jeleztük, a hálózati szoftvernek egy 32 bites Internet címre van szüksége ahhoz, hogy egy kapcsolatot felépíthessen, vagy hogy csomagokat küldhessen. A felhasználók viszont inkább a számítógépek neveivel mintsem számokkal szeretnének hivatkozni rájuk (a neveket könnyebben meg lehet jegyezni). Ezért létezik egy adatbázis, amelyből a hálózati szoftver kikeresheti a névnek megfelelő címet, és fordítva.

Amikor az Internet még nem volt ilyen kiterjedt, akkor ez viszonylag könnyen megoldódott, minden gépnek volt egy adatállománya, amelyben az összes többi rendszer nevét és címét felsorolták. Ma már túl sok rendszer létezik ahhoz, hogy az ilyen megközelítés praktikus legyen. Emiatt ezeket az állományokat olyan névkiszolgálók váltották fel, amelyek a gépek neveit és a megfelelő címeket tartják nyilván.

A valóságban egyetlen központi gép helyett az ilyen kiszolgálók egymással összekapcsolt halmaza használatos. Manapság már olyan sok különböző intézmény kapcsolódik az Internethez, hogy nem lenne praktikus, ha egy központi hatóságot kellene értesíteniük minden olyan esetben, amikor egy gépet a hálózatba be- vagy abból kikapcsolnak. Éppen ezért a névadásra az egyes intézmények a rendszerükön belül saját maguk jogosultak.

Az így kialakított névkiszolgálók közösen egy fa struktúrát alkotnak, amely az intézmények hálózati szerkezetének felel meg. Ezt a szerkezetet a nevük is tükrözik.

A tartományrendszer feladata nem merül ki az Internet címek megtalálásában. Minden egyes tartománynév csomópontként szerepel egy adatbázisban. A csomópontnak különböző tulajdonságokat jellemző rekordjai lehetnek. Ilyen az Internet cím, a számítógép típusa, és a számítógép által biztosított szolgáltatások felsorolása.

Egy program egy adott névvel kapcsolatban kérheti ezen információk valamelyikét, vagy az összeset. Megoldható az is, hogy egy számítógéphez az alapértelmezett nevén kívül több nevet is hozzárendeljünk. Az is lehetséges, hogy a tartományrendszerben felhasználókról, levelezési listákról, vagy más objektumokról tároljunk adatokat.

A fenti adatbázisok működését, illetve az azok lekérdezését megvalósító protokollokat is Internet szabvány írja le. Minden hálózati számítógépnek, alkalmazásnak meg kell tudnia valósítani ezeket a lekérdezéseket, mivel hivatalosan így történik a hosztnevek kiértékelése. Ezt az operációs rendszer végzi el az egyes alkalmazások számára.

A hálózati számítógépek a saját konfigurációjuk alapján keresnek egy névkiszolgálót. Ez a kiszolgáló aztán a felsőbb szinten (az ő tartományán) lévő kiszolgálókkal veszi fel a kapcsolatot. A tartományrendszer fontos szerepet tölt be az elektroni-

kus levelezésben. Az adatbázisokban szerepelhetnek olyan bejegyzések, amelyek megmondják, hogy melyik gép kezeli egy adott név leveleit, egy felhasználó levelei hová érkezzenek, illetve levelezési listákat is definiálhatnak.

A névkiszolgálók közösen egy fa struktúrát alkotnak, amely az intézmények hálózati szerkezetének felel meg. Ezt a szerkezetet a nevek is tükrözik. Tipikus példa erre a BORAX.LCS.MIT.EDU név, amely a MIT számítástechnikai laboratóriumának (LCS) egy számítógépét jelöli (ilyen példa lehetne még: maxi.inf.elte.hu, ami az ELTE Általános Számítástudományi tanszékének maxi nevű gépét adja).

A gép Internet címének meghatározásához 4 potenciális kiszolgálót kellene megkérdezni. Először egy központi kiszolgálótól (root - gyökér, ld. a fa struktúrát) kellene megtudakolni, hogy hol található az EDU kiszolgáló, amely nem más, mint a hálózatba kapcsolt oktatási intézmények nyilvántartása. A gyökérként szereplő kiszolgáló több EDU kiszolgáló nevét és Internet címét adná meg. (Minden szinten több ilyen névkiszolgáló van, hogy az esetleges meghibásodások ne okozzanak fennakadást.).

A következő feladat lenne az EDU kiszolgáló lekérdezése a MIT névkiszolgálójáról. Itt is több kiszolgáló nevét és Internet címét kapnánk meg. Ezek közül általában nem mindegyik található az intézmény területén (egy esetleges áramszünet fellépte miatt).

Ez után a MIT-től kérdeznénk le a számítástechnikai laboratórium (LCS) névkiszolgálójának adatait, majd végül a laboratóriumi névkiszolgálók egyike adná a BORAX adatait. A végső eredmény a BORAX.LCS.MIT.EDU gép Internet címe lenne.

