SZÉCHENYI ISTVÁN EGYETEM

Lencse Gábor

SZÁMÍTÓGÉP HÁLÓZATOK

UNIVERSITAS-GYŐR NONPROFIT kft.

Lencse Gábor

SZÁMÍTÓGÉP-HÁLÓZATOK

2. kiadás

UNIVERSITAS-GYŐR
NONPROFIT Kft.

Győr, 2008.

Tartalomjegyzék

Előszó

Ez a jegyzet a Széchenyi István Egyetem másodéves villamosmérnök BSc hallgatóinak oktatott *Számítógép-hálózatok* tárgyhoz készült. A tárgy egyrészt az összes villamosmérnöki szakirány számára nyújt alapvető' ismereteket a számítógép-hálózatokról, másrészt fontos alapot jelent a *távközlés-informatika szakirány*[1] több tárgya *(Protokollok és szoftverek, Hálózati operációs rendszerek, Kommunikációs rendszerek programozása* és *Hálózatok biztonsága)* számára.

A jegyzet először a szükséges alapismereteket tárgyalja. Ezután részletesen foglalkozik a legfontosabb lokális hálózati megoldásokkal (Ethernet és vezetéknélküli hálózatok különböző fajtái), majd az internet protokollkészlettel és az útvonalválasztással. Betekintés szintjén bemutatja a legfontosabb hálózati alkalmazásokat és a TCP/IP socket interface programozását. Végül egy rövid összefoglalást ad a számítógép-hálózatok teljesítőképesség-vizsgálatára alkalmazható eljárásokról és az eredmények megjelenítésének módszereiről, trükkjeiről. A függelékben található egy rövid Unix összefoglaló, amivel a tárgy laborgyakorlatain használt Linux operációs rendszer kezeléséhez kívánunk segítséget nyújtani a hallgatóink számára.

[1] A szakirányt csak a nappali tagozatos hallgatóink számára hirdetjük meg.

Az érdeklődő olvasók az egyes témaköröknél bőségesen találnak hivatkozásokat olyan művekre, melyek az adott témakört részletesebben tárgyalják. Ezek között szerepelnek magyar nyelvűek is, de a többségük angol nyelvű. Erősen bátorítjuk a hallgatóinkat az angol nyelvű szakirodalom olvasásának gyakorlására, mert mérnökként a megoldandó feladataikhoz sok esetben csak angol nyelvű szakirodalom áll majd rendelkezésükre.

Ha valaki hibát fedez fel ebben a jegyzetben, akkor azt a távközlés-informatika szakirány honlapján [20] a tárgynál megtalálható módon tudja bejelenteni.[2] Ugyanott megtalálható az aktuális hibajegyzék elérhetősége is.

Jelen kiadás a 2007. évi első kiadás javított változata. Köszönetemet fejezem ki mindazoknak, akik a hibákat bejelentették, és ezzel hozzájárultak a jegyzet minőségének javításához.

Lencse Gábor

[2] A tárgy hallgatói az elsőként bejelentett hibákért valamilyen jutalmazásban részesülnek, ennek módját és mértékét félévenként állapítjuk meg.

1. fejezet

Bevezetés

E jegyzet terjedelme nem teszi lehetővé, hogy a számítógép-hálózatok létrejöttének történetével, fejlődésével részletesen foglalkozzunk.[1] Először definiáljuk, hogy mit tekintünk *számítógép-hálózatnak,* majd megvizsgáljuk, mi is a célja, feladata.

Definíció: A **számítógép-hálózat** autonóm (önálló működésre képes) számítógépek összekapcsolt (információcserére képes) rendszere.[2]

De miért használunk számítógép-hálózatokat, mi a *számítógép-hálózat célja, feladata?*

kommunikáció ember-ember között, például: e-mail, IP telefon, hírcsoportok, stb.

erőforrás-megosztás - például adatok, háttértár, processzor, memória, perifériák megosztása más számítógépek, illetve azok felhasználói számára

[1] Ebben a fejezetben főleg az irodalomjegyzékben található [17] könyvre támaszkodunk, amit - mint a témában alapművet - minden érdeklődő számára ajánlunk.

[2] Ma már ennél szélesebb értelemben is használjuk.

nagy megbízhatóság erőforrás többszörözéssel, akár különböző telephelyeken (például számítási kapacitás többszörözése vagy adatbázisok replikálása katasztrófa elleni védekezésre)

költséghatékonyság feladatszétosztással: szuperszámítógép helyett munkaállomások csoportja (lényegesen olcsóbban elérhető a kívánt számítási kapacitás, természetesen csak bizonyos feladatosztályra hatékony)

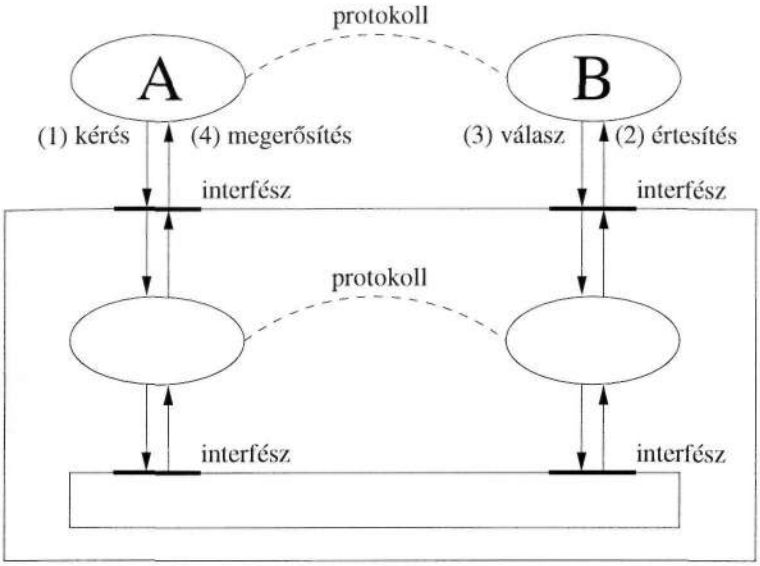
Hogyan valósíthatók meg ezek a célok és más hasonlók? Ez sajnos *nem* tartozik vizsgálódásunk körébe[3], a fenti alkalmazások viszont kellő motivációt nyújtanak a számukra nélkülözhetetlen számítógép-hálózatok alapos megismerésére.

1.1. Alapfogalmak

Nemsokára látni fogjuk, hogy nem kevés feladatot kell megoldanunk ahhoz, hogy két egymástól távoli számítógép kommunikálni tudjon egymással. A tervezési bonyolultság csökkentése érdekében a megoldandó feladatot rétegekre bontással strukturálhatjuk. Ezek a rétegek egymásra épülő szinteket jelentenek. Egy réteg az alatta levő (egyszerűbb feladatot megoldó) réteg szolgáltatásait felhasználva annál bonyolultabb, „értéknövelt" szolgáltatást nyújt a felette levő rétegnek. Egy adott réteg vizsgálatakor az alatta levő rétegeket *fekete doboznak* tekinthetjük: nem foglalkozunk azzal, hogy milyen a belső működése, számunkra csak az érdekes, hogy hogyan viselkedik.

A hálózat valamely rétege, mint fekete doboz az 1.1. ábrán látható. A rajz két oldalán található „A" és „B" *entitások,*

[3] Érdeklődő hallgatóink a *távközlés-informatika* szakirányon választ kaphatnak erre a kérdésre.



1.1. ábra. A fekete doboz modell

vagy processzek egymással kommunikálni akarnak. Az „A" *kérést4* intéz az alsóbb réteghez, amely számára fekete dobozként látszik, eltakarva a belső működését. A kérést ez a réteg valamilyen formában eljuttatja „B"-hez *(bejelentés).* „B" egy *választ* küld szintén az alsóbb réteget használva, mely „A" felé egy *megerősítést* továbbít. Példánkban „A" kommunikált „B"-vel. A kommunikáció „nyelve" *a protokoll,* amelyet mindkét entitás ért. Fizikailag azonban nem egymással „beszélnek", hiszen nem „látják" egymást, hanem mindketten az alsóbb szintet kérik meg a kommunikáció lebonyolítására. A fekete doboz belseje hasonlóképpen épül fel, benne két újabb entitás található, valamilyen

kérés, bejelentés, válasz, megerősítés: az 1.2 alfejezetben tárgyalt OSI modell kifejezései.

(a fentitől eltérő) nyelvvel, amelyet mindketten értenek, és az *interfészek* segítségével ismét egy alsóbb rétegen keresztül jut át az információ.

Definiáljuk most formálisan is az új fogalmakat:

Definíció: Az **entitás** (vagy más kifejezéssel: processz) az egyes rétegekben lévő aktív cselekvő. Az azonos rétegben lévő entitásokat *társentitásoknak* (peer entities) nevezzük.

Definíció: A **protokoll** a különböző gépeken futó n-edik szintű (= n-edik rétegben elhelyezkedő) entitások (vagy processzek) egymással való kommunikációja során használt szabályok és konvenciók összessége.

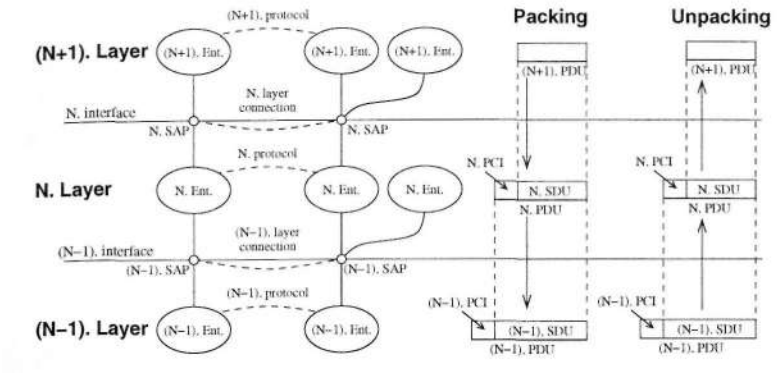
Definíció: A **szolgáltatás** elemi műveletek (primitívek) halmazával írható le, amelyek a szolgáltatást elérhetővé teszik a felhasználó vagy más entitások számára.

Definíció: Az **interfész** az adott réteg által az eggyel felette lévő réteg számára biztosított elemi műveletek és szolgáltatások összessége.

Definíció: A **hálózat architektúrája** rétegeket és protokollokat tartalmaz.

A hálózati architektúra a definíció szerint nem tartalmazza az interfészeket. A következő példánkban szemléletesen látszik, hogy azonos architektúrák, de eltérő interfészek használata esetén működik a kommunikáció. Az „A" entitásunk most legyen egy süketnéma ember, a „B" entitás pedig egy vak és béna ember. Az alkalmazott interfészek tehát eltérőek lesznek, az elsőnél monitor és billentyűzet, míg a másodiknál fejhallgató és mikrofon. Mivel azonban mindketten magyarul beszélnek, meg fogják érteni egymást, feltéve, hogy (például) a „B" gépén egy program az „A" levelét felolvassa, és a „B" beszédét ASCII szöveggé alakítva az „A"-nak elküldi.

Vizsgáljuk meg a rétegek közötti kapcsolatokat általánosan! Az 1.2. ábrán három réteg látható. (Az azonos szinten lévő entitások között kellően nagy fizikai távolság is lehet, például



1.2. ábra. Rétegek közötti kapcsolat

Az 1.2. ábra rövidítései:

- Ent. - Entity (entitás)
- SAP - Service Access Point (szolgáltatás elérési pont)
- PDU - Protocol Data Unit (protokoll adategység)
- PCI - Protocol Control Information (protokoll vezérlési információ)
- SDU - Service Data Unit (szolgáltatás adategység)

Magyarország - USA.) Az *N-edik* szinten az entitások egymással az iV-edik szintű protokollal kommunikálnak logikailag. Fizikailag az alattuk lévő N — 1-edik rétegtől az N — 1-edik szintű interfész megfelelő *szolgáltatás elérési pontján* (SAP) keresztül kérnek szolgáltatásokat. *Az* N — 1-edik réteg szolgáltatás elérési pontjai között N — 1-edik szintű kapcsolat jön létre. Az ábra jobb oldalán az N-edik rétegben láthatjuk az N-edik réteg *protokoll adategységét* (PDU), mely két részből, *a fejrészből* (PCI)

és a *szolgáltatás adategységéből* (SDU) tevődik össze. A PCI gyakorlatilag a szolgálati közlemény az N-edik szintű *társentitás* (peer entity) számára, az SDU pedig az átvinni kívánt adat, amit az N-edik szintű entitás az $N + 1$-edik rétegtől kapott. Az egyes rétegek tehát mindig hozzáteszik a saját vezérlési információjukat a felettük lévő rétegtől kapott információhoz mintegy *beágyazva,* (packing) az elküldendő adatot, míg a másik oldalon felfelé haladva pedig *kibontás* (unpacking) történik, aminek az eredménye a korábban becsomagolt információ lesz.

Természetesen a rétegek száma mindig véges, vagyis a legalsó rétegben már nem vesszük igénybe egy újabb réteg közreműködését, hanem valamilyen módon megtörténik az információ átvitele. A rétegek száma tervezői döntés.

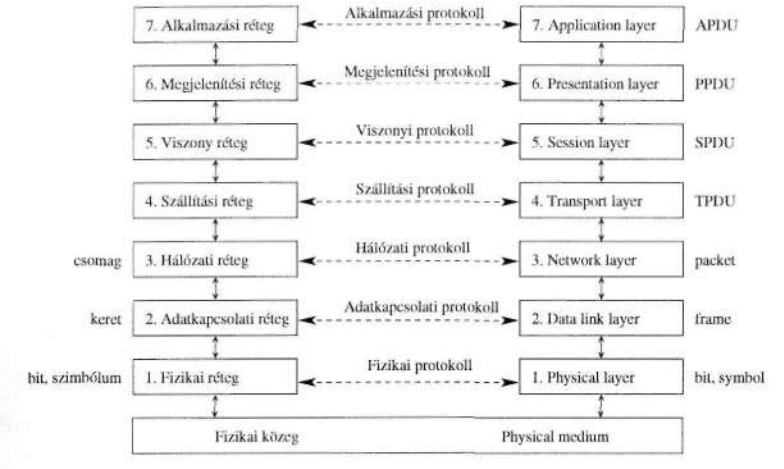
1.2. Az OSI és a TCP/IP referenciamodell

Az *OSI hivatkozási modell* az 1.3. ábrán látható. Ez a modell a Nemzetközi Szabványügyi Szervezet (ISO - International Organization for Standardization) ajánlása. Ezt a modellt hivatalosan OSI (Open System Interconnection = nyílt rendszerek összekapcsolása) hivatkozási vagy referenciamodellnek nevezik, és egy olyan ajánlást definiál, amelynek megfelelő rendszerek egymással elvileg képesek együttműködni. Bár valóban léteznek is OSI szerinti hálózatok, mi ezt csak referenciamodellként használjuk, azaz különböző hálózatok tárgyalásakor az egyes rétegekben definiált funkciókészletre a réteg nevével vagy számával[5] hivatkozunk.

Most röviden összefoglaljuk az egyes rétegek feladatát (alulról felfele haladva).

Fizikai réteg A fizikai réteg (physical layer) feladata az, hogy továbbítsa a *biteket* a kommunikációs csatornán. (Azaz a

[5]A rétegeket alulról felfele 1-től 7-ig számozzuk.



1.3. ábra. OSI referenciamodell

rétegnek biztosítania kell azt, hogy az egyik oldalon elküldött 1-es bit a másik oldalon is 1-ként érkezzen meg, a 0 értékű bit pedig 0-ként.) Ez a réteg tipikusan olyan kérdésekkel foglalkozik, hogy milyen átviteli közeget és milyen csatlakozókat használjunk, milyen kódolásokat (modulációt) alkalmazzunk (például: milyen feszültségszintet használjunk a logikai 1, és mekkorát a logikai 0 reprezentálásához), mennyi ideig tartson egy bit továbbítása, az átvitel megvalósítható-e egyszerre mindkét irányban, miként jön létre és hogyan bomlik le az összeköttetés, ha már nincs szükség rá, stb.

Az átvitel adategysége a *bit* vagy *szimbólum*.

Adatkapcsolati réteg Az adatkapcsolati réteg (data link layer) legfontosabb feladata az, hogy a fizikai szint szolgáltatásainak igénybevételével a hálózati réteg számára hibáktól mentes átvitelt biztosítson *szomszédos állomások*

között. Ehhez az átviendő információt keretekbe (frame) szervezi (ezek tipikusan néhány száz vagy néhány ezer bájtból állnak), amelyek az adatokon kívül természetesen „szolgálati közleményeket" (azaz a társentitásnak szóló vezérlő információt) is tartalmaznak (például melyik állomás küldi melyik állomásnak, a címzettnek mit kell a kerettel tennie) ezeket a megfelelő sorrendben elküldi a célállomásnak, majd végül feldolgozza a vevő által visszaküldött nyugtázó kereteket (aminek során esetleg néhány keretet újra kell adnia).
A kommunikáció adategysége tehát a *keret.*

Hálózati réteg A hálózati réteg (network layer) fő feladata az, hogy az adatkapcsolati réteg szomszédos állomások közötti átviteli képességére építve megoldja a *csomagok* eljuttatását a forrásgéptől a célgépig. A legfontosabb kérdés itt az, hogy milyen útvonalon kell a csomagokat a forrásállomástól a célállomásig eljuttatni. A *torlódásvédelem* (congestion control) megvalósítása is e réteg feladata. Ebben a rétegben az adategységet *csomagnak* nevezzük.

Szállítási réteg A szállítási réteg (transport layer) legfontosabb feladata az, hogy kijavítsa a hálózati rétegben előforduló esetleges hibákat (például csomagvesztés, sorrendcsere), és így végponttól végpontig terjedő megbízható átvitelt biztosítson.
Ebben a rétegben (a fölötte levő rétegekhez hasonlóan) nincs külön speciális neve az adategységnek, egyszerűen csak *szállítási szintű adategységnek* nevezzük.

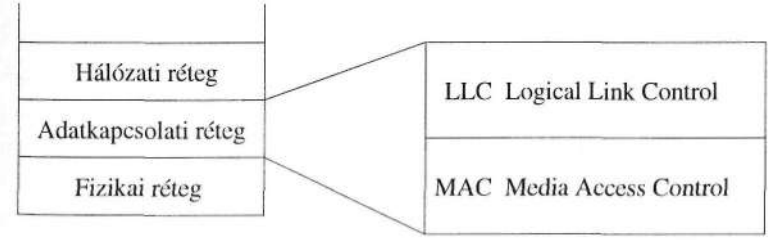
Viszony réteg A viszony réteg (session layer) feladatára jó példa az ellenőrzési pontok alkalmazása. Nagy fájlok átvitelekor a kapcsolat megszakadása esetén az utolsó ellenőrzési ponttól folytatható az átvitel.

Megjelenítési réteg A megjelenítési réteg (presentation layer) tipikus feladata az adatok szabványos módon történő kódolása. Például ASCII, EBCDIC, Unicode; a különböző számábrázolások, stb.

Alkalmazási réteg Az alkalmazási réteg (application layer) számos hálózati szolgáltatás protokollját tartalmazhatja, például: elektronikus levelezés, távoli bejelentkezés, különféle fájlátviteli és elérési protokollok. (Ezeket „látja" egy átlag felhasználó a hálózatból.)

A továbbiakban az egyes rétegek feladatával részletesen meg fogunk ismerkedni. Azonban már a fenti rövid összefoglalásból is látszik, hogy némely rétegnek több, másiknak kevesebb feladata van. így természetesen a későbbiekben más-más súllyal fognak szerepelni az egyes rétegek.