A fenti szintek mindegyike egy tartományt (domain) jelöl. A teljes BORAX.LCS.MIT.EDU név pedig egy tartománynév (domain name). (Ugyanígy a felsőbb tartományok nevei is tartománynevek: LCS.MIT.EDU, MIT.EDU és EDU.) Az esetek nagy többségében szerencsére nem kell a fenti lépések mindegyikét végrehajtani. A legfelső kiszolgáló (gyökér) ugyanis egyben a legfelső szinten lévő tartományok (pl. EDU) névkiszolgálójaként is szerepel. Tehát a gyökér kiszolgáló felé irányuló egyetlen kérdéssel a MIT névkiszolgálójához lehet eljutni.

A lokális névkiszolgálók a keresett nevekre kapott válaszokat egy dinamikus frissítésű táblában tárolják. Ez azt jelenti, hogy a LCS.MIT.EDU kiszolgáló le-

kérdezése után tudja, hogy hol keresse a LCS.MIT.EDU, a MIT.EDU és az EDU tartománybeli kiszolgálókat. A BORAX.LCS.MIT.EDU fordítására szintén emlékszik. Persze minden ilyen információnak van egy megfelelő élettartama, ami tipikusan pár napnak felel meg. Az élettartam lejárta után az információk elvesznek.

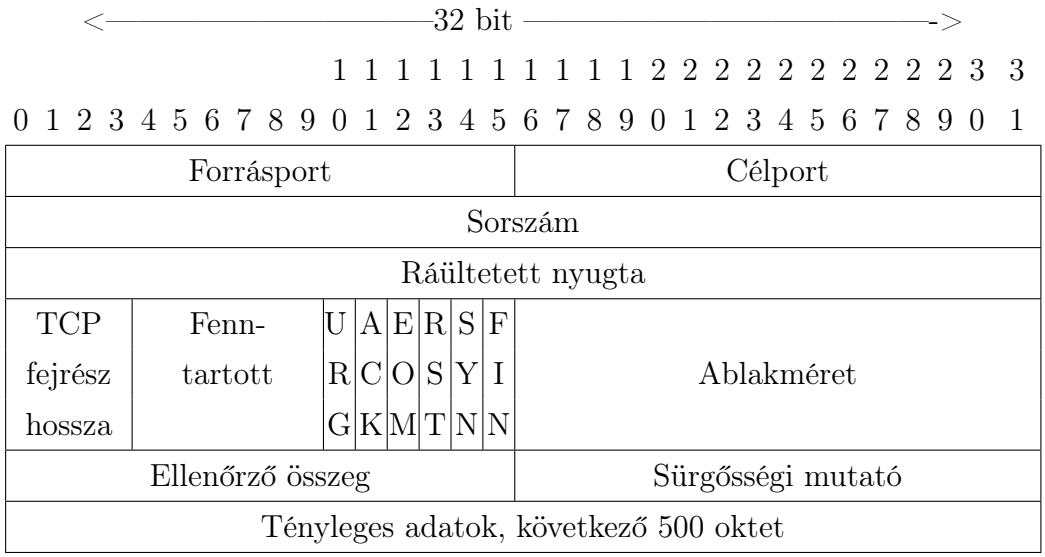
1.11.7. Szállítási protokollok

A TCP/IP család két legfontosabb transzport protokollja a TCP és az UDP. Ezek, bár a hálózati IP fölött működnek, mégsem egyértelműen feleltethetők meg az OSI modell 4. rétegének. Inkább úgy fogalmazhatunk, hogy a TCP-IP és UDP-IP együtt alkotja a 4. szintet vagy a 3-4. szintet együtt.

1.11.7.1. TCP (Transport Control Protocol)

A csomagok a hálózaton több útvonalon érkezhetnek, így megelőzheti a később küldött a korábbit. Mindkét állomás sorszámozza az adott és vett csomagokat, ezért a helyes sorrend biztosítható. De ennek feltétele egy kapcsolatorientált, byte-folyam (stream flow) jellegű, megbízható protokoll. A kommunikáció megkezdése előtt ki kell építeni a kapcsolatot, majd megkezdhetjük az adatátvitelt. A TCP protokoll hiba (elveszett vagy hibás csomag) esetén maga kér újraadást, elfedve ezzel az IP szint megbízhatatlanságát. A TCP protokoll a megbízhatóságot az ún. PAR (Positive Acknowledgement with Retransmission) technikával biztosítja. Ez azt jelenti, hogy a célállomás TCP-t megvalósító szoftvere nyugtázza a csomag kézbesítését, miután a hálózati szinttől (az IP-től) megkapta. A TCP processz egy több mint 20 állapotos állapotgép, meglehetősen bonyolult, hiszen az IP szint számos hibát képes produkálni. Az adat, amit átküldhetünk egy byte-folyam, amit a TCP csomagokra vág és elküld. A kapcsolat full-duplex és rendelkezik egy sebességszinkronizációs mechanizmussal, ami megakadályozza, hogy az adó elárassza a vevőt. A TCP ezen felül figyeli a kapcsolatot és megpróbálja megtippelni az effektív sávszélességet (torlódásokból, válaszidőből, ICMP üzenetekből, stb.), amit szintén felhasznál a kimenő adatsebesség beállításakor. A TCP-vel ezen kívül lehetőség van sürgős adatok továbbítására is. A protokoll előírja, hogy ha sürgős adatot küldtünk, akkor az adat fogadóját a cél-hoszton erről értesíteni

kell, és meg kell adni a lehetőséget a sürgős adat soron kívüli feldolgozására is. Az értesítés módja (mivel operációsrendszer függő) nincs a protokoll által specifikálva