Most még annyit említünk meg, hogy az OSI modellben az adatkapcsolati rétegbe nagyon sok funkció került, ezért az IEEE 802 szabványban ezt a réteget két alrétegre bontották. Közülük az alsó a *MAC* (Media Access Control - közeghozzáférés vezérlés), a felső az *LLC* (Logical Link Control - logikai kapcsolatvezérlés) alréteg.



1.4. ábra. Az adatkapcsolati réteg alrétegei

Az alrétegek az alábbiak szerint osztoznak az adatkapcsolati réteg feladatain:

- A **MAC alréteg** feladata meghatározni, hogy az adott pillanatban az állomások közül melyik adhat a csatornán.

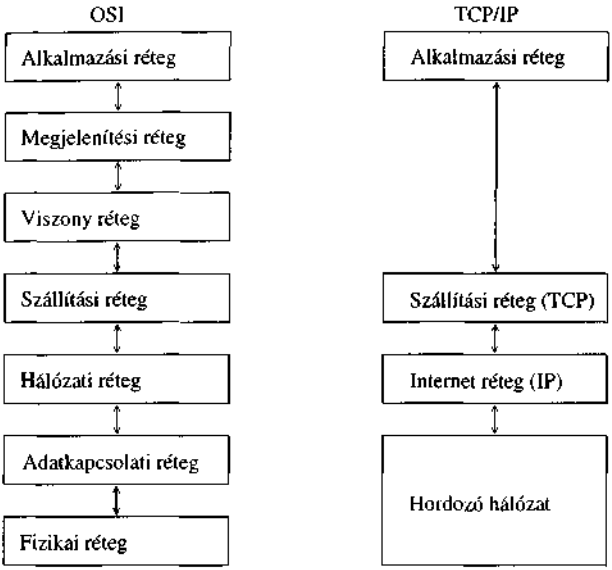
- Az **LLC alréteg** feladata a *forgalomszabályozás*[6] (flow control), a hibajavító kódolás, a nyugtázás és (szükség esetén) az ismétléskérés.

Az OSI modell a gyakorlatban nem terjedt el, helyette a *TCP/IP* modell kapta a nagyobb szerepet, de mint referenciamodell jól használható, mert szerepelnek benne a megvalósítandó funkciók, és hogy azok hol helyezkednek el, hogyan épülnek egymásra.

Az 1.5. ábrán jól láthatók a különbségek: a TCP/IP modell kevesebb réteget definiál, mint az OSI modell.

A TCP/IP modell *IP* (angolul *Internet Protocol,* magyarul internet protokoll) rétege megfelel az OSI modell hálózati rétegének, a *TCP (TransmÁssion Control Protocol* - átvitel vezérlési protokoll) rétege pedig a szállítási rétegnek. A TCP/IP modellben a *hordozó hálózat* az OSI modell két alsó rétegének a funkcióját látja el. Teljesen hiányzik viszont a megjelenítési és a viszony rétegek megfelelője. Az alkalmazások természetesen tartalmazhatják ezeket a funkciókat, de nem feltétlenül találhatók meg bennük. (Például egy hagyományos f tp programnál megszakadás esetén kezdhetjük elölről a letöltést, de vannak olyan fájlletöltő programok, amelyek képesek folytatni a megszakadt letöltést.)

[6] Egy lassabb állomásnak lehetősége van arra, hogy egy gyorsabb állomás neki szóló adását fékezze, és a gyorsabb állomás csak olyan sebességgel adjon, hogy a lassabb képes legyen feldolgozni.



1.5. ábra. OSI és a TCP/IP referenciamodellek összehasonlítása

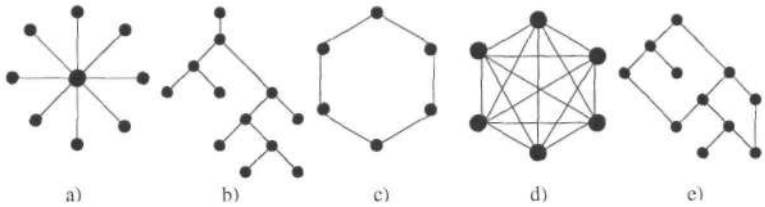
1.3. Hálózati topológiák

Egy számítógép-hálózat fontos jellemzője annak topológiája[7]. Attól függően, hogy az állomásainkat összekötő hálózat pont-pont kapcsolatokból áll, vagy üzenetszórásos csatornát tartalmaz, más-más topológiák lehetségesek. A *pont-pont összeköttetés azt* jelenti, hogy minden hálózati csatorna pontosan két csomóponthoz kapcsolódik, míg az *üzenetszórásos csatorna* esetén a csatornára (bizonyos határon belül) tetszőleges számú állomás kapcsolódhat.

[7] A topológiát a matematikusok *gumi geometriának* is nevezik, mert csak az számít, hogy mely pontok mely pontokkal vannak összekötve, és nem foglalkozik azok tényleges elhelyezkedésével, a köztük levő távolságokkal.

1.3.1. Pont-pont összeköttetés esetén

Ötféle topológiát különböztetünk meg: csillag, fa, gyűrű, teljes és szabálytalan (1.6. ábra).



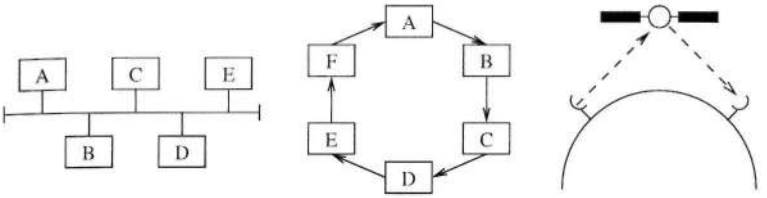
1.6. ábra. Topológiák: a) csillag, b) fa, c) gyűrű, d) teljes, e) szabálytalan

- A **csillag** topológiában egy központhoz (csillagpont) kapcsolódik az összes többi állomás.
- A fa struktúra egy körmentes gráf, így bármely két pontja között csak egy útvonal létezik.
- A **gyűrű** topológia gráfja egy körből áll, minden állomásnak két szomszédja van.
- A **teljes** gráf megoldásban minden állomás minden állomással közvetlenül össze van kötve. A kapcsolatok száma n csomópont esetén: $n(n-1)/2$.
- **Szabálytalannak** nevezzük mindazt, ami az első négybe nem fér bele. (Természetesen részgráfként tartalmazhatja azok bármelyikét.)

Az első négyet helyi hálózatoknál, a szabálytalant főleg nagy távolságok áthidalásánál alkalmazzák.

1.3.2. Többszörös hozzáférésű csatorna esetén

A topológia lehet busz vagy gyűrű, illetve megkülönböztetjük a műholdas adatszórás topológiáját (1.7. ábra).



a) b) c)

1.7. ábra. Topológiák: a) busz, b) gyűrű, c) műholdas

A busz rendszerben ha valaki ad, azt mindenki hallja. A busz végein hullámimpedanciával való lezárást kell alkalmazni, hogy a jelek ne verődjenek vissza (mintha végtelen hosszú lenne a vezeték).

A gyűrű megvalósítása pont-pont kapcsolatokból áll. Úgy lehet mégis üzenetszórásos csatornaként használni, hogy az állomások ismételnek, míg az információ vissza nem ér az adóhoz, és az eredeti adó vonja ki a gyűrűből. így minden állomáson keresztülhalad az információ, tehát lehetőség van arra, hogy akár mind az összes, akár csak a címzett vegye azt, hasonlóan a busz topológiához.

A műholdas rendszerben az állomások nem látják egymást (azaz közvetlenül nem képesek egymással kommunikálni, csak a műholdon keresztül), a műholdat a földi állomások egy bizonyos frekvencián szólítják meg, és az egy másik frekvencián válaszol nekik.

1.4. MAC protokollok

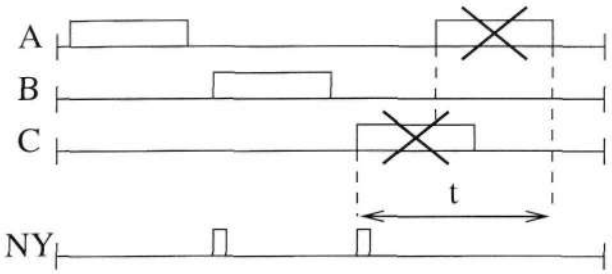
Most megismerünk néhány MAC protokollt. Ezek közül némelyeket olyan rendszerhez fejlesztettek ki, amelyet már nem használnak, de a történeti érdekességen túl azért is érdemes[8] mindegyiket alapötlet szintjén megismerni, mert a hálózatok fejlődésével ismét előállhat olyan helyzet, amikor alkalmazhatók lesznek.

1.4.1. ALOHA

Az ALOHA protokollt rádiós rendszerre tervezték. *Master-slave* (mester-szolga) hierarchiában levő gépek közötti kommunikációt biztosít. A kitüntetett állomást (master) két csatorna köti össze a többi állomással (üzemi és nyugtázó). Egy slave bármikor adhat az üzemi csatornán, azonban ha egy másik slave egy időben adott vele, akkor ütközés lép fel. Ütközés esetén mindkét slave kerete elveszik akkor is, ha az átlapolódás csak részleges. Amennyiben a master helyesen vette az adást, nyugtát küld a slave-nek. Mivel ezt a nyugtázó csatornán végzi, nem léphet fel ütközés (ugyanis ezen a csatornán csak a master adhat). Ha a slave nem kap nyugtát az elküldött keretről, akkor újra leadja azt, feltételezve, hogy az előző kerete ütközést szenvedett és elveszett. Egy ilyen ütközést szemléltet az 1.8. ábra. A protokoll előírja az azonos kerethosszak használatát.

Végtelen populációjú modell szerint, ha az igények Poisson eloszlásúak, a maximális kihasználtság legfeljebb 18% lehet, lásd az 1.14. ábrát a 19. oldalon. A görbén láthatjuk, hogy a kihasználtság egy határig nő, de aztán ahogy a próbálkozások

[8] Ezen kívül didaktikailag is hasznosak: némelyik ismerete elősegíti a másik megismerését, valamint teljesebb képet nyerünk a lehetőségekről, ha olyan MAC protokollokkal is találkozunk, amelyeket a jelenleg elterjedtebb hálózatokban éppen nem alkalmaznak.



1.8. ábra. ALOHA keretek ütközése három állomás esetén (t = ütközés miatt elveszett idő)

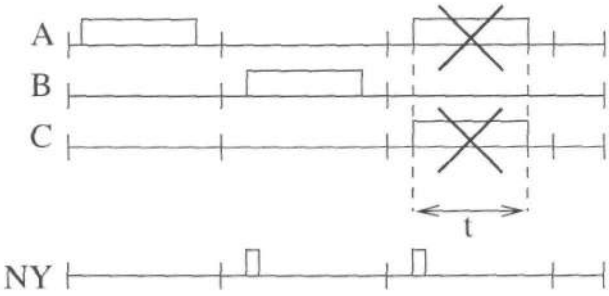
száma emelkedik, az áteresztőképesség egyre csökken. Ez természetesen az ütközések egyre növekvő aránya miatt van így.

1.4.2. Réselt ALOHA

Az ALOHA protokolljából nyerjük időrésekkel való kiegészítéssel. Az üzemi csatornán egymással versengő slave állomások az időrés elején kezdhetnek adni. Az időrések kezdetét a master által kibocsátott szinkron jel jelzi. (Egy időrés természetesen valamennyivel hosszabb az egy keret leadásához szükséges időnél, ez véd a terjedési idő okozta gondokkal szemben.) Az időrések használata miatt az ütközések *teljesek* (1.9. ábra). így ha ütközés történik, akkor azzal 1 időrést veszítünk, szemben az ALOHA protokollal, ahol szerencsétlen esetben két keret ütközése közel 2 keretidőnyi veszteséget okozhatott. A maximális kihasználtság így 36%-ra nőhet, lásd az 1.14. ábrát.

1.4.3. CSMA

A CSMA-t (Carrier Sense Multiple Access = vivőérzékeléses többszörös hozzáférés) kábeles rendszerekben alkalmazzák. Itt

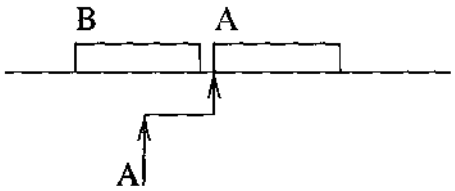


1.9. ábra. Réselt ALOHA keretek ütközése három állomás esetén (t = ütközés miatt elveszett idő)

biztonsággal megoldható, hogy az állomások figyelik a csatornát, és addig nem adnak, amíg az foglalt. A protokollnak több változata van.

1-perzisztens CSMA

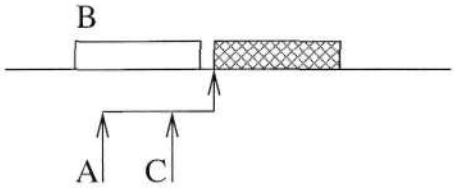
Az „A" állomás belehallgat a csatornába. Ha foglalt, addig vár, míg a „B" állomás be nem fejezte a keretet (közben figyeli a csatornát), és utána azonnal adni kezdi a sajátját (1.10. ábra).



1.10. ábra. 1-perzisztens CSMA

A módszer hátránya: ha a „B" állomás adása alatt az „A" és a „C" állomás is adásra kész, amikor „B" befejezi az adását,

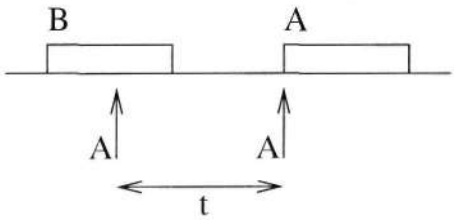
mind a kettő adni kezd; ekkor kereteik ütköznek és elvesznek (1.11. ábra).



1.11. ábra. Ütközés 1-perzisztens CSMA esetén

Nemperzisztens CSMA

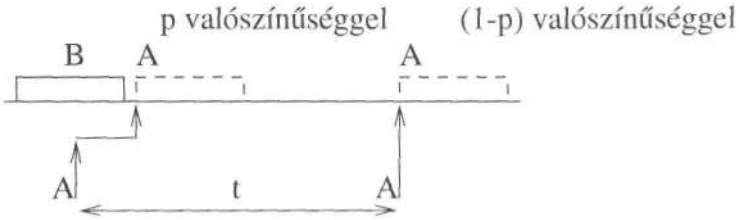
Az „A" állomás belehallgat a csatornába, mivel foglalt, *t* ideig vár, majd újra megvizsgálja, hogy szabad-e a csatorna. Ha szabad, akkor ad, ha nem, megint vár *t* ideig és újra próbálkozik (1.12. ábra). Ez abban az esetben előnyös, ha sokan akarnak adni, mert nagy valószínűséggel elkerülik azt, hogy egy keret befejeződésekor többen egyszerre adjanak.



1.12. ábra. Nemperzisztens CSMA

p-perzisztens CSMA

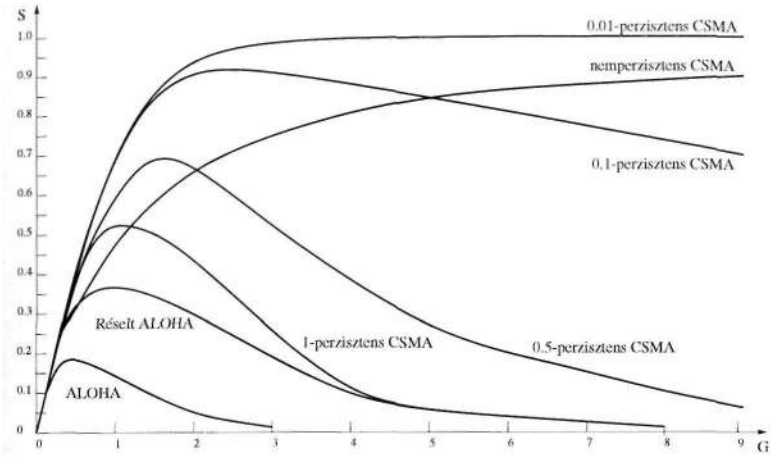
Az „A" állomás belehallgat a csatornába, ha foglalt, addig vár, míg „B" állomás be nem fejezte a keretet, és utána p valószínűséggel adni kezdi sajátját, illetve $(1 - p)$ valószínűséggel í ideig vár, majd újra próbálkozik $(0 < p < 1)$ (1.13. ábra).



1.13. ábra. p-perzisztens CSMA

CSMA változatok összehasonlítása

Az **1.14.** ábra azt sugallja, hogy minél kisebb a perzisztencia, annál jobb a csatornakihasználtság. Ez igaz is, de egy fontos tényről nem szabad megfeledkeznünk. A csatorna kihasználtság növelésének ára van. Ha a 0,01-perzisztens CSMA-t nézzük, ilyen nagy kihasználtság eléréséhez az kell, hogy egy adásra kész állomás csak 0,01 valószínűséggel kezdjen adni, miután a másik abbahagyta. Ennek az lesz a következménye, hogy az állomásoknak igen sokáig kell várniuk! A kihasználtság magas, viszont a felhasználók sokat várakoznak. Összegzésként megállapíthatjuk tehát, hogy egyik protokollt sem nevezhetjük jobbnak a másiknál, viszont a forgalom mértékétől függően egyik vagy másik hatékonyabb lehet.



1.14. ábra. Véletlen hozzáférésű protokollok összehasonlítása a terhelés függvényében mért csatornakihasználtság alapján (S = áteresztőképesség/keretidő, G = próbálkozások száma/keretidő) [17]

1.4.4. CSMA/CD

A CSMA/CD (Carrier Sense Multiple Access with Collision Detection = ütközésérzékeléssel kiegészített vivőérzékeléses többszörös hozzáférés) azt jelenti, hogy valamelyik CSMA-t kiegészítik ütközésérzékeléssel, tehát ha egy állomás nem érzékelt vivőt és adni kezdett, utána figyeli a csatornát, és ha egy másik állomás adását érzékeli, akkor abbahagyja az adást. Ezt teljesítménymméréssel valósítja meg: ha nagyobb teljesítményt érzékel, mint amit ő kiadott, ütközés történt. Ez a módszer azt feltételezi, hogy a két legtávolabbi állomás között is csak olyan kis csillapítás megengedett, hogy az állomások még érzékeljék

egymás teljesítményét.[9] A protokoll továbbfejlesztett változatát alkalmazzák az Ethernet hálózatoknál, így ezzel részletesen fogunk foglalkozni.

1.4.5. Token Ring

A Token Ring (vezérjeles gyűrű, IEEE 802.5, 1.15. ábra) már elavult hálózatnak számít, így ezzel nem fogunk részletesen foglalkozni, de a MAC protokollját érdemes egy kicsit megismerni. A főbb tulajdonságai:

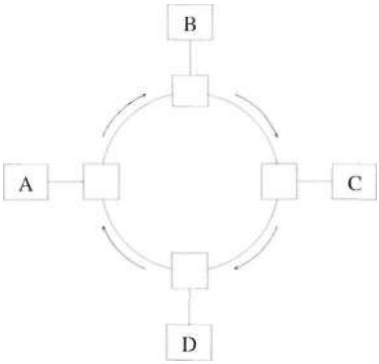
- gyűrű topológiát használ (pont-pont közötti kapcsolatok fizikailag)
- a token (vezérjel) egy speciális keret
- az adhat, akinél a token van
- többi állomás ismétel (a címzett tárolja is a keretet)
- a tokent megszabott idő után tovább kell adni
- ütközés nincs, így jó kihasználtság érhető el

1.4.6. Token Bus

A Token Bus (vezérjeles sín/busz, IEEE 802.4, 1.16. ábra) is a múlté, itt is csak a MAC protokolljára vetünk egy pillantást. Főbb jellemzői:

- a topológia: busz/sín

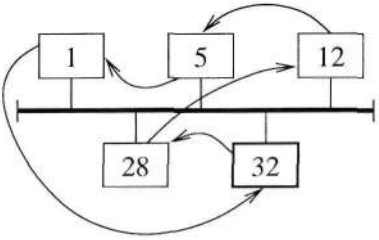
[9] Vegyük észre, hogy az egyre kifinomultabb protokollok (ALOHA, réselt ALOHA, CSMA, CSMA/CD) egyre több feltételt követelnek meg az alkalmazhatóságukhoz!



1.15. ábra. Token Ring

- az állomások a token továbbítása szempontjából gyűrűt alkotnak (logikai gyűrű), azaz mindegyik állomás tudja, hogy melyik állomástól kapja és kinek adja tovább a vezérjelet
- egy állomás adatkeretet bármely állomásnak küldhet

A logikai gyűrűbe való belépés versengéses protokollal történik, ahol ütközés lehetséges, de az adatforgalom a tokenes MAC protokoll miatt itt is ütközésmentes.



1.16. ábra. Token Bus

2. fejezet

Helyi hálózatok

A számítógép hálózatokat kiterjedésük alapján általában a következő kategóriákba szokták sorolni:

LAN (Local Area Network - helyi hálózat)
A hálózat átmérője néhány 100 méter (esetleg néhány km). Tipikusan egy épület vagy egy telephely számítógépeit kötikössze. Ide soroljuk az Ethernet hálózatok különféle fajtáit és a már elavultnak számító vezérjeles gyűrű és vezérjeles busz hálózatokat. Ujabban egyre terjednek a WLAN (wireless LAN - vezetéknélküli LAN) megoldások.

MAN (Metropolitan Area Network - nagyvárosi hálózat)
A hálózat átmérője néhányszor 10 km. Régebben első számú képviselőnek számított az FDDI, és ide tartozott a gyakorlatban nem sokat használt DQDB is. Ma a MAN részének tekintjük a - hozzáférési hálózatként használt - ADSL-t, ADSL2-t és VDSL-t is. A kábel TV fölötti adatátviteli megoldások (pl. DOCSIS) sorolhatók még ebbe a kategóriába.

WAN (Wide Area Network - nagy kiterjedésű hálózat)
A hálózat átmérője néhány 100 km-től terjedhet a kontinenseket behálózó hálózatokig. Technológiailag ide tartoznak: X.25, frame relay, bérelt vonal, SDH, ATM, stb.

A csoportosítás meglehetősen hozzávetőleges. A WLAN hálózatokat is szokták hozzáférési hálózatként használni akár nagyvárosban[1] is, de még inkább ritkán lakott területeken, illetve az üvegszálas Ethernet hálózatok bizonyos fajtái is használhatók ilyen célra!

Megkülönböztethetjük még a *Personal Area Network* (személyes térbeli hálózat) kategóriát is, ami tipikusan az lm - 10 m átmérőjű hálózatokat jelenti, ezeket azonban nem feltétlenül (klasszikus értelemben vett) számítógépek, hanem egyéb eszközök, például mobiltelefonok, PDA-k, stb. csatlakoztatására használják. A technológiák közül ide sorolható például a Bluetooth. Ezzel a kategóriával e jegyzet keretében mélyebben nem foglalkozunk.

2.1. Ethernet hálózatok

Az Ethernet hálózatokról több könyv is elérhető magyar nyelven, ezek közül egy áttekintő mű: [6], melynek eredeti angol változata is beszerezhető: [7]. Csak a Fast Ethernettel foglalkozik: [13].

Az Ethernet hálózatokat eredetileg három együttműködő cég, a DEC[2], az Intel és a Xerox alkotta meg.[3] Ezt 1980-ban publikálták az ún. *kék könyvben* (Blue Book). Azután 1982-ben kiadták a második verziót is: Ethernet v2.0, ami néhány dologban eltér az eredetitől.

[1] WMAN-nak kellene nevezni!

[2] Digital Equipment Corporation, ami később beolvadt a Compaq-ba, ami viszont azután egyesült a HP-vel.

[3] Kezdőbetűik alapján ezt a verziót DIX Ethernetnek is hívják.

Az IEEE lényegében átvette az Ethernet előírásait, de átfogalmazta, precízebbé tette, valamint bevezetett néhány módosítást is, ez lett az IEEE 802.3 szabvány. Mindezek csak a (nemsokára ismertetésre kerülő) vastag koaxiális kábel alapú technológiát tartalmazták. Azóta sok más közeget is használunk (vékony koaxiális kábel, csavart érpárnak és üvegszálnak többféle típusa), ezek az IEEE szabványban kiegészítésként szerepelnek, amit a szabványszámhoz írt kisbetűvel jelölünk, például IEEE 802.3a. A továbbiakban az egyszerűbb szóhasználat kedvéért az IEEE 802.3 családba tartozó összes technológiát egyszerűen csak Ethernetnek hívjuk.

2.1.1. Ethernet hálózatok fajtáinak áttekintése

Az Ethernet hálózatok fajtáit a következőképpen jelöljük. A *Base* szócskával fejezzük ki, hogy a fizikai rétegben alapsávi kódolást használunk (és nem valami vivőfrekvenciát modulálunk). A Base szócska elé írjuk az átviteli sebességet Mbit/s-ban megadva. A korábban kifejlesztett koaxiális kábel alapú típusoknál a Base után egy számjegy áll, ami (a típus azonosításán túl) megadja a szegmenshossz közelítő értékét 100 m-ben kifejezve: 10Base5 és 10Base2. Az összes többi esetben a Base után álló betű, betűkombináció vagy betű és számkombináció csupán azonosítja a típust, és nem hordoz közvetlenül szegmenshossz információt. Például: lOBaseT, 100BaseT4, lOOOBaseSX.

A 2.1. táblázatban röviden áttekintjük az Ethernet hálózatok fajtáit.[4] Az első kettő (10Base5 és 10Base2) busz topológiát használ, a többi csillagot. A busz topológia előnye, hogy nem igényel külön aktív eszközt, hátránya viszont, hogy ha valahol megsérül, akkor az egész szegmens üzemképtelen lesz. A csillag topológia esetében a csillagpontban valamilyen (2.1.7.-ben megismerendő) aktív eszköz található. Ebben az esetben egy

[4] Van 10 Gbit/s sebességű Ethernet szabvány is, de azt nem tárgyaljuk.

szegmens sérülése miatt csak az adott szegmensen levő állomás nem tud kommunikálni. Amint a táblázatban láthattuk, a busz topológiájú hálózatok koaxiális kábelt használnak. Ennek nyilván az az oka, hogy kifejlesztésük idején az akkori technológia mellett ez a kábel volt alkalmas a megbízható átvitelre az akkor meglehetősen nagynak számító 10Mbit/s sebesség mellett. A technológia fejlődése azonban lehetővé tette, hogy árnyékolatlan csavart érpárat használjunk ugyanakkora, sőt egyre nagyobb átviteli sebesség mellett. (Az átviteli közegekről bővebben: 2.1.2.) Az Ethernet hálózatok egyes fajtáival külön-külön is foglalkozunk 2.1.9-ben, de előbb még meg kell szereznünk a szükséges előismereteket.

2.1.2. Ethernet hálózatok fizikai közegei és csatlakozói

Vastag koaxiális kábel

Az Ethernet hálózat legelső fajtája, a 10Base5 50 Ohm-os *vastag koaxiális kábelt*[5] használ átviteli közegként. A busz topológiából adódóan a kábel végét a reflexió elkerülése érdekében hullámimpedanciával (50 Ohm) le kell zárni, ami azt jelenti, hogy a kábel végein a kábel belső (ún. meleg) ere és a külső árnyékolás közé kell bekötni a megfelelő teljesítményre méretezett 50 Ohm-os ellenállásokat, amint a 2.1. ábra is mutatja.

A vastag koaxiális kábel használatának egyik jelentős hátránya az, hogy a kábelt (a szerkezetének megóvása érdekében) csak kellően nagy sugarú ívben szabad hajlítani, így a szerelése nagy gondosságot igényel.

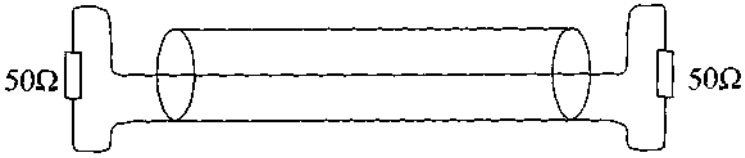
A kábelen 2,5 méterenként előre kialakított és megjelölt csatlakozási pontok találhatók. Ezeken a helyeken a kábel szerkezetét úgy alakították ki, hogy alkalmas legyen a *transcei-*

[5] A kb. hüvelykujjnyi vastagságú, sárga színű (egyesek szerint kerti locsolócsőre emlékeztető) koaxiális kábelt angolul *yellow cablenek* is nevezik.

Típus	Átviteli közeg	Max. szegm. hossz	Csatlakozó
10Base5	vastag koax	500 m	vámpír
10Base2	vékony koax (RG 58)	185 m (200 m)	BNC, T-dugó
10BaseT	UTP (Cat3)	100 m	RJ 45
10BaseF	üvegszál	2000 m	SC, ST
100BaseTX	UTP (Cat5)	100 m	RJ 45
100BaseT4	UTP (Cat3)	100 m	RJ 45
100BaseT2	UTP (Cat3)	100 m	RJ 45
100BaseFX	üvegszál	2000 m	SC, ST
1000BaseT	UTP (Cat5e)	100 m	RJ 45
1000BaseSX	üvegszál (MMF 62,5 μm)	220 m – 275 m	SC, ST, LC, MTRJ
1000BaseSX	üvegszál (MMF 50/125 μm)	500 m – 550 m	SC, ST, LC, MTRJ
1000BaseLX	üvegszál (MMF)	550 m	SC, ST, LC, MTRJ
1000BaseLX	üvegszál (SMF, 9 μm)	5 km	SC, ST, LC, MTRJ
1000BaseCX	STP	25 m	árnyékolt RJ 45

2.1. táblázat. Ethernet hálózat fajták néhány jellemzője [6].

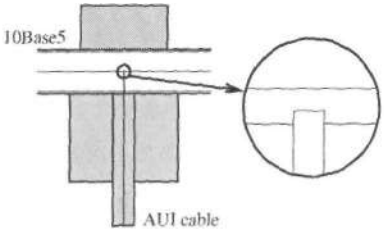
verek (magyarul adóvevők) *vámpírcsatlakozóval* történő csatlakoztatására. A vámpírcsatoló tüskéje éppen annyira hatol be



2.1. ábra. Kábellezárás

a kábelbe, hogy már biztos kontaktust teremtsen a kábel meleg erével, de ne vágja el azt. A 2.2. ábrán ez kinagyítva is látható. A transceiver látja el az adás, a vétel és az ütközés érzékelés feladatát. A transceiver az *AUI kábelen* (Attachment Unit Interface cable) keresztül 4 érpár segítségével kommunikál a számítógépben található hálózati kártyával: adás, vétel, vezérlés, tápellátás. (Régi hálókártyákon még megfigyelhető a 15 érintkezős AUI csatlakozó.) Az Ethernet hálózatok többi típusánál a transceiver a hálózati kártyán található!

A 10Base5 szabvány szerint egy szegmensre legfeljebb 100 állomás kapcsolható.



2.2. ábra. Transceiver vámpírcsatlakozóval

Vékony koaxiális kábel

A tipikusan szürke vagy fekete színű *RG 58-as* jelzésű vékony koaxiális kábel már jóval könnyebben kezelhető, mint a vastag. Egyrészt kisebb a hajlítási sugár, másrészt könnyebb a szerelése, mert közönséges ipari *BNC csatlakozókat* használhatunk. 10Base2 hálózat építésekor a kábelt minden olyan helyen el kell vágni, ahol esetleg számítógép csatlakoztatására lehet szükség. A kábelvégekre *anya típusú* (female) BNC csatlakozó kerül, és a kábelvégeket fali csatlakozó dobozban (face plate) rögzítik. Ahova nem kötnek be számítógépet, ott a két csatlakozót összekötik egy mindkét végén *apa típusú* (male) BNC csatlakozóval szerelt rövid koax kábellel *(átkötés).* Ha egy számítógépet szeretnénk a hálózatba kötni, akkor eltávolítjuk az átkötést, és két megfelelő hosszúságú koax kábellel egy T dugót kötünk be a hálózatba, a T „függőleges" szárát pedig a számítógépben található hálózati kártyához csatlakoztatjuk. Több gép esetén a gépeket egymás után felfűzzük. Természetesen a hálózati kártyák villamosan ekkor is egymással párhuzamosan vannak bekötve, és a 10Base5-höz hasonlóan itt is 50 Ohm x 50 Ohm = 25 Ohm ellenállást látnak egy helyesen lezárt hálózat esetén. A hálózati kártyáknak természetesen elvileg végtelen, gyakorlatilag 100 kOhm nagyságrendű ellenállást kell kifele mutatniuk.

A 10Base2 szabvány szerint egy szegmensre legfeljebb 30 állomás kapcsolható.

Csavart érpáras megoldások

A jelfeldolgozási technológia fejlődése lehetővé tette a közös módusú zavarok hatékony kiszűrését. így a drága és nehezen szerelhető koaxiális kábel helyett ma már csavart érpárat használhatunk az Ethernet hálózatainkhoz. (Például: lOBaseT, lOOBaseTX, lOOOBaseT.) Ezek közös jellemzője, hogy egy kábel 8 eret tartalmaz, amelyeket páronként összesodortak: így 4 érpárat

kaptunk. A sodrás célja, hogy az érpárak mindegyikében (jó közelítéssel) azonos zavarjelek indukálódjanak. Az érpárak közötti elektromágneses csatolás csökkentése érdekében az egyes érpárak *sodrási száma*[6] eltérő. Az ennek következtében fellépő hosszkülönbségből adódó ellenállás-különbség kompenzálására a nagyobb sodrási számú érpárak vastagabbak.

A csavart érpáras kábelek egyik fontos jellemzője az *átvitt frekvenciatartomány*. Ennek alapján definiálták a következő *kábelkategóriákat:*

Cat3 16MHz-ig

Cat4 20MHz-ig

Cat5 100 MHz-ig (eredetileg)

Cat5e 100 MHz-ig + további követelmények teljesítése: NEXT (Near End Cross Talk - közelvégi áthallás), FEXT (Far End Cross Talk - távolvégi áthallás). Aztán megszűnt a Cat5e és a Cat5-nek nevezett tudja azt, amit addig Cat5e-nek hívtak.

Cat6 250 MHz-ig (2002. nyarától szabvány.)

Cat7 600 MHz-ig (Ajánlások vannak rá, de teljes rendszer nem létezik belőle.)

Egyes gyártók ettől eltérő értékeket is garantálhatnak, például 150 MHz. Azt azonban fontos látnunk, hogy az átvitt frekvenciatartomány dimenziója a MHz. Bizonyos gyártók ezzel szemben Mbit/s dimenziójú jellemzőt adnak meg, ami nem összemérhető vele! Az elérhető átviteli sebesség természetesen az alkalmazott kódolástól is függ, ami NEM a kábel jellemzője!

méterenként hány sodrás van

A gigabites hálózatok az átvitt frekvenciatartományon túl a kábellel szemben más fontos követelményeket is támasztanak, úgy mint csillapítás, NEXT (Near End Cross Talk - közelvégi áthallás) és FEXT (Far End Cross Talk - távolvégi áthallás).

Mind az áthallás, mind a külső zavarjelek bejutása ellen védekezhetünk az egyes érpárak, illetve a kábel árnyékolásával. Árnyékolásra kábelharisnyát vagy fémfóliát használnak. A 2.3. ábra különböző megoldásokat mutat be azok szabványos megnevezésével. A „/" jel előtti betűk a kábel külső árnyékolására, az utána állók pedig az érpárak árnyékolására vonatkoznak:

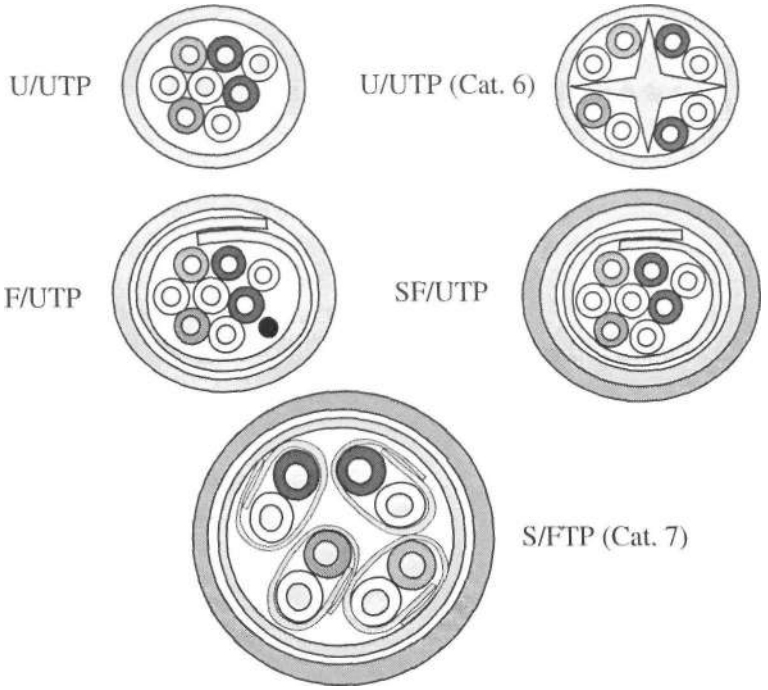
- U: Unshielded = árnyékolatlan
- S: Shielded = kábelharisnyával árnyékolt
- F: Foiled = fémfóliával árnyékolt

Csavart érpáras hálózataink esetén a kábeleket *RJ-45* jelű csatlakozóval csatlakoztatjuk. Ez a telefonos RJ-11 csatlakozóhoz hasonló, de 8 ér csatlakoztatására alkalmas. Az ereket a műanyag borításuk színével azonosítjuk. Az érpárak egyik ere a *narancs, zöld, kék* és *barna* színek közül valamelyik, a vele összecsavart pedig fehér az adott színű csíkkal. Az erek bekötése így a *színsorrenddel* adható meg. A 2.2. táblázatban az EIA/TIA 568 szabvány szerinti A és B színsorrend látható.

Erek sorszáma	1	2	3	4	5	6	7	8
EIA/TIA 568 A	ZF	Z	NF	K	KF	N	BF	B
EIA/TIA 568 B	NF	N	ZF	K	KF	Z	BF	B

2.2. táblázat. Csavart érpáras kábel színsorrendje

A ma legelterjedtebb lOOBaseTX hálózatnál csak két érpárat használnak: egyet adásra, egyet pedig vételre. A helyes működéshez az egyik állomás adóját a másik állomás vevőjére kell



2.3. ábra. Csavart érpáras kábelek [38]

kötni és viszont. Ha két állomást közvetlenül szeretnénk egymással összekötni, akkor tehát *keresztkábelt* (cross-over cable) kell használnunk, amelynek az egyik végének a bekötése az *A,* a másik végének bekötése pedig a *B* sorrendet követi. Az aktív eszközök csatlakozójánál már elvégezték a cserét, ezért aktív eszköz és számítógép közé egyenes kábelre van szükség. (Ez lehet akár A-A, akár B-B bekötésű, az utóbbit szoktuk használni.) Két aktív eszköz összekötésére természetesen keresztkábelt kell használni.

A *strukturált kábelezéssel* a 2.2. alfejezetben külön foglalkozunk, mivel a strukturált kábelezés az Ethernet hálózatokon kívül másra is használható.

Üvegszálas megoldások

Az üvegszál a teljes visszaverődés fizikai jelensége miatt alkalmas átviteli közegnek. Alapvetően két fajtája van: a *többszörös módusú üvegszál* (MMF - Multi Mode Fibre), aminek a magátmérője viszonylag nagyobb (50 um vagy 62,5/ u m) , és az *egyszeres módusú üvegszál* (SMF - Single Mode Fibre), aminek a magátmérője kisebb (9/um).