0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1								
Forrásport								Célport
Sorszám								
Ráültetett nyugta								
TCP fejrész hossza	Fenn-tartott	U R G	A C K	E O M	R S T	S Y N	F I N	Ablakméret
Ellenőrző összeg								Sürgősségi mutató
Tényleges adatok, következő 500 oktet								

A TCP forgalomirányítási elve az ablakméret mezőn alapul. Az ablakméreten az elküldetlen adatok egy részét, és a nem nyugtázott adatokat látja. Az adatok továbbítása során az ablakméret csúszik előre, az el nem küldött adatok irányába. A címzett állomás szabad vételi pufferterületének megfelelően a TCP csomagok fejlécében írja az ablakméret mezőt. Az ablakméret mező határozza meg, hogy a nyugtázott bájttal kezdődően hány bájtot lehet elküldeni és így informálja a feladót, hogy mennyi adatot képes venni.

Ha a címzett nem képes adatot venni 0-t ír a mezőbe. Ekkor mindaddig szünetel az adatátvitel, míg a címzett nullától különböző számot nem ír a mezőbe.

Végül essék szó az ellenőrzőösszegről. Ez egy olyan szám, amelyet a csomagban lévő oktetek összeadásával kapunk (Az OSI szállítási rétege ezt úgy képzi, hogy az adatokat 16 bites számokként összeadja, majd veszi ennek egyes komplemensét.) Az eredmény aztán bekerül a TCP fejlécbe. A vevő oldali TCP is kiszámítja a fenti algoritmus szerinti ellenőrzőösszeget. Ha a kettő nem egyezik, akkor a csomaggal az átvitel közben valahol valami baj történt és azt a protokoll eldobja.

A TCP fejrész bitjei a következők:

< 32 bit >	
Forrásport	Célport
Hossz	Ellenőrző összeg
Tényleges adatok ...	

1.9. táblázat. Az UDP fejléce

- URG: Ennek a bitnek 1 az értéke, ha a sürgős adatok offszetje fejrészmező ki van töltve (és érvényes ...). Ez az offszet azt mutatja majd meg, hogy az aktuálisan átvitt byte után hányadik byte után következik a sürgős adat.
- SYN: Összeköttetés-felépítéskor fontos
- ACK: "Nyugta" mező érvényes adattal van-e kitöltve
- FIN: A küldőnek nincs több elküldendő adata (ezzel zárul le az egyik irányban a kapcsolat).
- RST: Inkonzisztens állapotba került összeköttetés megszüntetésére vonatkozó kérelem.
- EOM: End Of Message flag (nem kell a TCP kapcsolat használójától adatokra várni (amíg a 64 byte-os szokásos TCP üzenethossz összejönne), mert rövid időn belül lehet, hogy nem lesz új elküldendő adat).

1.11.7.2. User Datagramm Protocol (UDP)

Az UDP (User Datagramm Protocol) egy összeköttetés-mentes protokoll. Az UDP információját egy IP csomagba helyezi, ellenőrző összeget számol hozzá és feladja.

A kézbesítést nem garantálja, de a hibás kézbesítést észlelhetővé teszi. Olyan kérdés-válasz jellegű szolgáltatásokhoz használatos, ahol ha a kérdés vagy a válasz elvész, a hiba egyszerű újrakérdezéssel megoldható. Az, hogy egy csomag elvész, ritka esemény és ilyen kérdés-válasz esetén könnyen felderíthető.

Az UDP egyszerűsége miatt sokkal kíméletesebben bánik a hálózati erőforrásokkal, mint a TCP. (Egy TCP kapcsolat kiépülése minimum 3 IP csomagba kerül, mielőtt még akár 1 byte-os felhasználói adatot átvittünk volna.) A TCP-hez hasonlóan az UDP is rendelkezik portokkal, melyek számkiosztása független a TCP portokétól.

Az UDP nem végez annyi feladatot, mint a TCP, nem tördeli szét az üzenetet datagrammokra, nem figyeli a már elküldött adatokat, hogy majd esetleg újraadja őket. Az UDP csak portszámokat biztosít, hogy egyszerre több program is használhassa a protokollt.

Az UDP portszámok ugyanúgy használatosak, mint a TCP portszámok. Az UDP-t használó kiszolgálókhoz is léteznek jól ismert portszámok. Megjegyezzük még, hogy az UDP fejléc sokkal rövidebb, mint a TCP fejléce. Ebben is szerepel a forrás- és a célport száma, valamint egy ellenőrző összeg, de ennyi az egész. Nincs benne sorszám, mert nincs szükség rá.

1.11.7.3. Portszám kiosztások

A portszám azonosítja a célállomáson belül megszólítandó kommunikációs partnert.

A 0 és 1023 közötti portszámok foglaltak, ezeken találhatóak az ismert szolgáltatások (well-known-services), e fölött szabadon használhatóak a port-számok. (1024-65536-ig).

Például a HTTP processz hallgatja a 80-as TCP portot, a bejövő kérelmet feldolgozza és válaszol (például küldi a WWW oldalt). Aki tehát egy állomás HTTP proccesszével kíván kommunikálni, az a 80-as TCP portot szólítsa meg.

Minden TCP által előállított IP csomagban a TCP protokollazonosítója található, így az állomás az IP csomag vételekor az információt el tudja juttatni a TCP feldolgozóba. Ott a TCP fejlécből kiolvasva, a megfelelő portot hallgató processzhez juttathatjuk el az információt.

(MEGJEGYZÉS) Néhány példa foglalt portszámokra (lásd még rfc 1060)

Port szám	Program	Megjegyzés
20	FTP-DATA	File Transfer [Default Data]
21	FTP	File Transfer [Control]
23	TELNET	
25	SMTP	Simple Mail
79	FINGER	
110	POP3	Post Office Protocol

Egy gépen több alkalmazás is futhat egyidejűleg. A protokolloknak tehát egyidejűleg több applikációt is ki kell szolgálni. Ahogy megérkezik az IP csomag,

az abban lévő protokoll azonosító alapján szétválogatja a hoszt a TCP-nek és az UDP-nek szóló csomagokat, majd azok a megfelelő applikációhoz továbbítják.

1.11.7.4. TCP Logikai kapcsolatok és ICMP

Fontos hogy megértsük, hogyan épül fel és működik egy TCP kapcsolat a gyakorlatban. A TCP kapcsolat három fő részből áll: kapcsolat létrehozás, adatforgalom-adatcsere, kapcsolat megszakítás.

Az alábbi példa egy POP3 szerver- (110 TCP porton) kliens (1967-es TCP portról csatlakozik) kapcsolatot mutat be.

```
KLIENS                                  SZERVER
_____________________________________________
syn, SEQ=800                                    1
------------------------------------------>     1
src_port=1967, dst_port=110                     1
                                                1
                  syn, ack, SEQ=1567, ACK=801   1 KAPCSOLAT
<-------------------------------------------    1
                  src_port=110, dst_port=1967   1 LÉTREHOZÁS
                                                1
ack, SEQ=801, ACK=1568                          1
------------------------------------------>     1
src_port=1967, dst_port=110                     1
```

```
                  ack, SEQ=1568, ACK=801        2
<-------------------------------------------    2
                  src_port=110, dst_port=1967   2
            DataLen=18 (POP3 Szerver V1.12\n)   2
                                                2
ack, SEQ=801, ACK=1586                          2
------------------------------------------>     2
src_port=1967, dst_port=110                     2
DataLen=5 (quit\n)                              2 ADAT-
                                                2 CSERE
                  ack, SEQ=1586, ACK=806        2
<-------------------------------------------    2
                  src_port=110, dst_port=1967   2
                  DataLen=9 (Sayonara\n)        2
                                                2
ack, SEQ=806, ACK=1595                          2
------------------------------------------>     2
src_port=1967, dst_port=110                     2
```

```
fin, ack, SEQ=806, ACK=1595                          3
------------------------------------------------->  3
src_port=1967, dst_port=110                          3
                                                     3
                    ack, SEQ=1595, ACK=807           3
<-----------------------------------                 3
                    src_port=110, dst_port=1967      3 KAPCSOLAT
                                                     3
                    fin, ack, SEQ=1595, ACK=807      3 MEGSZAKÍTÁS
<-------------------------------------------------   3
                    src_port=110, dst_port=1967      3
                                                     3
ack, SEQ=807, ACK=1596                               3
------------------------------------------------->  3
src_port=1967, dst_port=110                          3
```

A kapcsolat-létrehozás fázis magába foglal egy három irányú handshake-t ("kézfogás", internetes terminológiában ez egy kölcsönös, nyugtázott kapcsolatfelvételt jelent) mialatt a kliens és a szerver átadja majd nyugtázza egymás az ISN-ét (initial sequence number - kezdeti szekvencia/sor érték).

Példánkban (első szakasz) a kliens indítja a folyamatot, elküld a szervernek egy TCP szegmenst előre beállított syn-bit-tel és 800-as SEQ-el (sequence number, lásd előbb). A syn-bit megadja a fogadónak (jelen esetben a szerver POP3 proccesszének), hogy a küldő éppen ISN inicializációs fázisban van, és hogy az ISN még nem került nyugtázásra. A szegmens ACK számát nem mutatjuk, mert ebben a fázisban még érvénytelen.