A többszörös módusú üvegszálban a fény több úton is terjedhet. Természetesen az eltérő úthosszak megtételéhez szükséges idő is eltérő, így a jel diszperziót szenved, aminek az a következménye, hogy az áthidalható távolság kisebb, mint az egyszeres módusúnál.

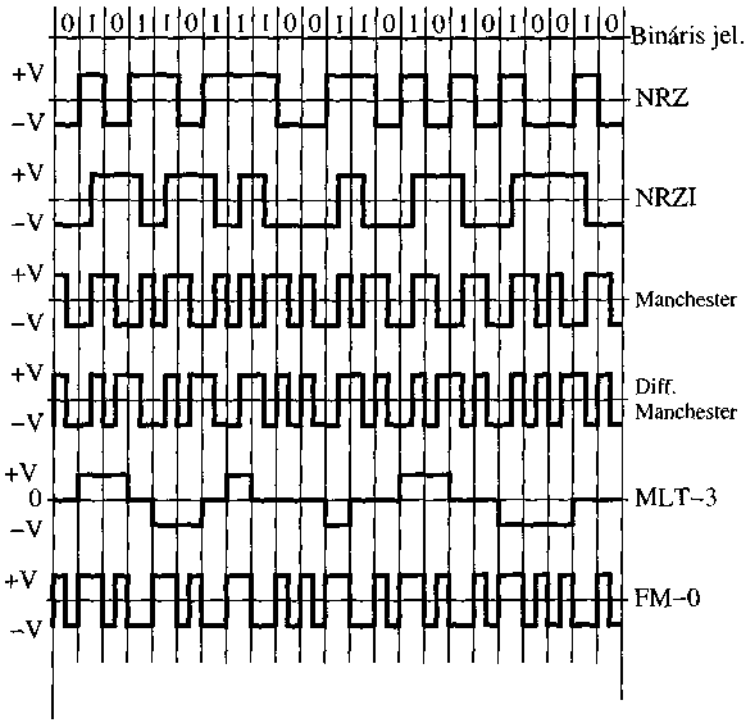
A magátmérők ismeretében érthető, hogy az üvegszálak csatlakoztatása kellően nagy pontosságot igényel. Az egyik legelterjedtebb csatlakozó az SC, ami közelítőleg négyzet keresztmetszetű és csatlakoztatáskor bekattanva rögzítődik, így nem eshet ki, csak viszonylag nagyobb erővel lehet kihúzni. Van amikor a két irány (adás és vétel) üvegszálának csatlakozója egyben van, így egy darab téglalap keresztmetszetű dugóval találkozunk. A másik elterjedt csatlakozó az ST, ami a BNC-hez hasonló, de vékonyabb. Ezt főleg a patch paneleken használják. Egy harmadik megoldás, amivel például 3Com eszközöknél találkozhatunk, az MTRJ. Ez jóval kisebb méretű, és a két szálat együttesen csatlakoztatja (duplex). Újabban egyre jobban terjed az LC csatlakozó. Ez kinézetre az SC-hez hasonlít, de lényegesen kisebb annál, aminek előnye, hogy körülbelül akkora helyet foglal el mint egy RJ 45, így portsűrűségben jó. Általában duplex.

A lefektetett optikai kábelek több (4-8-12, stb.) optikai szálat tartalmaznak, és megfelelő mechanikai védelemmel vannak

ellátva.

2.1.3. Vonali kódolási megoldások

Digitális információ átvitelére sokféle kódolás alkalmazható. Ismerjünk meg néhányat a 2.4. ábra alapján!



2.4. ábra. Kódolási eljárások

NRZ (Non Return to Zero) kódolás esetén egy bit 0 értékénél a jel a teljes bitidő alatt $-V$ feszültségszintet vesz fel, 1 értékénél pedig $+V$ értéket. Gondot jelenthet, ha

hosszú egyes (vagy nullás) sorozat van, mert pozitív (vagy negatív) egyenáramú komponens jelenik meg, valamint a szintváltás hiánya miatt *szinkronvesztés*[1] következhet be.

NRZI (Non Return to Zero Inverted) kódolásnál l-es bit esetén a bitidő közepén invertálni kell a feszültségszintet, 0 esetén nincs váltás. Hosszú 0 sorozat esetén itt is bekövetkezhet a szinkronvesztés.

Manchester kódolás esetén a bitidő közepén 0 értékű bitnél lefelé kell vinni a feszültségszintet, míg I-nél felfelé. Ha azonos értékű bitek érkeznek egymás után, a bitidő határán váltani kell, különben nem. A megoldás előnye, hogy mivel a bitidő közepén mindig van szintváltás, ezért nem következhet be szinkronvesztés. Ennek az ára viszont a megnövekedett sávszélesség igény!

Differenciális Manchester kódolást használva 0-nál a bitidő elején és közepén is van váltás, míg I-nél csak a közepén. Szintén véd a szinkronvesztés ellen, szintén a sávszélesség rovására.

MLT-3 kódolás esetén a jelnek 3 feszültségszintje lehet: pozitív, negatív, és 0. A váltások a következő sorrendben történnek: 0 -» +*V* -» 0 -» -*V* -» 0 -» ... stb. Az 1-esnél a bitidő elején váltás történik a megadott sorrend szerint, 0-nál pedig nincs váltás. Hosszú 0 sorozat esetén itt is bekövetkezhet szinkronvesztés. A sávszélesség felhasználása kedvező.

FM-0 kódolásnál a jelszint a bitidő elején és végén mindig az ellenkezőjére vált, 0-nál a közepén is vált, l-esnél középen nem vált. Az órajelet megtartja.

[1] A vevő nem tudja az adó órajelét követni.

A fentiekből úgy tűnhet, mintha a szinkronvesztés kiküszöbölése és a sávszélesség hatékony felhasználása egymást kizáró követelmények lennének. Ez nincs így, csak fejlettebb kódolási eljárásokra van szükség. Nézzünk meg néhányat ezek közül is!

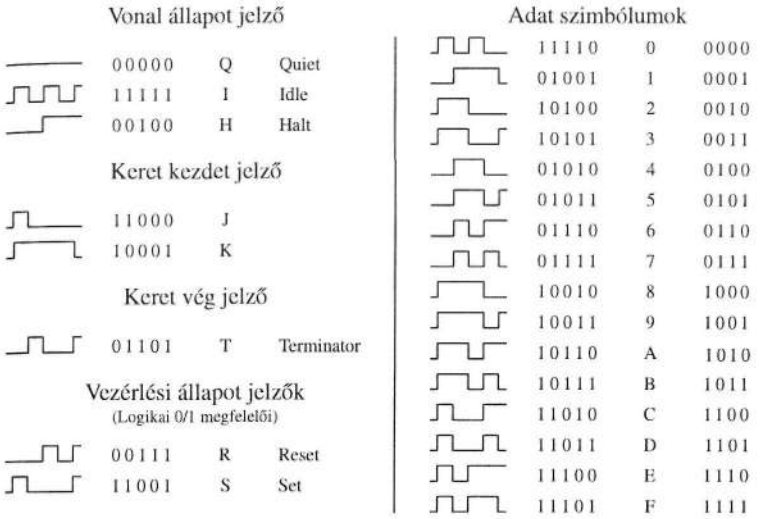
4b/5b kódolás esetén a bemenet minden 4 bitjét egy-egy alkalmas 5 bites szimbólummal helyettesítjük. Mivel így a $2^4 = 16$ helyett $2^5 = 32$ szimbólum áll rendelkezésünkre (2.5. ábra), megoldható, hogy csak olyan szimbólumokat használjunk, melyeknek az elején legfeljebb egy, a végén legfeljebb két nulla értékű bit áll. Ezzel a választással elértük, hogy egymás után maximum három nulla értékű bit helyezkedhessen el. így megakadályoztuk a hosszú 0 sorozat kialakulását, ami azért fontos, mert a 4b/5b kódolás után általában az NRZI kódolást használjuk, ahol a hosszú 0 sorozat szinkronvesztést okozhatna.

A 16 db adatszimbólum mellett van 8 speciális szimbólum, ami például keret eleje/vége, vonalállapot, logikai értékek, stb. jelzésére használható. A további 8 szimbólum érvénytelen.

8b/10b kódolásnál a $2^{10} = 1024$ lehetséges kódszimbólumból $2^8 = 256$ db-ot használunk az adatok kódolására.

8b/6t kódolásnál minden 8 bitet 6 db három szintű (terciális) szimbólummal kódolunk.

PAM-5 kódolásnál 5 szintet használunk: -2, -1, 0, 1, 2. Tehát: ha már nem lehet a kábelen a frekvenciatartományt növelni, a feszültségszintek számát még igen!



2.5. ábra. 4b/5b + NRZI kódolás érvényes szimbólumai [6]

2.1.4. Ethernet hálózatok MAC protokollja

A 10Base5 és a 10Base2 hálózatok MAC protokollja az 1-perzisztens CSMA/CD kettes exponenciális visszatartással kiegészítve. A csavart érpáras hálózatok közül *half duplex*[8] üzemmódban ezt használja például a lOBaseT és a lOOBaseTX.

Hogyan működik ez a protokoll? Ha egy állomásnak van adni valója, belehallgat a csatornába. Ha adást érzékel, akkor megvárja, amíg az abbamarad, közben folytonosan figyeli a csatornát. Amikor szabad(dá vált) a csatorna, azonnal adni kezd. Ha a keret adása során nem érzékel ütközést, akkor a feladatát befejezte. Ha ütközést érzékel (teljesítménymméréssel), akkor egy rövid ideig (32 bit ideje) zavarjelet (álvéletlen bitsorozatot) ad,

[8] Olyan kétirányú átvitel, amikor az egyik és a másik irányban csak egymás után váltakozva történhet az adattovábbítás.

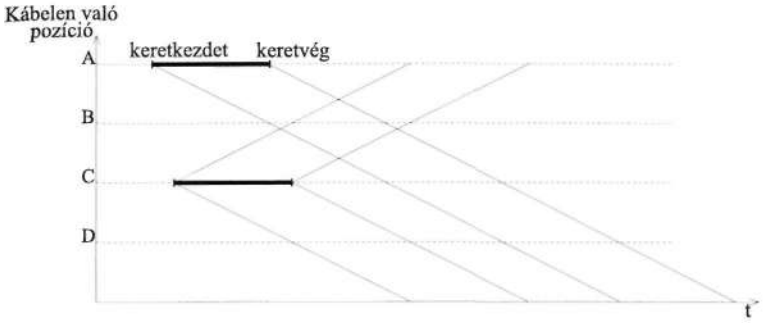
hogy a többi állomás is érzékelje az ütközést. Ezután az adását abbahagyja. Ez most az első ütközése volt, amiben részt vett. Tegyük fel, hogy már az n-edik sikertelen kísérletnél tart. Ekkor a 2^nT hosszú (ahol T egy adott érték) intervallumból véletlenszerűen választott időt vár, mielőtt újra próbálkozna. Az intervallum hossza legfeljebb 2^{10}T-ig nő, a későbbi próbálkozásoknál változatlan, és összesen legfeljebb 16-szor próbálkozik. Mivel az intervallum hossza exponenciálisan nő, ezért még ha nagy számú állomás vett is részt az ütközésben, akkor is kellően hamar megtörténik az ütközés feloldása. (Valamelyik állomásnak sikerül a többit megelőznie, és amikor a többiek érzékelik az adását, akkor már természetesen nem vágnak a szavába.)

Ahhoz, hogy ez az algoritmus használható legyen, feltétlenül szükséges, hogy még a keret adása alatt minden résztvevő állomás felismerje az ütközést! Nézzük meg, mi történne, ha a 2.6. ábrán látható szituáció állna elő! Az ábrán a függőleges tengely mentén helyezkednek el az állomások, a vízszintes tengely pedig az időt ábrázolja. Az „A" és a „C" állomás adása szemmel láthatóan átlapolódik, de az ütközést egyikük sem érzékeli, mert mire a másik állomás kerete odaér hozzájuk, arra már befejezték a saját keretük leadását. Annak érdekében, hogy ezt a szituációt elkerüljük, a keret hosszának[9] el kell érnie egy bizonyos értéket. Helyesen megválasztott kerethossz esetén a probléma nem áll elő: 2.7. ábra. A legkisebb megengedett kerethossz 64 byte.

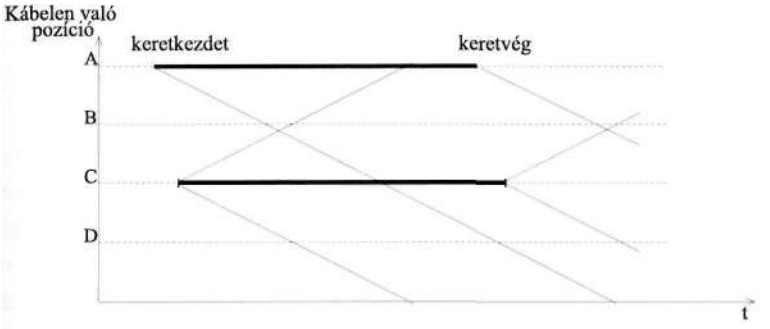
2.1.5. Ethernet keret felépítése

Topológiától és adatsebességtől függetlenül az Ethernet hálózatok összes fajtája a 2.8. ábrán látható keretszerkezetet használja. Az első két mező tulajdonképpen a fizikai réteghez tar-

[9] Valójában adási időtartamának, de ez a 10Base5 és 10Base2 hálózatoknál még ugyanazt jelentette.



2.6. ábra. Ütközés túl rövid keretidő esetén



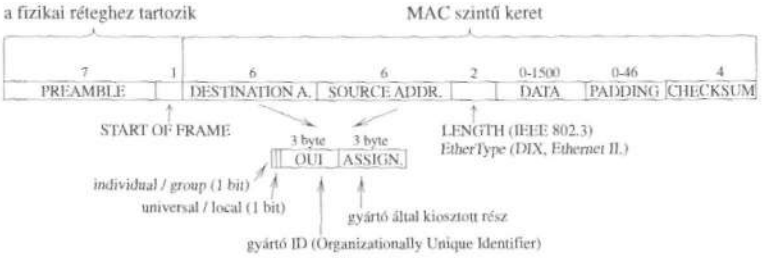
2.7. ábra. Ütközés helyesen megválasztott keretidő esetén

tozik, de ezeket is bele szoktuk érteni a keretbe.

Az IEEE 802.3 keret az alábbi mezőket tartalmazza:

PREAMBLE előtag, (bájtonként: 10101010)

Ez a bitsorozat arra szolgál, hogy a vevő az óráját az adó órájához szinkronizálhassa. (Például 10 Mbit/s sebességű hálózatnál 1 bit ideje: 100 ns, a 7 bájt 56 bitjének ideje tehát 5,6 us.)



2.8. ábra. Az IEEE 802.3 keretszerkezete

START OF FRAME keretkezdet határoló (10101011)
Az előtaghoz képest az utolsó bit l-es értéke jelzi, hogy utána már a célállomás címe következik.

DESTINATION ADDRESS célcím
Bár az IEEE 802.3 megengedi a 2 bájtos hosszt is, gyakorlatilag mindig 6 bájt hosszú. Célszerű volt a többi mező elé tenni, hogy minden állomás minél előbb felismerhesse, hogy kinek szól a keret.

SOURCE ADDRESS forráscím

LENGTH adatmező hossza (IEEE 802.3)
A DIX Ethernet és annak v2.0 változata ezt a mezőt EtherType-nak nevezi, ami azt adja meg, hogy az adatmezőben milyen protokoll adategysége utazik. (A két megoldás akár egy hálózaton belül is használható, ugyanis az aktuális értékéből felismerhető, hogy hogyan kell értelmezni! Ha a mező értéke 0-1500 bájt közötti, akkor hossz, ha ettől eltérő, akkor EtherType.)

DATA adatok
Itt található beágyazva a felsőbb rétegbeli protokoll adategysége.

PADDING kitöltés
A keret minimális hossza 64 bájt, ha nincs annyi, ki kell egészíteni.

CHEKCSUM ellenőrző összeg
Ha megegyezik a számított és a vett érték, akkor a keretet hibátlannak tekintjük, ellenkező esetben a keret sérült.

Az összes Ethernet hálózati kártya egyedi címmel rendelkezik. Ezt úgy valósítják meg, hogy a cím 6 bájtját két részre bontották. Az első 3 bájtot az IEEE osztja ki a gyártók részére.[10] A másik 3 bájtot pedig a gyártók osztják ki az általuk gyártott eszközöknek. Bár az OUI (Organizationally Unique Identifier) elvileg 24 bites, gyakorlatilag csak 22 bitet használnak a gyártók azonosítására, ugyanis az első két bit más célt szolgál: az I/G bit (individual / group) 0 értéke azt jelenti, hogy valóban egy kártya egyedi címéről van szó, az 1 értéke esetén a cím egy *csoportcím* (multicast address). Ha az U/L (universal / local) bit értéke 0, akkor valóban univerzálisan (helyesebben: globálisan) adminisztrált címről van szó, aminek az egyedisége az imént említett módon biztosított, ha pedig a bit értéke 1, akkor a címet a rendszergazda osztotta ki (erre bizonyos protokollok esetén szükség lehet).

Itt említjük meg még a *broadcast címet,* ami 48 darab l-es bitből áll. Az ilyen címre küldött kereteket minden hálózati kártya veszi.

2.1.6. Ethernet keretek hibái

Ethernet hálózatokban akár üzemszerű/helyes működés közben, akár meghibásodás miatt keletkezhetnek *hibás keretek*. Ezeket a következő (nem diszjunkt) csoportokba soroljuk:

[10]A kiosztott azonosítók listája megtalálható: [25]

Runt A kerethossz kisebb, mint 64 bájt. Leggyakoribb oka az, hogy ütközés miatt az állomások abbahagyják az adást. Ez a fajta hibás keret tehát egy hibátlanul működő hálózatban is előfordulhat.

Jabber „fecsegés" A kerethossz nagyobb 1518 bájtnál (a második rétegben mérve).[11]

Keletkezésének lehetséges okai:

- az állomások nem veszik észre az ütközést
- adó (hálózati kártya) meghibásodása

Ez tehát minden esetben hálózati hibát jelent.

Misaligned frame „rosszul elrendezett/igazított" keret. A keret bitjeinek száma nem osztható 8-cal. Ha például runttal együtt fordul elő, akkor nem jelent hálózati hibát.

Bad FCS A számított ellenőrző összeg nem egyezik a vett keret utolsó 32 bitjével. Oka lehet bithiba vagy csonkolódás. Ütközés miatt előfordulhat jól működő hálózatban is, de utalhat hibára is. (Például UTP kábelnél elcserélt ér miatti áthallás *full duplex*[12] módban a két irány között.)

Természetesen egy misaligned frame esetén az ellenőrző összeg is hibás. Runt és Jabber esetén valószínű a misaligned frame és szinte biztos a bad FCS.

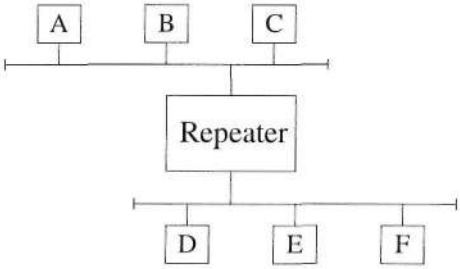
2.1.7. Ethernet hálózatok aktív elemei

A 10Base5 és 10Base2 hálózatok busz topológiájúak, és a működésükhöz elegendő, ha a buszt mindkét végén hullámimpedanciával lezárjuk, valamint természetesen állomásokat kötünk

[11] Azaz 1526 bájtnál az első rétegben mérve.
[12] azonos időben történő átvitel mindkét irányban

rá. Mivel a szegmenshossz korlátos, felmerül az igény több szegmens összekapcsolására. Az összekapcsolás többféle eszközzel is történhet, attól függően, hogy mi a célunk.