A következő szakaszban a szerver válasz szegmense tartalmazza a syn- és ack-bit-eket, ahol a SEQ=1567 és az ACK=801. A syn/bit és az 1567-es ISN ugyanazt jelenti mint az előző szakaszban. Az ack-bit mutatja, hogy az ACK mező érvényes, és a 801-es ACK értékkel (SEQ+1) nyugtázza a szerver a kliens ISN-ét.

Az utolsó része a handshake-nek, amikor a kliens elküld egy szegmenst a szervernek, csak ack-bit beállítással. Az ACK mező (1568) egyel nagyobb mint a szerver ISN-e. Ezt a három-irányú handshaket szokták "syn, syn/ack, ack" szegmensek cseréjének is nevezni.

Sok szempontból fontos, hogy ez a handshake segít meghatározni később egy kapcsolat eredetét. A tűzfalaknál, proxy szervereknél, egyéb belépés érzékelő rendszereknél megad egy irányt, hogy megtudjuk, hogyan épült fel a TCP kapcsolat, mivel itt a szerver és kliens között még ott van egy harmadik eszköz is.

A második rész a kapcsolat létrehozás után az adatcsere. Jelen példánk csak annyira terjed ki amennyi feltétlen szükséges. A POP szerver egy egyszerű üzenetet küld a kliensnek, majd a kliens küld egy "quit" parancsot, mire a szerver kijelentkezik ("\n" a sorvége jel).

Végül az utolsó szakaszban láthatjuk a kapcsolat megszakítását. Habár a TCP Full-Duplex kapcsolat (még akkor is, ha a felhasználó nem engedélyezi), a TCP protokoll logikailag a kapcsolatot két pár egyirányú kapcsolatként kezeli. Ebből adódóan a kapcsolat megszakításhoz négy szegmens, pontosabban két pár szegmens szükséges. Esetünkben a kliens elküld egy szegmenst a szervernek fin- és ack-bit beállítással, és a szerver válaszol egy ack-bittel és egy növelt ACK számmal, majd elküldi a szerver is a fin/ack segmentet a kliensnek.

A fenti példa egy szabályos felállásban tárgyalja egy TCP kapcsolat felépítését, az adatcserét és a kapcsolat lebontását.

De mi történik akkor, ha pl. a hoszt nem figyeli azt a portot amelyen a kapcsolatot létre akarjuk hozni, vagy a hoszt nem is létezik?

A következők történnek nem-normális esetekben:

- *Host not listening on TCP port:* Ha az A hoszt megpróbál a B hosztra csatlakozni egy olyan TCP portra amit B hoszt nem tart fenn, akkor a B hoszt egy RST és ACK beállított szegmenssel válaszol.
- *Host not listening on UDP port:* Ha az A hoszt megpróbál csatlakozni a B hosztra egy olyan UDP porton amit a B hoszt nem tart fenn, akkor a B hoszt küld egy ICMP port unreachable üzenetet az A hosztnak.
- *Host does not exist:* Ha A hoszt megpróbál a B hoszthoz csatlakozni de az nem érhető el, vagy nem létezik, akkor a B hoszt alhálózatának routere küld egy ICMP hoszt unreachable üzenetet az A hosztnak.

1.11.8. Összefoglalás

Az információ datagrammokra bontva terjed. A datagramm az üzenetben elküldött adatok összessége, továbbításukra az egyes hálózati szintek protokolljai szolgálnak.

Tekintsünk egy egyszerű, ámde szemléletes példát abban az esetben, ha a hoszt Ethernet kártyával, TCP és IP protokollon keresztül akar adatot küldeni.

A TCP végzi ez esetben az üzenetek darabolását, míg a másik oldalon az összerakást. Kezeli az esetleges elvesző csomagok újrakérését és a sorrendváltozást.

Az IP az egyedi datagrammok továbbításáért felelős. A példánkon a következő adathalmazt akarjuk a hálózaton átvinni:

xxxxxxxxxxxxxxxxxxxxxxxxxxxx

A TCP ezt datagrammokká darabolja:

xxxx xxxx xxxx xxxx xxxx xxxx xxxx

ezután minden datagramm elejére egy fejlécet rak, ami tartalmazza a forrás- és a célportot, a sorozatszámot és az ellenőrző összeget:

Txxxx Txxxx Txxxx Txxxx Txxxx Txxxx Txxxx

ezt adja tovább az IP-nek, ami elé rakja a saját fejrészét a forrás és a cél IP címével:

ITxxxx ITxxxx ITxxxx ITxxxx ITxxxx ITxxxx ITxxxx

a hálózati elérési szint protokollja (PPP, Ethernet, SLIP) is hozzá teszi a saját keretét, ami Ethernet esetében egy fejrészből (E) és az ellenőrző összegből (C) áll:

EITxxxxC EITxxxxC EITxxxxC EITxxxxC EITxxxxC EITxxxxC EITxxxxC

Az ilyen módon csomagolt datagrammok megérkezése után az egyes fejléceket leszedi a megfelelő protokoll.