A legegyszerűbb esetben, ha csak a szegmenshossz által okozott korlátokat szeretnénk tágítani, elegendő a szegmenseket első rétegbeli funkcionalitással összekapcsolnunk. Ezt valósítja meg a *repeater* (2.9. ábra).



2.9. ábra. Hálózati szegmensek összekapcsolása repeaterrel

A **repeater** olyan hálózati eszköz, amely az egyik portján vett adatokat a többi portján (jelregenerálás után) kiadja. A szegmensekből egy *ütközési tartományt* (collision domain) alakít ki, ha tehát a szegmenseken lévő bármelyik két állomás egyszerre kezdeményez adást, ütközhetnek. (Az első rétegben működő eszköz.)

10Base5 hálózat esetén a szegmensek összekapcsolására az alábbi szabályt[13] kell alkalmazni az összes jelútra vonatkoztatva.

- maximum 5 szegmenst tartalmazhat
- maximum 4 repeatert tartalmazhat

[13]Úgy is hívják, hogy: 5-4-3 szabály.

- **maximum 3** szegmensen lehetnek állomások (a többi csak inter-repeater link lehet)

Természetesen a repeaterek alkalmazása nem mindig felel meg a céljainknak. Az így összekapcsolt szegmenseken levő állomások ugyanazon a sávszélességen osztoznak, nem lehetséges, hogy például a 2.9. ábrán látható hálózat esetében az „A" állomás a „B"-nek, a „D" pedig az „F'-nek egyidejűleg küldjön keretet, hiszen a repeater miatt ezek ütköznek. Ehhez 2. rétegbeli összekapcsolást végző eszközre van szükség, amit *bridgenek* nevezünk.

A **bridge** olyan hálózati eszköz, amely miután megtanulta, hogy melyik portján milyen MAC című állomások vannak, egy keretet csak akkor továbbít (azon a portján keresztül, ahol a címzett található), ha az nem azon a porton érkezett, amelyiken a címzett van. A keretek továbbításakor szabályosan lejátssza a MAC protokollt, hiszen a 2. rétegben működik. A bridge tehát a hozzá kapcsolódó szegmensekből külön ütközési tartományokat képez. így a 2.10. ábrán található hálózat esetén már képes az „A" állomás a „B"-nek, a „D" pedig az „F"-nek egyidejűleg keretet küldeni. Általános esetben egy bridge-nek kettőnél több portja is lehet, ilyenkor természetesen csak arra **a** portra továbbítja a keretet, ahol a címzett van.[14]

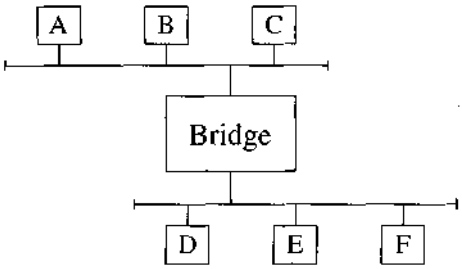
Ennél magasabb szintű, 3. rétegbeli összekapcsolással majd a 3. fejezetben foglalkozunk.

A csillag topológiájú hálózatokban pedig eleve szükség van egy aktív eszközre, ami összekapcsolja az állomásokat. Ezt megtehetjük akár az 1. rétegben a *hub* vagy a 2. rétegben a *switch* segítségével.

A **hub** a többportos repeater újabb neve a csillag topológiájú hálózatokban.

A **switch** pedig egy többportos bridge, ami külön ütközési

[14]Amíg nem tudja, hogy hol van, addig minden portra továbbítja.



2.10. ábra. Hálózati szegmensek összekapcsolása bridge segítségével

tartományokat képez. Alapvetően két módban működhet:

- *store and forward* - végigveszi a keretet és tárolja, majd a MAC protokoll szabályai szerint továbbítja
- *cut through* - elkezdi venni a keretet, majd kis késleltetéssel (a célcím megállapítása után) a MAC protokoll szabályai szerint továbbítja

Lehető'ség van egy harmadik, *adaptív módra* is, ami azt jelenti, hogy a fenti két mód közül a forgalomnak megfelelőt használják: kis forgalom esetén a cut through üzemmódot használják, majd ha az ütközések másodpercenkénti száma meghalad egy korlátot, átváltanak store and forward üzemmódba. Ha az ütközések száma lecsökken, akkor természetesen ismét cut through üzemmódba váltanak.

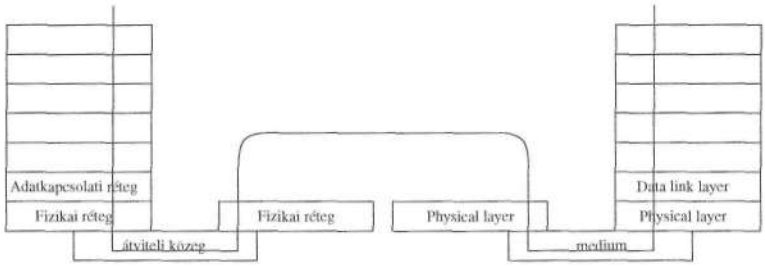
Megjegyezzük, hogy (amint fent említettük) a swicth portjai külön *collision domaint* alkotnak, de egy *broadcast domaint* képeznek, broadcast címre küldött kereteket a switch az összes portjára továbbítja.

Összefoglalásul azt mondhatjuk, hogy a szegmensek összekapcsolása történhet fizikai szinten (ekkor az összes állomás

egy ütközési tartományt alkot, 2.11. ábra) és adatkapcsolati szinten (ekkor a szegmensek külön ütközési tartományt képeznek, 2.12. ábra). A csillag topológiájú hálózatoknál mindenképpen szükséges valamilyen aktív eszköz, ezeket másképp nevezik, mint a busz topológiájú hálózatoknál. Az elnevezéseket a 2.3. táblázatban foglaltuk össze.

	busz topológia	csillag topológia
adatkapcsolati szinten	bridge	switch
fizikai szinten	repeater	hub

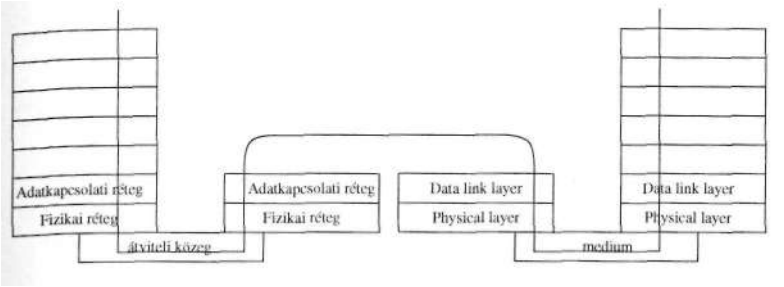
2.3. táblázat. Hálózatrészek összekapcsolására szolgáló aktív eszközök megnevezése



2.11. ábra. Hálózatrészek összekapcsolása fizikai szinten

2.1.8. Ethernet hálózatok fejlődése

Azt már említettük, hogy az Ethernet fajtái között a 10Base5 volt az első, majd a 10Base2 követte. A lOBaseT sem volt gyorsabb, a jelentősége az, hogy bevezették a csavart érpáras kábelek használatát.



2.12. ábra. Hálózatrészek összekapcsolása adatkapcsolati szinten

Az igazi előrelépést a *Fast Ethernet* jelentette, ami megtízszerezte a sebességet. Ehhez a *lOOBaseX* technológiát az FDDI-tól[15] (Fibre Distributed Data Interface) vették át. A 100 Mbit/s-os Ethernet hálózatok fejlődését a 2.13. ábrán követhetjük. A lOOBaseX technológia a csavart érpárból már a Cat5 minőséget igényli, és ezen full duplex működésre képes: ez a 100*BaseTX,* míg üvegen 100*BaseFX*-nek hívják.

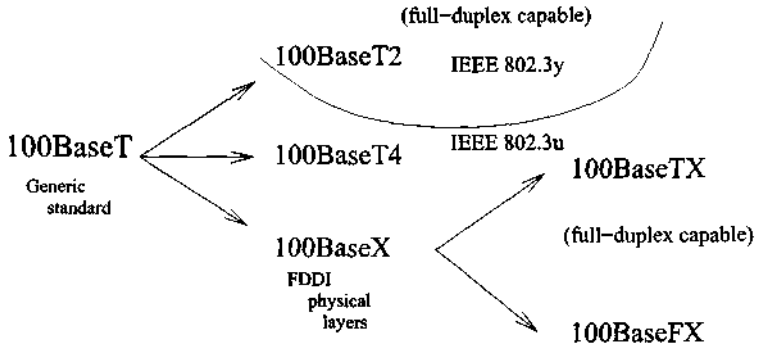
E jegyzet írásának idején[16] a használatban levő és a telepítésre kerülő Ethernet hálózatok döntő többsége Magyarországon a lOOBaseTX (Cat5/Cat5e kábelezéssel). Máshol[17] azonban már egy-két évtizeddel korábban nagy összegeket ruháztak be a Cat3 (esetleg Cat4) kábelezésbe. Annak érdekében, hogy ez a beruházás ne vesszen el, kifejlesztettek ilyen közegeken működő 100Mbit/s sebességű Ethernet hálózatokat is. Előbb a *100BaseT4-et,* amely mind a négy érpárat használja[18], ezért egyszerre csak egy irányban tud forgalmazni *(half duplex),*

[15] Az érdeklődőknek ajánlunk az FDDI-ról egy kiváló könyvet: [11].

[16] 2006.

[17] például: USA

[18] Egészen pontosan két érpár kétirányú a másik két érpár pedig egyik, illetve másik irányra van fenntartva, így irányonként három érpárat használ.

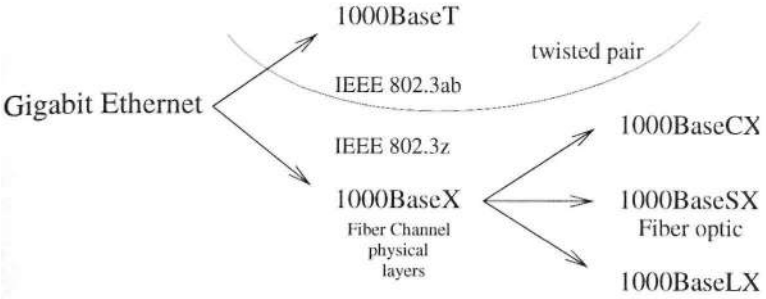


2.13. ábra. A 100Mbit/s-os Ethernet hálózatok fejlődése [6].

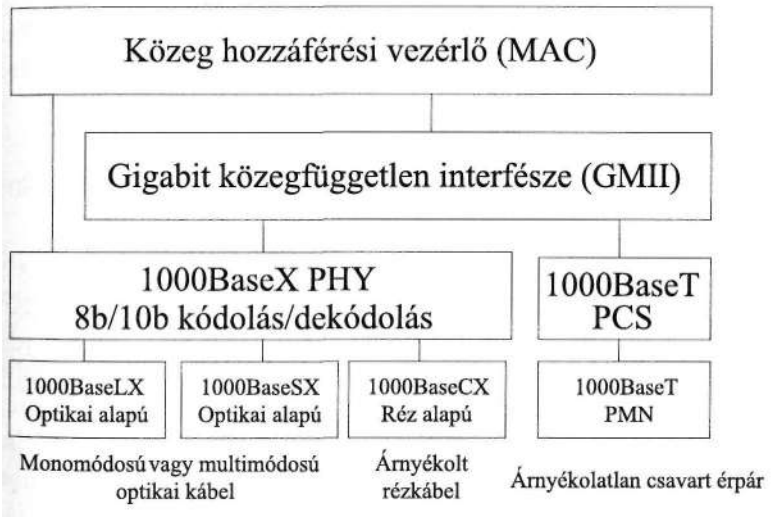
majd a *100BaseT2-t,* amely egy érpáron éri el ezt a sebességet, így *full duplex* működésre képes. Nálunk azonban ezek a rendszerek egyáltalán meg sem jelentek, nem volt miért!

A gigabites Ethernet hálózatok fejlődése két irányból merített. Egyrészt a 100BaseT2 technológiára építve kifejlesztették a Cat5 kábelen működő 1000BaseT hálózatot, másrészt a Fibre Channel technológia fizikai rétegét átvéve megalkották az üvegszálat használó 1000BaseSX és 1000BaseLX valamint az STP kábelen működő 1000BaseCX hálózatokat: 2.14. ábra.

A 2.15. ábrán követhetjük a gigabites technológia logikai felépítését. A MAC réteg (ami azonos az alacsonyabb sebességű Ethernet hálózatokéval) vezérli a közegfüggetlen gigabites interfészt (GMII - Gigabit Medium Independent Interface). Ha ez alatt a Fibre Channelből átvett megoldás van, akkor előbb egy 8b/10b kódolás történik, majd a megfelelő optikai (1000BaseLX, 1000BaseSX) vagy réz alapú (1000BaseCX) interfész következik. Egyébként pedig az 1000BaseT megoldás megfelelő rétegei találhatók.



2.14. ábra. Az 1 Gbit/s sebességű hálózatok fejlődése [6].



2.15. ábra. Gigabites technológia architektúrája [40]

Gigabites hálózatokat jelenleg még főként gerinchálózati[19]

[19] Hálózatok felépítésével a 2.2. alfejezetben foglalkozunk, ott tisztázzuk majd a gerinchálózat jelentését.

célra használnak, de már terjedőben vannak a gigabites munkaállomások is.[20]

Léteznek már 10 Gbit/s sebességű Ethernet hálózatok is. Ezekkel nem foglalkozunk.

Végül fontosnak tartjuk megemlíteni, hogy a csavart érpáras technológiák eszközei általában (de nem mindig) visszafele kompatibilisek abban az értelemben, hogy egy nagyobb sebességű eszköz képes alacsonyabb sebességen is működni. A működési módban a hálózati eszközök automatikusan megállapodnak *(auto negotíation)*. A következő sorozatból a lehető legjobb olyat választják, amit mindkét eszköz és a kábelezés is lehetővé tesz: lOOOBaseT full duplex, lOOOBaseT half duplex, 100BaseTX full duplex, 100BaseT2 full duplex, lOOBaseTX half duplex, 100BaseT2 half duplex, 100BaseT4, lOBaseT full duplex, lOBaseT half duplex.

2.1.9. Ethernet hálózatok egyes fajtáinak összefoglalása és értékelése

Összefoglaljuk az Ethernet hálózatok egyes fajtáinak legfontosabb jellemzőit. Természetesen az egyes fajtákkal jelentőségüknek megfelelően foglalkozunk. A 27. oldalon található 2.1. táblázat adatait nem ismételjük meg. (A hivatkozásra kerülő kódolások leírása a 34. oldaltól található meg.)

10Base5 és 10Base2

Vastag és vékony koaxiális kábelen működő, busz topológiájú hálózatok. A fizikai rétegben Manchester kódolást használnak. Csupán történeti jelentőségük miatt és didaktikai célból (a későbbiek megértése érdekében) érdekesek, egyébként a gyakor-

[20] Jobb minőségű PC alaplapokon az integrált hálózati illesztő már általában lOBaseT/lOOBaseTX/lOOOBaseT, de általában ezek csak lOOBaseTX-ként működnek, mivel ilyenek a telepített hálózati aktív eszközök.

latban a hallgatóink főleg úgy kerülhetnek kapcsolatba velük, hogy le kell bontaniuk őket. :-)

lOBaseT

Legalább Cat3 csavart érpáras kábelen működő csillag topológiájú hálózat. A gyakorlatban főleg úgy találkozunk ilyen eszközökkel, hogy a Cat5 kábelezésű hálózatainkon található még néhány ilyen eszköz is. Mivel a lOOBaseTX aktív eszközei általában mind 10 mind 100Mbit/s működésre képesek (dual speed), ezért legtöbbször nem okoznak problémát. Gondot akkor okozhat, ha valamely aktív eszköz csak 100Mbit/s működésre képes.[21]

lOOBaseTX

Legalább Cat5 minőségű csavart érpáras kábelen működő csillag topológiájú hálózat. A fizikai rétegben először 4b/5b kódolást használ, az eredményt pedig NRZI kódolással adja ki a vonalra. Működhet half duplex üzemmódban hub használatával is, de ma már általában full duplex üzemmódban switch használatával működtetjük a hálózatainkat. Hallgatóink a tárgy gyakorlatain alaposan megismerhetik, mert jelenleg ez a legelterjedtebb hálózat. Figyeljük meg, hogy a 4b/5b és NRZI kódolás miatt a *jelzési sebesség* 125Mbaud!

Természetesen a probléma nem megoldhatatlan: PC esetén ilyenkor általában olcsón lOOBaseTX-re cserélhetjük a lOBaseT hálózati illesztőkártyát. Vannak olyan eszközök, műszerek, ahol ez nem tehető meg, ekkor pedig keresnünk kell egy dual speed aktív eszközt, amit egyrészt összekötünk az üzemszerűen használt, csak 100 Mbit/s-os működésre képes aktív eszközünkkel, másrészt rákötjük a 10 Mbit/s-os eszközöket.

lOOBaseFX

A lOOBaseTX üvegszálas párja. Jelentó'ségét főleg a nagyobb áthidalható távolság adja. Használatát ezenkívül villámvédelmi, speciális körülmények között érintésvédelmi szempontok indokolhatják. Új gerinchálózat építése esetén valószínűleg inkább gigabites optikai kapcsolatot építünk.

100BaseT4

8b6t kódolást használ, majd a kódolt információt az irányonkénti 3 fizikai csatornán meglehetősen bonyolult módon viszi át. így 100Mbit/s sebességű half duplex átvitelt tesz lehetővé legalább Cat3 minőségű sodrott érpáras hálózaton.

Magyarországon nincs gyakorlati jelentősége.

100BaseT2

Ötszintű impulzus-amplitúdó moduláció (PAM-5) használatával 100Mbit/s sebességű full duplex átvitelt tesz lehetővé legalább Cat3 minőségű sodrott érpáras hálózaton.

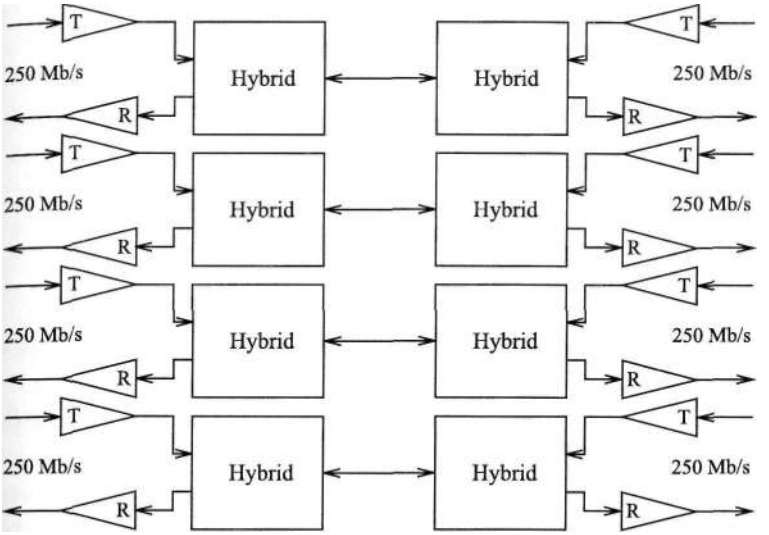
Magyarországon nincs gyakorlati jelentősége.

lOOOBaseT

A 100BaseT2-höz hasonlóan az 1000BaseT is PAM-5 kódolást alkalmaz, viszont a 100BaseTX-hez hasonlóan Cat5/Cat5e kábelen 125 Mbaud jelzési sebességgel működik, de a 4 érpár mindegyikét egyidejűleg használva.