Az Ethernet interfész az Ethernet fejrészét és az ellenőrző összeget. Ezután ellenőrzi a protokollra utaló típuskódot. Ha az IP-re mutat, akkor a datagrammot átadja az IP-nek, amely leszedi az IP fejrészt, és a fejrész protokoll mezőjének tartalmát megvizsgálja.

Itt általában azt találja, hogy TCP (vagy UDP), és tovább adja a datagrammot a TCP-nek (vagy az UDP-nek), ami szintén leszedi a saját fejrészét. A TCP a sorszám mező tartalma és egyéb információk alapján állítja össze az eredeti állományt.

1.11.9. Az alkalmazási réteg

A felhasználó számára a legkézenfekvőbb ez a réteg, ez az, amivel kapcsolatban van a számítógépe révén, hiszen az alkalmazási réteg a köznapi értelemben vett Internetes programok rétege.

A szolgáltatások alapvetően két csoportba sorolhatók: az *off-line* szolgáltatások közvetlen hálózati kapcsolatot nem igényelnek, míg az *on-line* szolgáltatások igen.

Mivel az Interneten eltérő felépítésű hálózatokat kötnek össze, szükséges a kommunikáció közös szabványainak kidolgozása, amelyeket az RFC (Request for Comments, megvitatásra készített anyag) dokumentumok tartalmazzák. Ha megszületik egy szabványtervezet, akkor azt először ajánlásként teszik közzé, és kap egy RFC számot. Ha végül az ajánlást elfogadják, akkor Hivatalos Internet Protokoll (Official Internet Protocols) lesz belőle, de továbbra is az RFC számmal hivatkoznak rá. Megállapodás szerint minden RFC új számot kap, ha átdolgozzák.

Az alkalmazási réteg lényegesebb protokolljai a következők:

SMTP: (Simple Mail Transfer Protocol, egyszerű levéltovábbítási protokoll): a hálózati felhasználók egymással való kommunikációját teszi lehetővé, leveleket tud küldeni és fogadni. Sokszor alkalmazzák a POP3 protokollal együtt, ami levelek letöltésére használható.

TELNET: Terminál emuláció segítségével a saját gépét terminálnak használva egy távoli hosztra felhasználóként lehet bejelentkezni.

FTP: (File Transfer Protocol, fájl átviteli protokoll): A felhasználónak lehetővé teszi az általános könyvtár és fájlműveletek végrehajtását a saját gépe és egy távoli hoszt lemezegysége között.

GOPHER: Hierarchikusan felépített információban kereső protokoll, a WWW őseként tartják számon.

HTTP: (HyperText Transport Protocol, hiperszöveg átviteli protokoll): Ezen protokoll segítségével lehet a weblapokat továbbítani. NNTS (Network News Transport Protocol, hálózati hírszolgáltatás átviteli protokoll).

A **NEWS:** hírszolgáltatást megvalósító protokoll, segítségével csatlakozhatunk a hírcsoportokhoz.

1.11.9.1. Az Internet szolgáltatásainak rövid ismertetése

Az Internet által jelenleg biztosított lényegesebb szolgáltatások ABC sorrendben:

- *Archie* - szoftverkereső szolgáltatás

- *Ejournals* - hálózaton keresztül terjesztett újságok
- *E-mail* - elektronikus levelezés
- *Finger* - hálózati felhasználók adatszolgáltatása
- *FTP* - távoli gépek közötti állománycsere
- *Gopher* - menürendszerű adatforrás-tallózó
- *IRC* - többcsatornás, többirányú párbeszédes kapcsolat
- *Jughead* - korlátozott képességű Gopher menütallózó
- *Listszerver* - levelezési csoportok kiszolgálása
- *Usenet* - hírcsoportok kiszolgálása
- *Talk* - kétirányú párbeszédes kapcsolat
- *Telnet* - távoli gépeken történő munka
- *Veronica* - világméretű Gopher menütallózó
- *WAIS* - adatbázis alapú információszolgáltató
- *WWW* - hiperszöveg és multimédia alapú gopher
- *White Pages* - hálózati-felhasználó kereső szolgáltatás
- *Whois* - hálózati-felhasználó kereső szolgáltatás
- *Zines* - hálózaton keresztül terjesztett magazinok.

Ebben a fejezetben ezek közül a legfontosabb, a hétköznapjainkban is használt szolgáltatások bemutatása következik.