A rendszer azért képes mégis full duplex működésre, mert a szuperpozíció elvét kihasználva az állomások a vett jelből kivonják a saját maguk adását, és így megkapják, hogy mit küldött a másik állomás, lásd 2.16. ábra.

Az 1000BaseT megoldás jelentősége azért igen nagy, mert az elterjedten használt Cat5/Cat5e kábelezésen képes gigabi-



2.16. ábra. Full duplex átvitel megoldása [40]

tes sebességet nyújtani. így kiválóan alkalmas akár gerinchálózatnak, akár szerverek csatlakoztatására, akár - igény esetén - egyedi munkaállomások számára nagy sebességű hálózati kapcsolatnak.

lOOOBaseSX

A Fibre Channel technológiát vette át. Többszörös módusú üvegszálon, rövid hullámhosszon (850 nm) működik. A fizikai rétegben 8b/10b kódolást használ.

Kiválóan alkalmas gerinchálózati célokra. (Az ára miatt asztali gépekhez nem használjuk.)

lOOOBaseLX

Az lOOOBaseSX-hez képest annyi a különbség, hogy a hullámhossza 1300 nm és egyszeres módusú üvegszálon az áthidalható távolság lényegesen nagyobb. (Lásd a pontos adatokat a 27. oldalon található 2.1. táblázatban.)

lOOOBaseCX

Bár a Fibre Channelbó'l ezt is átvették és a szabványnak része, a gyakorlatban az lOOOBaseT-vel szemben meglehetősen esélytelen a negyedakkora áthidalható távolság miatt, ráadásul árnyékolt technológiát követel.

2.2. Strukturált kábelezés - a passzív hálózat[22]

Ez nem egy másik hálózattípus, hanem csupán egy kábelezési szabvány, azért tárgyaljuk külön alfejezetben és nem az Ethernettel együtt, mert másra is használható. Éppen az a lényege, hogy *egy végpontról nem kell előre eldöntenünk, hogy mire használjuk.* Nem készítünk külön telefonhálózatot és számítógéphálózatot, hanem csak egyetlen strukturált hálózatot. Ugyanaz a csavart érpár mindkét célra alkalmas lehet, csak a bekötéskor kell döntenünk, és igény szerint később a célját meg is változtathatjuk.

A strukturált kábelezési rendszer passzív részei:

- Főrendező (számítógép-hálózati, illetve telefonos rendező együtt)

[22] Kárpáti László, a PrimusNet Kft. munkatársa elő'adásanyagának [38] felhasználásával készült.

- Gerinckábelezés (vertikális kábelezés, újabban optikai közegen)
- Alrendezők (vízszintes kábelezés elosztóközpontjai)
- Vízszintes kábelezés (rézalapú sodrott érpáras kábelezés)
- Fali csatlakozók, padló dobozok

Egy strukturált kábelezéssel készített számítógép- és telefonhálózat látható a 2.17 ábrán.

A rendezőszekrényekben *patch paneleken* végződtetik a kábeleket. A rendszer részeit hierarchikus rendszerben számozzák, így például az „R1 2/15" az 1. sz. rendezőszekrény 2. sz. patch panelének 15. sz. végpontját jelenti. Itt találhatók az aktív eszközök is, amelyekbe *patch kábelek* segítségével kötik be a kívánt végpontokat. Ugyancsak patch kábel segítségével kötik be a fali csatlakozóra (vagy padlódobozba) a számítógépeket: 2.18 ábra.

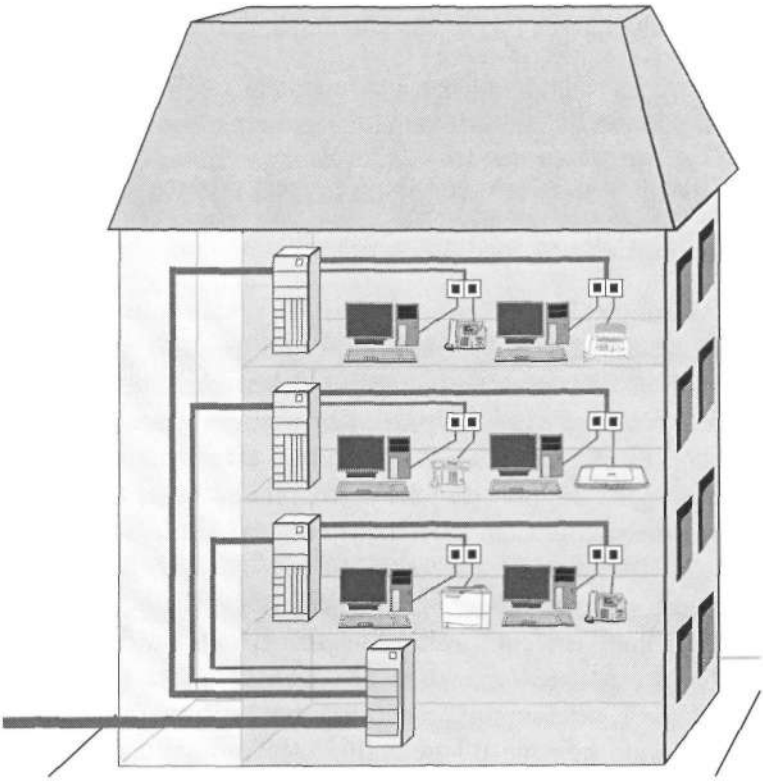
Fontos, hogy a strukturált kábelezésnél minden csatlakozó RJ-45-ös, így ha analóg távbeszélő készüléket szeretnénk bekötni, akkor a telefonzsinór egyik végére is ilyen csatlakozót kell szerelnünk, mert az RJ-ll-es dugó tönkreteszi az RJ-45-ös végpontot! (IP telefon esetén ilyen probléma nem merül fel.)

Egy rendszer tervezésekor fontos a megfelelő kábel kiválasztása. Ez egyrészt jelenti a kábel típust/fajtát, tehát hogy a csavart érpár milyen árnyékolással rendelkezzen, másrészt a kábel kategóriát.

Ami a kábel kategóriát illeti, Cat5/Cat5e-nél gyengébb kábelt sehol sem használunk. Ennél jobb minőségű kábel használatát a beruházás időállóságával szokták indokolni, ami az adott esetben megfontolás tárgyát képezi.

Az árnyékolás tekintetében országonként eltérő a hozzáállás. Néhány példa:

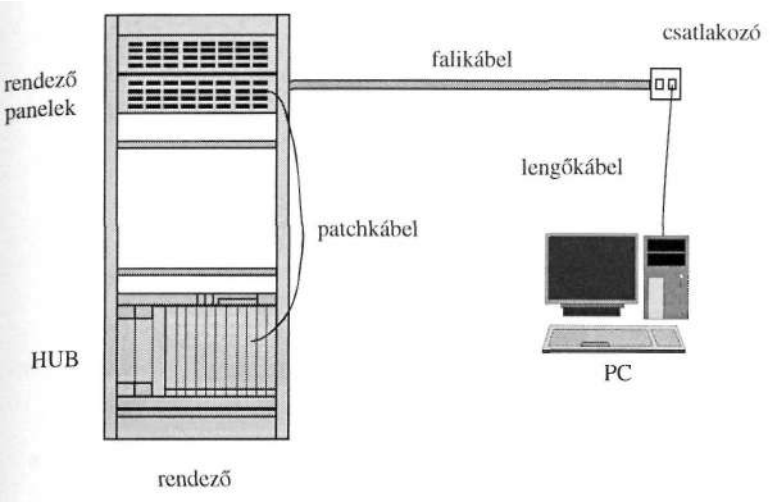
- USA: elég az UTP, mert az megfelel.



2.17. ábra. Strukturált kábelezés

- Németország: legyen árnyékolt, biztos ami biztos.
- Magyarország: UTP, mert az olcsóbb (de árnyékolt, ha német érdekeltségű a cég)

Műszakilag alapesetben teljesen megfelelő az UTP, viszont sok elektromos zajjal terhelt (például ipari) környezetben, illetve olyan környezetben, ahol éppen a hálózat elektromágneses sugárzása okozhat gondot, megfontolandó valamilyen árnyékolt



2.18. ábra. Elemek a strukturált hálózatban

megoldás választása.

Ha az árnyékolás mellett döntünk, akkor annak úgy van értelme, ha az egész rendszerre kiterjed, tehát falkábel, csatlakozók, patchkábelek. Fontos, hogy az árnyékoló rendszert földelni kell!

2.2.1. Tervezési szabályok

Egy strukturált hálózat megtervezéséhez a tárgy kereteit meghaladó ismeret és jártasság szükséges, az alábbiakban csak tájékoztató jelleggel közöljük a Krone rendszer tervezési szabályainak kivonatát.

Csatlakozók száma:

- Fali csatlakozó:
  * 1 munkahely / 10 négyzetméter

* 2 csatlakozó / munkahely (telefon + LAN)
* 2 tartalék csatlakozó / szoba (vagy: +10%)
  például: iroda 2000 négyzetméter = 220 dupla csatlakozó

- Padlódoboz:
  * 1 munkahely / 10 négyzetméter
  * 2 csatlakozó / munkahely (telefon + LAN)
  * 2 tartalék csatlakozó / szoba (vagy: +10%)
    A végpontoknak 1/3-a nem hozzáférhető az asztalok és szekrények miatt!

• Port szám és a rendezó'k szükséges mérete:

- Általában 24 és 32 portos RJ 45-ös felületű *patch panelt* használunk. Leginkább 24 portosat, mert akkor sokkal kezelhetőbb a szekrény.
- A rendezőszekrények magassága a kiszolgálandó végpontok számától függ, az egysége a *rack unit*[23]. Minden két patch panel után egy *kábelterelőt* (gyűrűs panel) kell tervezni a kezelhetőség érdekében.
- A szükséges aktív eszközöket, szünetmentes tápegységeket, villamos-hálózati csatlakozókat, esetleg szerverek helyigényét is be kell tervezni a rendező szekrények magasságába. Ügyeljünk a szükséges hűtés betervezésére is! Szükséges lehet a hűtő levegő szűrése, a por ugyanis tönkreteheti az aktív eszközöket!
- Fontos a szekrény szélessége is, lehetőleg 800 mm széleset tervezzünk, hogy legyen hely a patch kábelek függőleges elvezetésére!

• Kábelhosszak:

[23] 1 rack unit (rövidítve 1U) = 44,45 mm (1.75 inch)

- Szigorú szabály EIA/TIA 568 és ISO/IEC 118021: Minden link (patch paneltől az aljzatig) kevesebb legyen mint 90 m!

— Egy végpont kábelezésének a hossza =
az épület szintjeinek belmagassága +
a rendező és a gerinc közötti nyomvonal hossza +
a kábel gerincen futásának hossza +
a szoba hossza +
a szobában való ráhagyás +
1,5 m ráhagyás a bekötésnél

Patch kábel hossza:

— Szigorú szabály EIA/TIA 568 és ISO/IEC 118021: Minden channel (switch - patch kábel - patch panel - falikábel - aljzat - patch kábel - számítógép) kevesebb kell legyen, mint 100 méter.

— A rendező oldali patch kábel minimum 1 m, a számítógép lengőkábele maximum 9 m[24].

Struktúra meghatározása

— Mindig kellő biztonsági távolsággal tervezzünk

— Megfelelő helyekre tervezzük a nyomvonalakat

— Kellő távolság az erősáramú hálózattól

— Csillag topológiával tervezzünk

[24] Vegyük észre, hogy a legfeljebb 100 méter szegmenshosszból legfeljebb 10 m lehet patch kábel és legfeljebb 90 m falikábel. Ez azért van, mert a falkábel és a patch kábel különböző felépítésű, a patch kábel hajlékonyabb, de kedvezőtlenebbek a villamos jellemzői.

2.3. Vezetéknélküli helyi hálózatok

Ez az alfejezet egykori hallgatóm, Lugosi Zoltán szakdolgozatának [39] felhasználásával készült. A téma iránt mélyebben érdeklődőknek ajánljuk: [5]

2.3.1. Vezetéknélküli helyi hálózatok alapvető kérdései

A következő kérdésekkel fogunk foglalkozni: Milyen vezetéknélküli átviteli megoldások léteznek? Milyen problémák adódnak a vezetéknélküli átvitelből és hogyan tudunk ellenük védekezni? Milyen frekvenciasávokat és milyen modulációs megoldásokat használhatunk?

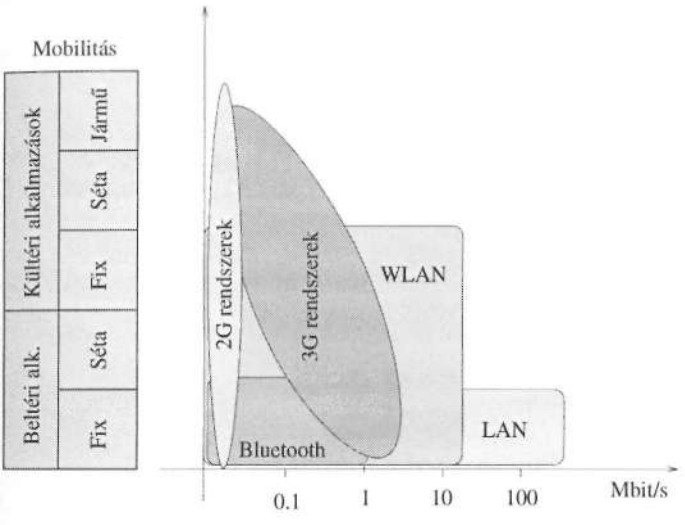
Vezetéknélküli átviteli megoldások

Vezetéknélküli számítógép-hálózathoz szükséges átvitelre a következő megoldások léteznek:

- Optikai úton:
  - Infra
  - Laser
  - Bluetooth egyes változatai
- Rádiócsatornán keresztüli
  - Bluetooth
  - HiperLAN/2
  - IEEE 802.11 és változatai
  - IEEE 802.16 (hivatalosan Wireless MAN)
  - PAN megoldások, például: IEEE 802.15
  - (GSM adatcsatorna, GPRS, EDGE, HSDPA)

Az optikai megoldásokkal egyáltalán nem foglalkozunk, a WLAN-ok közül is csak néhánnyal.

Mobilitás szempontjából a 2.19. ábrán látható módon csoportosíthatjuk a jellegzetesebb megoldásokat.



2.19. ábra. Egyes hálózati megoldások mobilitás és átviteli sebesség jellemzői

A rádiós átvitel problémái

A rádiós hálózatok működését negatívan befolyásoló tényezők a következők:

Fading Többutas terjedésnél ellentétes fázisban érkezhet meg a direkt és a visszavert hullám, amik igen erősen gyengítik egymást. Lásd a 2.20. ábrán.

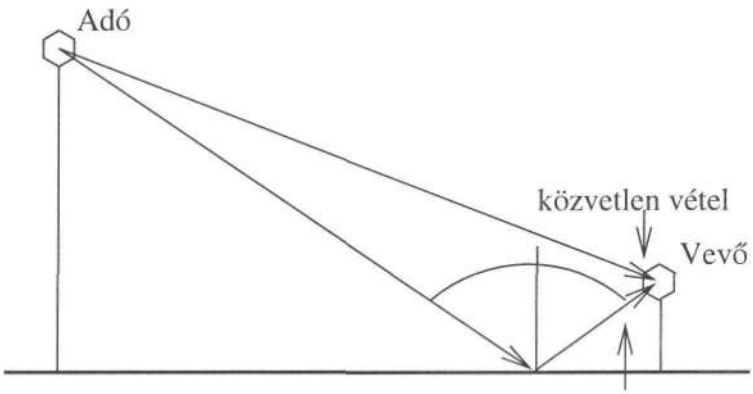
Védekezés: a rádiós átvitelnél kezeljük, például: térbeli *diverziti* (diversity) vétel (több vevő használata eltérő elhelyezkedésű antennákkal).

Zaj Nem a rendszerből származó (véletlenszerű) villamos jel.
Védekezés: hibajavító kódolással.

Interferencia (vagy ütközés) más állomás adásával
Védekezés: szórt spektrumú megoldások

Rálátás hiánya Az adó és a vevő között a Fresnel-zóna [36] nem (teljesen) üres.

Védekezés: visszavert jel használata "feljavítással": szórt spektrum



2.20. ábra. Fading kialakulása

Felhasználható **frekvenciasávok**

Két alkalmazott frekvenciasáv létezik:

- ISM (Industrial, Scientific and Medical)
  - 2,4-2,4835 GHz/14 előre kijelölt frekvencia
  - Földrajzi régiók szerint különbözhet.

    (Európa, USA, Japán, Franciaország, Spanyolország)
- UNII (Unlicenced National Information Infrastructure)
  - 5,150-5,250 GHz/12 előre kijelölt frekvencia
  - Földrajzi régiók szerint különbözhet.
    (Európa, USA, Japán, Franciaország, Spanyolország)

Szórt spektrumú modulációs eljárások

Három szórt spektrumú megoldás létezik:

- DSSS (Direct Sequence Spread Spectrum)
- FHSS (Frequency Hopping Spread Spectrum)
- CDMA (Code Division Multiple Access)

Mindhárom megoldásnál a jel spektrumát mintegy „szétkenik". Az átviendő jel spektrumát valamilyen transzformációval az eredetinek több tízszeresére kiszélesítik, és kisebb teljesítménysűrűséggel viszik át. Az általunk tárgyalt rendszerek a DSSS és az FHSS mellett nem a CDMA-t, hanem az OFDM-et (Orthogonal Frequency Division Multiplexing) használják még, ami nem szórt spektrumú megoldás.

DSSS (Direct Sequence Spread Spectrum) esetén *chipek*[25] megfelelő sorozatával kódoljuk az egyest és a nullát. A 2.21. ábrán láthatunk egy példát. Ha néhány chip invertálódik is

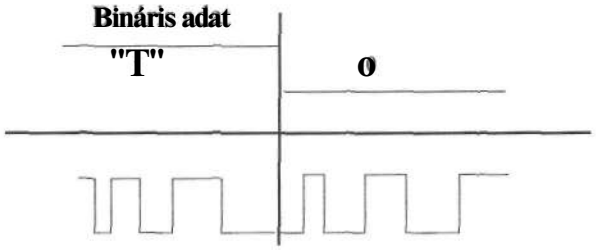
[25] a bitnél kisebb adategység

valamilyen zaj miatt, nagy valószínűséggel még felismerhető lesz a bit.

Az "l"-es kódolása:

+1-1+1+1-1-1+1+1+1-1 -l-l

A "0" kódolása ennek az inverze.

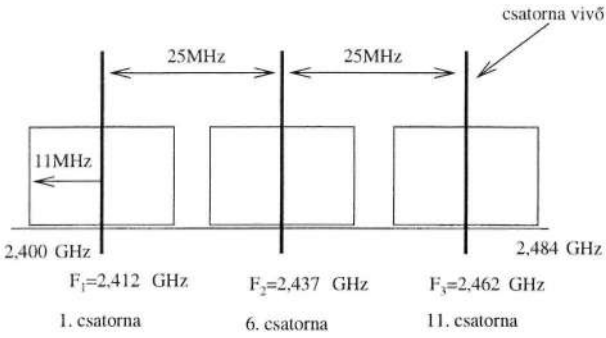


2.21. ábra. Direct Sequence Spread Spectrum

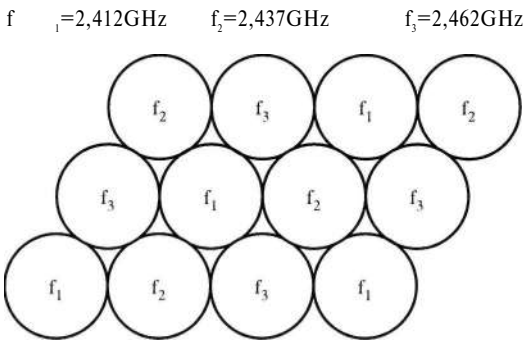
Ezzel a megoldással jócskán megnőtt a szükséges sávszélesség. Három vivő fér bele oldalsávjaival (22 MHz) a rendelkezésre álló tartományba: 2.22. ábra.

Három különböző frekvenciával már lefedhető úgy a sík, hogy a szomszédosak minden oldalról különböznek, így elkerülhető **a** zavartatás, az interferencia minimális lesz: 2.23. ábra.

FHSS (Frequency Hopping Spread Spectrum) rendszerben a frekvenciasávban 75db vivőfrekvenciát definiálunk. Az adó egy ál véletlen generátorral választja ki közülük, hogy melyiken adjon. (2.24. ábra). Természetesen a vevő ugyanazt az álvéletlen generátort használja, és azonos értékről



2.22. ábra. Vivőfrekvenciák és oldalsávjaik.

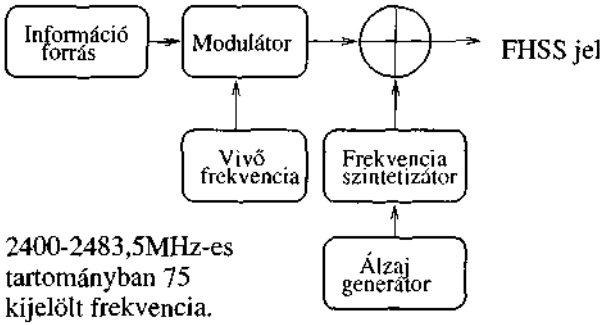


2.23. ábra. Frekvencia újrahasznosítás (DSSS)

indulnak.[26] Egy adott területen egyszerre adni kívánó állomások nagy valószínűséggel eltérő frekvenciákat válasz-

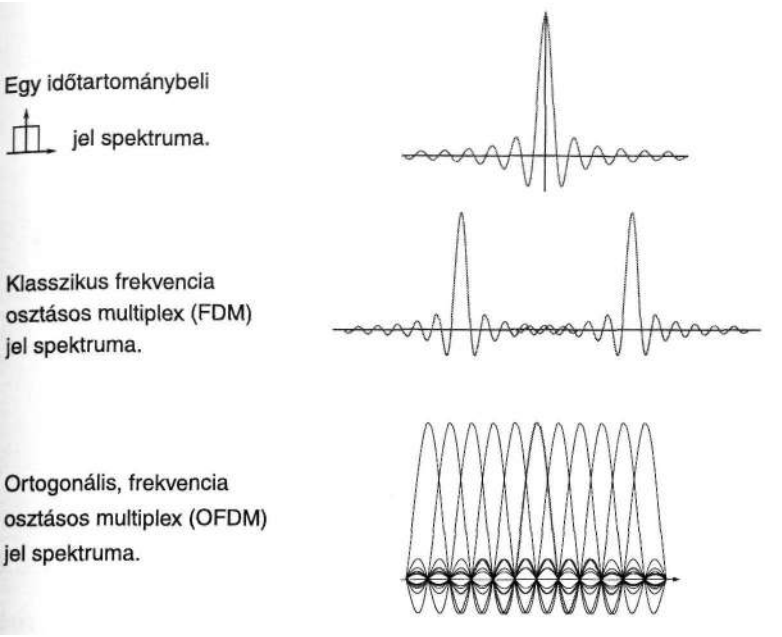
[26] Ha az álvéletlen generátor egy kriptográfiailag erős véletlenszám generátor, azaz az éppen használt frekvenciából nem lehet előrejelezni a következőt, akkor a megoldás katonai alkalmazásokhoz is kiváló, hiszen az eredményes zavaráshoz a teljes sávban kell zavaró jelet adni, ami lényegesen nagyobb energiát igényel, mint a kommunikációhoz használt adás.

tanak (feltéve, hogy nincsenek túl sokan).



2.24. ábra. Frequency Hopping Spread Spectrum

OFDM (Orthogonal Frequency Division Multiplexing) esetén a 2.25. ábrán bemutatott megoldást alkalmazzuk. Legfelül egy négyszög jel és a spektruma látható. Alatta található - viszonyítási alapként - egy klasszikus frekvencia osztásos multiplex (FDM - Frequency Division Multiplex) jel spektruma. Látható, hogy a szomszédos csatornák vivőfrekvenciái csak annyira közelíthetők egymáshoz, hogy az átlapolódás energiatartalma olyan kicsi legyen, hogy a dekódolás hibamentesen elvégezhető legyen. Az ábra harmadik része egy ortogonális frekvencia osztásos multiplex (OFDM) jel frekvenciatartománybeli „összerakását" mutatja. Miért „tolhatók egymásba" ennyire az egyes csatornák vivőfrekvenciái? Az eljárás alapja az *ortogonális frekvenciák* használata. Ortogonálisak azok a frekvenciák, amelyekre fennáll az $F_n = (F_0 + n/T)$ kapcsolat. Tekintsük ezeknek a frekvenciáknak a spektrumát: 2.26. ábra. Az egyes alvivők középfrekvenciáján a többi jel spektruma mindig 0 értéket vesz fel. Ez a magyarázata annak, hogy miért „tolhatók egymásba" a klasszikus FDM-hez képest



2.25. ábra. Orthogonal Frequency Division Multiplexing

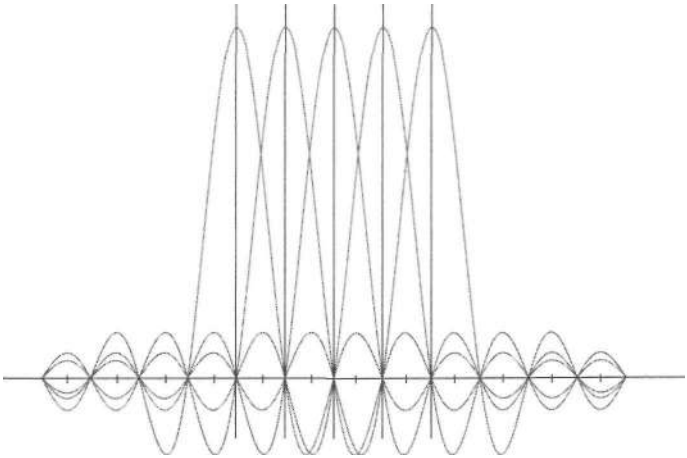
sokkal sűrűbben az OFDM vivőfrekvenciái.

Az OFDM előnyei:

- jobb spektrum kihasználás
- külső zavarok elleni hatásosabb védelem
- közvetlen rálátást nem igénylő (non-line-of-sight) működés

2.3.2. Vezetéknélküli hálózati termékek fontosabb jellemzői

Ismerjünk meg néhány vezetéknélküli hálózati megoldás fontosabb jellemzőit!



2.26. ábra. Ortogonális alvivők spektruma

- Bluetooth
  - Ez egy PAN megoldás
  - Bluetooth Special Interest Group fejlesztette
  - Alacsony fogyasztású eszközök
  - Maximum 8 eszköz egy hálózatban
  - Rövid távolságokra (< 10 méter)
  - Számítógép és kézi eszközök között
  - Alacsony sebességű (1 Mbps)
  - ISM frekvenciatartományban működik
  - Pont-pont, pont-multipont összeköttetés
- HiperLAN/2 (High Performance Radio LAN)
  - European Telecommunications Standards Institute (ETSI) fejlesztése

- Áthidalható távolság 100 méter
- Átviteli sebesség 54 Mbps
- Mind az ISM, mind az UNII frekvenciasávban
- OFDM modulációs mód

• IEEE 802.11.x változatok fő'bb paraméterei

- IEEE 802.11: 2 Mbps, 300 m, ISM, DSSS
- IEEE 802.11b: 11 Mbps, 100 m, ISM, DSSS
- IEEE 802.11a: 54 Mbps, 200 m, UNII, OFDM
- IEEE 802.llg: 54 Mbps, 200 m, ISM, OFDM, kompatibilis az IEEE 802.11b-vei

2.3.3. Az IEEE 802.11 szabványcsalád

Ezekkel egy kicsit részletesebben is megismerkedünk.

2,4 GHz-es WLAN szabványok

• IEEE 802.11

— DSSS vagy FHSS moduláció

— 1-2-3 Mbit/s bruttó átviteli sebesség

• IEEE 802.11b (Wi-Fi) High Speed Wireless LAN

— DSSS moduláció

— 11 Mbit/s bruttó átviteli sebesség

• IEEE 802.llg Very high speed Wireless LAN

— OFDM moduláció
— 54 Mbit/s bruttó átviteli sebesség

5 GHz-es WLAN szabvány

- IEEE 802.11a High Speed Wireless LAN
  - OFDM (64 FFT - 54 alvivő)
  - 54 Mbit/s bruttó átviteli sebesség

3. fejezet

Internet protokollkészlet

Először is tisztázzunk néhány fogalmat! Az *internet* jelentése: összekapcsolt hálózat. Az *Internet* egyetlen világméretű nyilvános IP alapú internet. Az *intranet* internet technológiát használó intézményi belső hálózat. Az *internet protokollkészletbe* (internet protocol stack) tartozó legfontosabb protokollok:

IP (Internet Protocol, magyarul: internet protokoll[1])
: Az IP egy nem megbízható *datagram*[2] szolgálatot biztosít forrás és célgép között, függetlenül attól, hogy ezek hol helyezkednek el (azonos, szomszédos, vagy egymástól távoli hálózatban).

TCP (Transmission Control Protocol)
: A TCP az IP-t felhasználva két végpont közötti megbízható kétirányú bájtfolyam átvitelt biztosít. Ehhez a két végpont között kapcsolatot épít fel, amit az átvitel végén le kell bontani.

UDP (User Datagram Protocol)
: Az UDP szintén az IP-re építve összeköttetés nélküli (itt

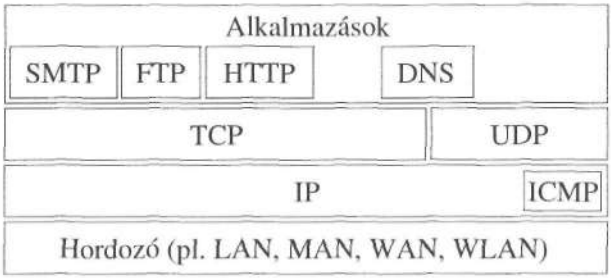
[1] Figyeljünk oda az eltérő helyesírásra!

[2] A csomag speciális neve az internet protokollkészletben.

nincs kapcsolat felépítés, bontás) végpontok közti nem megbízható datagram szolgáltatást nyújt a felhasználóknak.

ICMP (Internet Control Message Protocol)
Az ICMP az IP szolgálati közleményeit hordozza (szintén az IP-re építve) két IP-t használó állomás között.

A 3.1. ábrán megfigyelhetjük a felsorolt protokollok elhelyezkedését a TCP/IP modellben. A hordozó rétegre épül az IP, erre pedig a TCP és az UDP. Az ICMP-t az általunk használt „egysíkos" számítástechnikai architektúrában az IP réteg részének tekintjük.[3] Először ezzel a négy protokollal ismerkedünk meg.



3.1. ábra. A TCP/IP protokollcsalád

Az internet protokollkészlet elemeinek (és sok más számítástechnikával, hálózattal kapcsolatos protokollnak) a definícióját az *RFC-kben* találjuk meg. Az RFC a *Request For Comments* rövidítése. Ezek a dokumentumok elérhetők például a http://rfc.net webcímen. Az egyes protokolloknál a továbbiakban mindig megadjuk az azt eredetileg definiáló RFC számát

[3]A távközlési architektúrában pedig egy külön úgynevezett *vezérlő* (control) síkon helyezkedik el.

is.

Ajánlott irodalom: Mivel az RFC-k nem túl olvasmányosak, az internet protokollkészlethez ajánlunk néhány könyvet. Elsőként Comer könyvének első kötetét [4] ajánljuk, ami alapján e fejezet is készült. Szintén jól használható mű [19]. Nem olvasmányos, mivel kézikönyv [14]. Magyar nyelvű és újabb könyv: [18].

3.1. Internet Protocol

Az IP protokollt eredetileg az RFC 791-ben definiálták.

3.1.1. Az IP címek

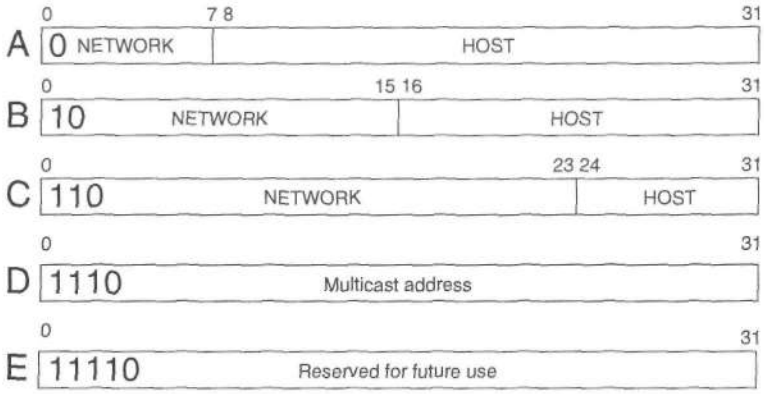
Az IP címek felépítése

Az IP-nek két, bárhol elhelyezkedő számítógép között képesnek kell lennie adatokat szállítani. Ehhez mindenek előtt megfelelő címzésre van szüksége. Amikor egy levelet postán szeretnénk elküldeni valakinek, akkor meg kell adnunk, hogy a címzett mely ország mely városában, ott milyen utcában és hányas szám alatt lakik. A címzés tehát hierarchikus.

Az IP 32 bitet használ egy állomás megcímzésére. A hierarchikus címzés két szinten valósul meg. A cím első része a *hálózatcím* (network address) kiválasztja azt a hálózatot, amelyikre az állomás csatlakozik, a második része a *gépcím* (host address) pedig a cím első része által kiválasztott hálózaton belül azonosítja a gépet.

A protokoll tervezői úgy döntöttek, hogy több lehetőséget adnak arra, hogy hol legyen a két rész határa a 32 biten belül. Annak ismeretében alakították ki az *IP cím osztályokat,* hogy nagy szervezetből kevés van, közepesből jóval több, kicsiből pedig kifejezetten sok. A cím első néhány bitje meghatározza, hogy az IP cím melyik osztályba tartozik (A, B, C, D,

E). A 3.2. ábra megmutatja, hogy melyik osztályban mekkora rész jut a hálózat, illetve a gép címének megadására.[4]



3.2. ábra. Az IP címek felépítése

Speciális jelentése van, ha egy IP cím host részében minden bit 0 értékű: ez az egész hálózatot jelentő *network address;* hasonlóan annak is, ha a host részben minden bit 1 értékű: ez az adott IP hálózatba tartozó összes gépet jelentő *broadcast address.*

Az IP címek *kanonikus* (egyezményes, szabványos) írásmódja a következő: A 32 bites címeket 4 db *oktettre*[5] bontjuk, majd ezeket tízes számrendszerben, egymástól ponttal elválasztva írjuk le. Például: 193.224.128.1. Ez a cím binárisan 110-val kezdődik, mert a 193 kettes számrendszerben: 11000001. Tehát ez a cím „C" osztályú. A hálózatcím 3 oktett, a gépcím 1 oktett.[6]

[4] Az osztályokon alapuló *classful addressing* ma már részben csak történeti jelentőségű, lásd: 4.1.6.

[5] A bájt megnevezése a TCP/IP protokollcsalád fogalomtárában. Hangsúlyozottan 8 bitet jelent.

[6] Vegyük észre, hogy milyen kényelmes az, hogy az összes IP cím osztályban a két rész határa mindig oktett határra esik!

Az IP hálózatcímek kiosztása

Az **IP** hálózatcímek kiosztását legfelső szinten az *Internet Assigned Number Authority* (IANA) [27] végzi. Érdeklődők bővebben olvashatnak a témáról az alábbi weboldalon: [23]

A fenti lapról elérhető, jelenleg kiosztott tartományok jelölésének megértéséhez szükséges a 4.1.6-ban tárgyalt CIDR jelölés ismerete.

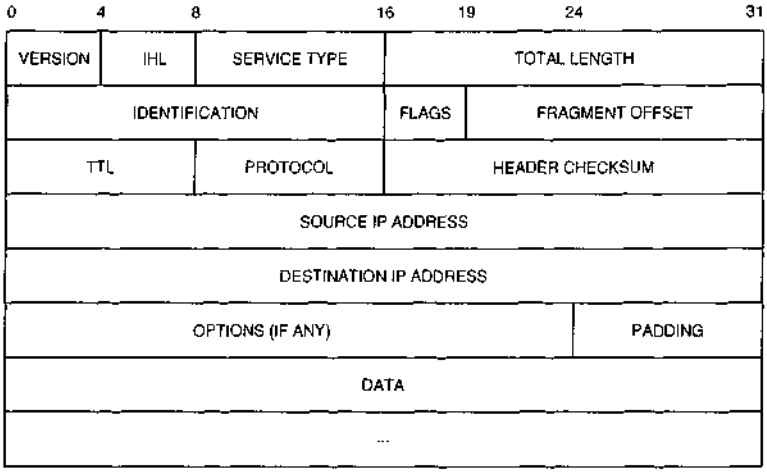
Egyelőre annyit említünk meg, hogy vannak *privát,* más szóval *nem publikus* IP cím tartományok, amelyeket bárki használhat, anélkül, hogy igényelné, de ezek használatával csak a saját intézményén belül tud kommunikálni, a világ többi része felé nem. (A címfordítástól[7] eltekintve, de ahhoz is kell egy darab publikus IP cím.) Ezekből az érdeklődő hallgatóink bátran oszthatnak a saját otthoni számítógépeik számára IP címeket, ha hálózatba szeretnék kötni őket. Az RFC 1918 szerint ilyenek:

- a 10.0.0.0 „A" osztályú hálózat
- a 172.16.0.0-tól 172.31.0.0-ig terjedő 16 darab „B" osztályú hálózat
- és a 192.168.0.0-tól 192.168.255.0-ig terjedő 256 db „C" osztályú hálózat

3.1.2. Az IP datagramok felépítése és használata

A 5. oldalon található 1.2. ábrán elviekben már láttuk, hogyan történik az elküldendő információ beágyazása és kibontása. A protokoll adategység (PDU) két részből áll, a protokoll vezérlő információból (PCI) és a szolgáltatás adategységéből (SDU). A következőkben az IP datagram felépítésével fogunk részletesen megismerkedni.

[7] NAT: Network Address Translation - e jegyzet keretében bővebben nem tudunk vele foglalkozni.



3.3. ábra. IP datagram felépítése

A 3.3. ábrán feltüntettük egy IP datagram összes lehetséges mezőjét. Az egyes mezők nevét rövidítettük, ezeknek megadjuk a teljes nevét, valamint megadjuk az egyes mezők funkcióját is, közben lépésről lépésre megismerkedünk az internet protokoll működésével.

Version verzió szám
Értéke: 0100, mivel az IP 4-es verziójáról van szó.
(A későbbiekben megismerendő IPv6 esetén értéke: 0110)

IHL (Internet Header Length) fejrész hossza
1 egység = 32 bit (4 oktett), ezért a fejrész oktettekben mért hosszának oszthatónak kell lennie 4-gyel. Tipikus értéke az 5; ha vannak opciók a fejrész végén, akkor lehet 6 vagy több.

Type of Service szolgálat típusa

Ez a mező további almezőkre bomlik, ráadásul az értelmezése változott, ezért külön foglalkozunk vele.

Total Length a datagram teljes hossza

Identification azonosítás

Szétdarabolt keretek esetén azonosítja az összetartozó töredékeket. (A tördeléssel is külön foglalkozunk.)

Flags jelzőbitek

Három bitből áll, ezek rendre:

0 (reserved) kihasználatlan
Értéke kötelezően 0.