1.11.9.2. Böngészés az Interneten (GOPHER, WWW)

Az Interneten az információk összegyűjtése, rendszerezése, megkeresése nem könnyű feladat. Az információk közötti kapcsolat megszervezésére az alábbi lehetőségek adódhatnak:

Hierarchikus szervezés: az információk közötti összefüggéseket egy szintekből álló rendszerbe szervezzük. Elindulunk a legmagasabb szintről, és minden szinten kijelöljük, amihez tartozó alatta lévő szinten folytatjuk a keresést. Ha belegondolunk, kicsiben a háttértárakon ilyen szisztémával tároljuk a fájljainkat, így ez nem új elv. A hierarchikussága viszont utal arra, hogy eléggé rugalmatlan. A felsőbb szinteken szinte nincs lehetőség módosításra, átstrukturálásra, hiszen megbomlik az eddigi rendszer.

A másik egy sokkal rugalmasabb rendszer, ami az egymásra hivatkozásokon alapul. Erre jó példák a lexikonok. Az ilyen módon kialakított dokumentumokat hívjuk hiperszövegnek (hypertext).

1.11.9.3. A Gopher

A hálózaton való hierarchikus keresésre jó példa a Gopher. Tipikus példa egy könyvtárban való keresés, pl. egy adott könyv címe alapján. A gopher szerverek egy könnyen kezelhető menürendszert adnak a kezünkbe. Sokan a www ősének is tartják, ez volt az első lehetőség szabadon keresni az adatbázisokban az Internetre föltett dokumentumok között.

1.11.9.4. A World Wide Web

A WWW (World Wide Web) általános ügyfél-kiszolgáló hálózati koncepcióra épül. A Gopherhez képest a legnagyobb különbsége és egyben újítása az, hogy grafikus és multimédiás elemek is tartalmaz (míg a Gopher kizárólag szövegest), és a második, hipertextes szervezést támogatja. A felhasználó gépén futnia kell egy böngésző-programnak, és akkor a www szerverek kiszolgálóprogramjai által küldött lapokat meg tudják jeleníteni.

Minden információs egység - kép, hang, animáció, szöveg - forrásként jelenik meg a hálózaton. Ezekre a forrásokra olyan módon lehet hivatkozni, hogy meg kell adni a forrás helyét és annak módját, hogy a használt program hogyan tudja megjeleníteni, használni azt. Az alkalmazott megjelenítési módot az URL (Uniform Resource Locator, egységes forrásazonosító) adja meg. Pl.:

http://webmail.hszk.bme.hu:8080/program/belep.html

Az URL a következő információkat tartalmazza (utána zárójelben a példa azon eleme):

- A protokollt, amelyet az adott forrás eléréséhez használunk (http, ftp, gopher, stb.) (http)
- Annak a kiszolgálónak az IP címét vagy teljes domén-nevét, amelyen az adott forrás található. Ez az információ két perjellel kezdődik (//). (//webmail.hszk.bme.hu
- A kiszolgáló portjának a számát, amelyen a kívánt adatot el lehet érni. Ha ez nem szerepel, akkor az általánosan használt alapértelmezett fogja a böngésző használni. Egy kettőspont (:) vezeti be. (:8080)
- A forrás helyét a kiszolgáló lemezegységének állomány-rendszerében. Ez közvetlenül a nevét lezáró perjel (/) után következik. (/program/belep.html)

Ha csak a könyvtár nevét adjuk meg, akkor az abban található index.html oldalt nyitja meg. Amennyiben ez nem létezik, akkor a könyvtár tartalmát tekinthetjük meg.

A dokumentumok logikai struktúráját a HTML (HyperText Markup Language, hiperszöveg leíró nyelv) jelölései segítségével lehet szabályozni. A HTML arra készült, hogy segítségével a dokumentumok szokásos, sorban egymás utáni olvasása helyett a szövegben elhelyezett kapcsolatok alapján az egész dokumentum könnyebben legyen áttekinthető és elolvasható. Segítségével logikusan szervezett és felépített dokumentumokat lehet készíteni, hiszen a nyelv alkalmas logikai kapcsolatok létrehozására a dokumentumon belül és dokumentumok között, amit a dokumentum olvasója kezelhet.

A WWW maga úgy is tekinthető, mint egy dinamikus információ tömeg, amelyben a hiperszöveg segítségével kapcsolatok (linkek) vannak. A HTTP ügyfél-kiszolgáló protokollt hiperszöveg dokumentumok gyors és hatékony megjelenítésére tervezték. A protokoll állapotmentes, vagyis az ügyfélprogram több kérést is küldhet a kiszolgálónak, amely ezeket egymástól teljesen függetlenül kezeli és minden dokumentum elküldése után le is zárja a kapcsolatot.

Ennek megfelelően négy lépése van a HTTP-kapcsolatnak:

1. A kapcsolat megnyitása. Az ügyfél meghívja a kiszolgálót az Interneten keresztül az adott cím és port alapján.
2. A kérés elküldése. Az ügyfélprogram üzenetet küld a kiszolgálónak, amelyben valamilyen kiszolgálást kér. A kérés HTTP-fejlécből és a kiszolgálónak küldött adatokból áll (ha van ilyen)
3. A válasz. A kiszolgáló a választ visszaküldi az ügyfélprogramnak. Ennek része a fejléc, amely leírja a válasz állapotát (sikeres-e), és ezt követik maguk az adatok.
4. A kapcsolat lezárása. A kiszolgáló a válasz elküldése után lezárja a kapcsolatot, így az erőforrások megint felszabadulnak a következő kéréshez.

1.11.9.5. Az E-mail

A legalapvetőbb szolgáltatás, a legelső, amit az Interneten használtak, az az elektronikus levelezés.

Egy levelezőprogram segítségével szöveges állományt küldhetünk az Internet bármelyik felhasználójának. Egy ilyen levél - a hagyományoshoz hasonlóan - a szövegből és a címzette(ke)t a feladót és néhány kiegészítő információt tartalmazó fejlécből áll.

A levelezés megvalósításához a legfontosabb szükséglet, hogy mind a feladó, mind a címzett rendelkezzen e-mail címmel. Az E-mail-en keresztül közvetlenül csak a 0-127 ASCII karakterek küldhetők át. Ha olyan karaktert küldünk, aminek a 8. bitje 1, azt a rendszer levágja, elvész. Így közvetlenül bináris fájlok átvitele nem lehetséges.

Az UUENCODE eljárás a fájlt alkotó bájtsorozatot konvertál 7 bites szöveggé oly módon, hogy a fájl elejéről kezdve sorban vesz 3 db 8 bites bájtot, és azt szétdarabolja 4 db 6 bites darabra. Az UUDECODE eljárás ebből visszaalakítja az eredeti információt.

Egy másik megoldás esetén már lehetőség van levélben nem ASCII karakterek, képek, hangok küldésére. Ez a MIME-eljárásnak (Multi-purpose Internet Mail Extensions, mindenféle internetes levélalkotók) nevezik. Amelyik levelezőprogram ismeri ezt, azzal írható illetve olvasható akár magyar ékezetes betűket tartalmazó levél is. A MIME egy speciális fejléccel látja el a leveleket.

1.11.9.6. Az FTP

Az FTP protokoll a hálózatban lévő gépeken megtalálható fájlok átvitelére szolgál. Használata folyamatos hálózati kapcsolatot igényel.

Az FTP protokoll két átviteli módban működhet: ASCII és bináris. Az előbbi, mivel 7 bites kódokat használ, szövegállományok átvitelére szolgál (de nem a dokumentumokra!), az utóbbi bináris kódok továbbítására alkalmas.

A letöltés megkezdése előtt jól gondoljuk meg, hogy melyikre van szükségünk, mert rossz választás esetén használhatatlan lesz a letöltött fájl. Az ASCII mód csak a txt fájlok átvitelére alkalmas, amire viszont a bináris nem!

A legfőbb különbség, hogy amíg a szövegfájlok sorai EoLn bájttal, maguk a fájlok pedig EoF bájttal vannak lezárva addig a bináris (gépi kódú) fájlok tagolatlanok.

Ha bináris módban txt fájlt töltünk le, két jelentős hibával kell számolnunk. Egyrészt nem lesz sorokra tagolva a fájl, másrészt az összes 128-nál nagyobb ASCII kódú karakter hiányozni fog. ???????????

Ha ASCII módban bináris fájlt viszünk át, egy komoly hibával kell számolnunk: a fájl végére be fog kerülni az EoF jel, így megváltozik a fájl mérete, ami hibás feldolgozást eredményezhet!

A kapcsolat egy FTP programmal lehetséges, ahol a felhasználói név és jelszó megadása után a szokásos fájlkezelő parancsokkal dolgozhatunk. A legtöbb FTP program ma már grafikus felületű, esetleg a számítógépen amúgy is használt fájlkezelő része.

A mindenki számára publikus adatok elérésére az ún. anonymous FTP-vel lehetséges. Ilyenkor felhasználói névnek az anonymous szót, míg jelszónak az e-mail címünket kell megadni.

1.11.9.7. Telnet, SSH

Egy távoli gépre úgy lehet belépni, mintha egy terminálja előtt ülnénk. Azaz a Telnet a gépek közti távoli bejelentkezést lehetővé teve protokoll neve. (Manapság egyre inkább a Telnet alapjaira épülő, sokkal biztonságosabb SSH-t használják, de az elve teljesen ugyanaz, ezért tárgyaljuk együtt a kettőt.) Ez is folyamatos hálózati kapcsolatot igényel, a sebességigénye hasonló az ftp-hez, (persze, csak ha nem szeretnénk, hogy egy billentyű lenyomása után 10 másodperccel jelenjen meg a karakter).

Telnettel csak akkor tudunk egy másik gépre belépni, ha ahhoz van hozzáférési jogosultságunk. Bejelentkezés után a rendszer úgy viselkedik, mintha ott ülnénk a távoli gép előtt, azaz a távoli gép operációs rendszerének konvenciói érvényesek, parancsainkat a telnet protokoll adja át a távoli gép operációs rendszerének, és az a rendszer hajtja végre. Így távoli gépen programokat futtathatunk, megnézhetjük az oda érkezett leveleinket, stb.