DF (Do not Fragment) ne tördeld
0: szabad tördelni
1: nem szabad tördelni

MF (More Fragments) van még töredék
0: ez az utolsó töredék
1: ez még nem az utolsó töredék

Fragment Offset töredék eltolás

Tördelés esetén megadja, hogy az adott töredék adatrészében a kezdő bájt hányadik volt az eredeti datagram adatrészében. Mivel a mérete 13 bit, ezért 8 oktettes egységekben értendő!

Time to Live élettartam

Legfeljebb 255 lehet az értéke, minden csomópontnál csökkentik az értékét a várakozással eltöltött másodpercek számával, de legalább 1-gyel. (A gyakorlatban 1-gyel.) Ha 0-ra csökken az értéke, a csomagot eldobják. Ennek a célja az, hogy ha esetleg az útvonalválasztás hibája miatt egy csomag „eltéved", ne maradjon korlátlanul hosszú ideig a hálózatban (erőforrás pazarlás).

Protocol protokoll
Megadja, hogy az IP felett milyen protokoll helyezkedik el. (pl. TCP, UDP, ICMP)

Header Checksum a fejrész ellenőrző összege

Source Address forrás IP címe

Destination Address cél **IP** címe

Options opciók
Szerepelhetnek egyéb beállítások is, de nem feltétlenül vannak. Ha vannak, a méretük nem feltétlenül a 4 oktett egész számú többszöröse.

Padding helykitöltés
Mérete 1, 2 vagy 3 oktett lehet. Azért szükséges, mert az „IHL" egysége 32 bit, így a fejrész méretét mindig ki kell egészíteni annak többszörösére.

Data adatok
Itt található beágyazva az IP fölötti protokoll adategysége.

Most visszatérünk a Type of Service mezőre, utána pedig a datagramok tördelésével fogunk foglalkozni.

A Type of Service mező régi jelentése

Már az **IP** protokoll tervezésekor is gondoltak arra, hogy attól függően, hogy mit szállít egy datagram, más-más elbánásra lehet szüksége. A Type of Service mezőt arra tervezték, hogy ezt támogassa. Az eredeti, RFC 791-ben leírt mezőit látjuk a 3.4. ábrán.

Ezek értelmezése a következő:

3.4. ábra. A „Type of Service" mező almezői (RFC 791 szerint)

- Az első három bit a prioritás (elsőbbség) megadására szolgál.
- A D, T és R bitek közül legfeljebb két darab értéke lehet 1-es. Amelyik értéke 1-es, az útvonalválasztók a datagram továbbítása során annak a jellemzőnek az értékét igyekeznek optimalizálni, de garanciát nem vállalnak. Úgy nevezik ezt, hogy a hálózat „best effort" jellegű.

  D (Delay) 1-es értéke azt jelenti, hogy a késleltetés minél kisebb értéke a fontos számunkra.

  T (Throughput) az időegységenkénti minél több adat átvitelét jelenti.

  R (Reliability) pedig a megbízhatóságot, hibamentességet kéri.

  A 3.1. táblázatban látunk néhány példát, hogy a különböző adatfajták mire érzékenyek a három jellemző közül.

- Az utolsó két bitet eredetileg nem használták, értékük kötelezően 0.

Azonban a routerek (útvonalválasztók) általában nem vették (és legtöbbször ma sem veszik) figyelembe a Type of Service mezőben található értékeket.

A Type of Service mező új jelentése

A Type of Service mező használatának megváltoztatására többféle javaslat is született:

	adat-fájl	hang	tömörített hang	videó	tömörített videó
delay		*	*	*	*
throughput				*	*
reliability	*		*		*

3.1. táblázat. Mire érzékenyek az egyes tartalmak?

RFC 1349 A Type of Service oktettben hagyjuk meg a három bites prioritást, és a következő három bites mezőt (korábban DTR) TOS néven bővítsük ki még egy bittel, amivel a költségek minimalizálását lehet kérni. (Az utolsó bitet pedig 0-ra kell állítani.)

RFC 1455 Az RFC 1349-ben definiált négy bites TOS mező (eddig nem megengedett) 1111 értéke jelentse azt, hogy a továbbítás során olyan útvonal választását kérjük, ami a datagram lehallgatása ellen legjobban véd. (Például rádiós helyett vezetékes csatornán vigyük át, ha lehet.)

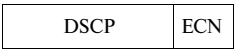
RFC 2474 Használjuk a 8 bitből a baloldali 6 bitet (RFC 791 szerinti prioritás és TDR) *különleges kiszolgálás*[8] (Differentiated Services, szó szerint: megkülönböztetett szolgáltatások) nyújtása érdekében! Ezt a 6 bitet DSCP-nek (differentiated services codepoint) nevezik.

RFC 2481 Használjuk az utolsó két bitet Explicit Congestion Notification (ECN) célokra. Ennek a belső megoldása változott RFC 3168-ban, ezért bővebben nem ismertetjük.

RFC 3168 Használjuk az utolsó két bitet Explicit Congestion Notification célokra, de más módon, mint RFC 2481 javasolta.

[8] Ezt a fordítást [18] használja.

Tehát a jelenlegi állás szerint a Type of Service mező első 6 bitje DSCP, az utolsó két bitje pedig ECN. (3.5. ábra) Ezekkel e jegyzet keretében mélyebben nem foglalkozunk.



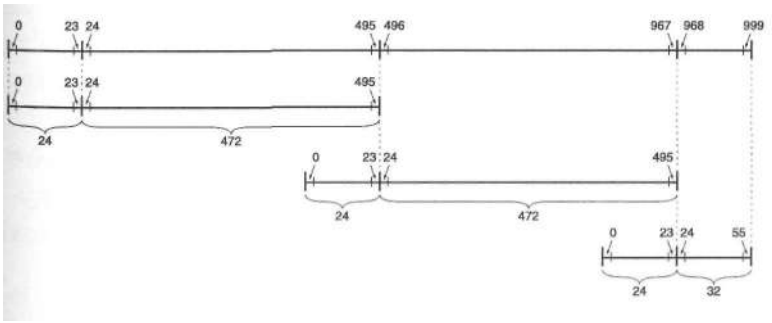
3.5. ábra. A „Type of Service" mező almezőinek új jelentése

Datagram tördelése

A tördelés működését egy példán keresztül mutatjuk be.

Feladat: Egy 1000 oktett méretű IP datagramot tördeljünk a lehető legkevesebb részre úgy, hogy egy darabja maximum 500 oktett méretű lehet.[9] Adatok: IHL = 6, DF = 0. Hány darabra kell tördelni? A keletkező datagram darabokban mi lesz a *Total Length,* a *Fragment Offset* és a *Flags* mezők értéke?

Megoldás: A datagramot 3 darabra kell tördelni: 3.6. ábra.



3.6. ábra. Az IP datagram tördelése

[9] Ezt úgy hívják, hogy: MTU (maximum transmission unit).

A 24 oktett méretű fejrészt mindhárom új datagram fejrészébe bemásoljuk. Egy datagram mérete legfeljebb 500 oktett lehet. Legfeljebb 500 - 24 = 476 oktett adat kerülhetne egy datagramba, de sajnos ez nem osztható 8-cal. A 472 a lehető legnagyobb 8-cal osztható szám, ami 476-nál nem nagyobb. Az 1000 oktett méretű eredeti datagram 1000 — 24 = 976 adat oktettjéből tehát 472-t elhelyezünk az első, majd a második datagramba, így 976— (2*472) = 32 adat oktett kerül a harmadik datagramba. Az első két datagram mérete: 472 -+- 24 = 496 oktett, a harmadiké: 32 + 24 = 56 oktett. A fragment offset mező értéke rendre: 0, 472/8 = 59, 2 * 472/8 = 118. A datagram darabok továbbra is tördelhetők maradnak; a harmadik után pedig már nincs további töredék. Az eredményt a 3.2. táblázatban foglaltuk össze.

	Flags	Fragment Offset	Total Length
1. datagram	001	0	496
2. datagram	001	59	496
3. datagram	000	118	56

3.2. táblázat. IP datagram tördeléséről szóló feladat megoldása

3.1.3. Amivel az IP nem foglalkozik

Most, hogy már ismerjük az IP működését, vizsgáljuk meg, mire van még szükség rajta kívül ahhoz, hogy az alkalmazásaink egymással kommunikálni tudjanak.

Megbízható átvitel

Az **IP** fejrésze tartalmaz ugyan ellenőrző összeget, de ez csak a fejrészre vonatkozik, ebből következik, hogy a datagram adatrészének sérülését nem képes detektálni.

Az átvitel során a datagramokkal a következő - az IP által nem kezelt - problémák fordulhatnak elő:

- tartalom megsérülhet
- datagram elveszhet
- a datagramok sorrendje felcserélődhet
- datagram megkettőződhet

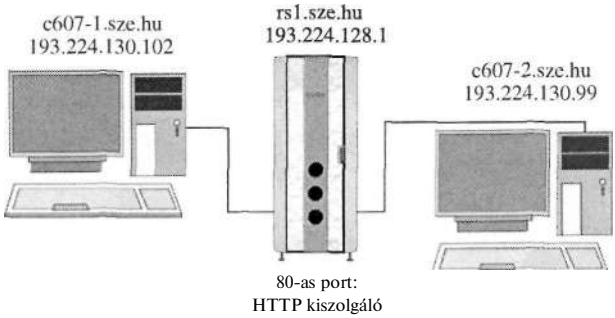
Szükség van tehát olyan protokollra, ami az IP-re építve *megbízható átvitelt* nyújt.

Kapcsolatok azonosítása

A 3.7. ábrán egy webszerver és két kliens gép látható. Természetesen annak sem lehet akadálya, hogy egy kliens gépről egyidejűleg két böngészővel is elérjük ugyanazt a szervert. Ekkor sem szabad a két weblap tartalmának összekeverednie. Márpedig a két **IP** cím azonos az egy gép két külön böngésző esetében! Tehát szükséges a *kapcsolatok azonosítása*. Erre természetesen nem csupán két azonos szolgáltatás, hanem különbözőek (például egy fájl letöltés és egy távoli bejelentkezés) esetén is ugyanígy szükség van. Sőt, már a kapcsolat létrejöttekor egyértelműen ki kell jelölni, hogy milyen szolgáltatáshoz szeretnénk csatlakozni.

Szolgálati közlemények

Ezt ugyan a felhasználók nem igénylik, de teljesen nyilvánvaló, hogy szükség van *szolgálati közlemények* továbbítására is. Az OSI modell fogalmaival élve, időnként egy adott réteg entitásainak egymással is kommunikálniuk kell, például valamilyen **hiba** esetén. **A TCP/IP** modellben nem használjuk az entitás



3.7. ábra. Példa egy TCP kapcsolatra

kifejezést, inkább a *protocol stack*[10] belső kommunikációjáról beszélünk.

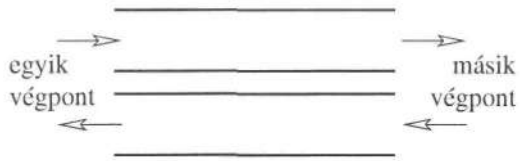
3.2. Transmission Control Protocol

A *Transmission Control Protocolt* az RFC 793-ban definiálták. A TCP megoldja a megbízható átvitelt is, és a kapcsolatok azonosítását is.

Egy kapcsolat két végpontja között megbízható átvitelt biztosít kétirányú adatforgalommal: ami az egyik végponton elindul, a másikon pontosan annak, és ugyanolyan sorrendben kell megjelennie. Olyan ez, mint két szivárgás mentes csővezeték: 3.8. ábra.

A hálózati kapcsolat *végpontját* pedig a *portszám* azonosítja. A portok mindig konkrét szolgáltatást azonosítanak. Például a 80-as porton web kiszolgáló, a 22-es porton ssh szerver működik, stb. Egy *kapcsolatot* pedig 4 szám azonosít egyértelműen:

[10] Az internet protokollkészlet implementációját értjük alatta.



3.8. ábra. TCP kapcsolat modellje

a küldő és a címzett gép IP címe, valamint a küldő és a címzett gépen a portszámok. (így ha például ugyanazon két gép között van két azonos szolgáltatást igénybe vevő kapcsolat, akkor a négyből három szám (a két IP cím és a szerver portszáma) megegyezik, de a negyedik alapján akkor is megkülönböztethető a két kapcsolat.) A portszámok kiosztásával majd később foglalkozunk: 5.1.1.

3.2.1. A TCP szegmens felépítése

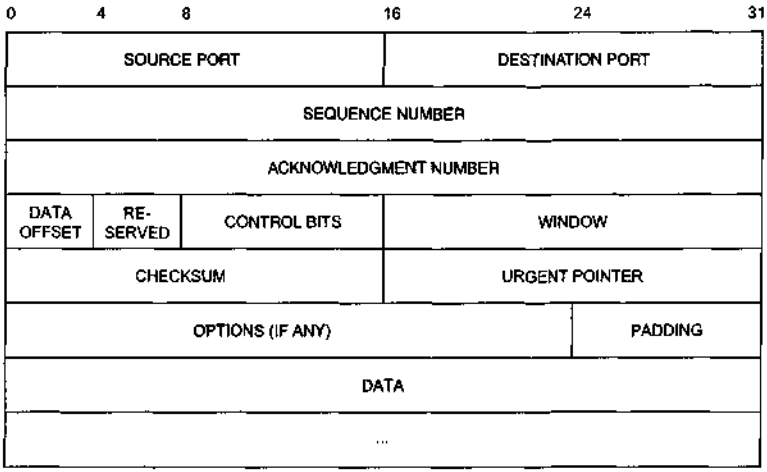
Ismerjük meg most a *TCP szegmens*[11] felépítését! Emlékezzünk rá, hogy a beágyazás-kibontás elvének megfelelően, az egész TCP szegmens az IP datagram adatrészében utazik a hálózaton!

A TCP szegmens felépítése a 3.9. ábrán látható. A TCP szegmens az alábbi mezőket tartalmazza:

Source Port forrás port
: A portszám azon a gépen, ahonnan a szegmenst küldték.

Destination Port cél port
: A portszám azon a gépen, ahova a szegmenst küldték.

[11] Ez a TCP PDU-jának a megnevezése. (Mint ahogyan az IP PDU-ját *datagramnak* hívják.)



3.9. ábra. TCP szegmens felépítése

Sequence Number sorszám
: Az átvitel során az oktetteket sorszámozzuk. A sorszám megmutatja, hogy a szegmens adatmezőjének kezdő oktettje hányas sorszámot kapott.

Acknowledgment Number nyugta
: Annak az oktettnek a sorszámát adja meg, amelyiket várja; addig az összes nyugtázva van.

Data Offset adatmező eltolás
: 32 bites egységekben mérve megadja az adatmező kezdetét a TCP szegmens kezdetéhez képest. Tulajdonképpen a fejrész hossza, mint **IP-nél** az **IHL** mező.

Reserved későbbi használatra fenntartva
: Az RFC 793 szerint még a 4 bites Data Offset után következő 6 bit nem volt használatban. Ez jobbról 2-vel csökkent RFC 3168 alapján.

Control bits vezérlőbitek

RFC 793 szerint csak 6 darab volt, az RFC 3168 vezette be a CWR és az ECE vezérlőbiteket a korábban fenntartott 6 bites terület végén.

A vezérlőbiteknél mindig azok 1 értéke esetén áll fenn az, amit a nevük jelent.

CWR (Congestion Window Reduced) - RFC 3168
A torlódási ablakot szűkítették. Bővebben nem foglalkozunk vele.

ECE (ECN-Echo) - RFC 3168
Torlódásjelző visszhang. Bővebben ezzel sem foglalkozunk.

URG (urgent)
A *sürgős adat mutató* érvényes.

ACK (acknowledgment)
A *nyugta* mező érvényes.

PSH (push)
Jelzi az adat késedelem nélküli továbbításának igényét.

RST (reset)
Egy host összeomlása, vagy más okból összezavart összeköttetés helyreállítására szolgál, illetve ha egy porton semmilyen program sem figyel, akkor az adott portra megkísérelt kapcsolatfelépítés esetén mindjárt az első SYN csomagra RST a válasz.

SYN (synchronize)
Összeköttetés létesítésére szolgál.

FIN (finish)
Összeköttetés bontására szolgál.

Window ablak

Az ablakméretet adja meg. A megbízható átvitelhez szük-

séges nyugtázási mechanizmus használja, részletesen foglalkozunk vele.

Checksum ellenőrző összeg
A teljes szegmensre képezik.

Urgent **Pointer** sürgős adat mutató
A *sürgős adat* az adatfolyam menetét megszakítva a többi adatot megelőzve feldolgozandó adat. A sürgős adat az adatmező elejétől kezdődik, a mutató a sürgős adat után következő (már nem sürgős) adat kezdetére mutat.

Options opciók
Mint IP-nél.

Padding helykitöltés
Mint IP-nél. (A magyarázatban természetesen IHL helyett Data Offset értendő.)

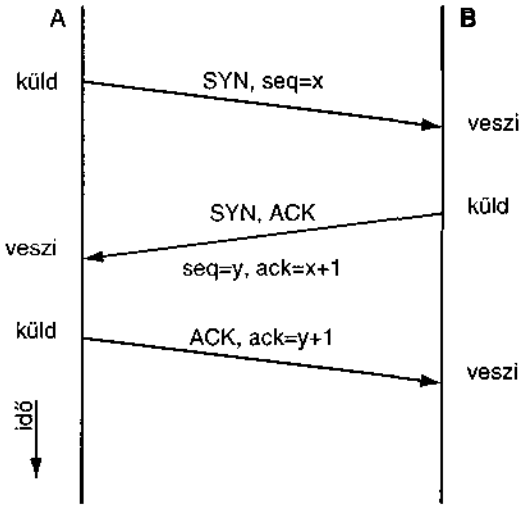
data adatok
Itt található beágyazva a TCP fölötti protokoll adategysége.

Az IP-vel szemben a TCP kapcsolat orientált protokoll, így nem véletlen, hogy a mezők jelentős részének a használata nem magyarázható meg kielégítően egy-két mondattal. Most egyenként megvizsgáljuk, hogyan történik a *kapcsolat felépítése,* a *megbízható átvitel* és a *kapcsolat lebontása.*

3.2.2. TCP kapcsolatfelvétel

Tegyük fel, hogy egy „A" állomás kapcsolatot szeretne kezdeményezni egy „B" állomással. Az „A" állomás létrehoz egy *kapcsolat struktúrát,* amiben a kapcsolat paramétereit tárolja, majd összeállít egy TCP szegmenst a „B" számára: generál egy *kezdő*

sorszámot[12] (x), amit elhelyez a *sequence number* mezőben, valamint beállítja a SYN kódbit értékét 1-re; ezzel jelzi, hogy ez egy új kapcsolat. A szegmenst (az IP réteg segítségével) elküldi „B"-nek. Kövessük ezt nyomon a 3.10. ábrán! A „SYN" feltüntetése azt jelenti, hogy ennek a kódbitnek az értéke 1, a $seq = x$ önmagáért beszél.



3.10. ábra. TCP kapcsolatfelvétel

Amikor a „B" állomáshoz megérkezik a kapcsolat kezdeményezés, a „B" állomás is lefoglal egy kapcsolat struktúrát, amiben eltárolj értéket (és a továbbiakban ebben tartja nyilván a kapcsolat paramétereit). A „B" állomás összeállít egy válasz szegmenst. „B" is generál egy kezdő sorszámot *(y)* amit elhelyez a válasz szegmens *sequence number* mezőjében, és be-

[12]A modern operációs rendszerek (például OpenBSD) kriptográfiailag erős véletlenszám generátort használnak, mert ha bármilyen módon elő're-jelezhető lenne a kezdő sorszám, az támadásra adna lehetőséget.

állítja a SYN kódbitet. Ezen kívül a szegmens *acknowledgment number* mezó'jébe az $x + 1$ értéket helyezi el[13] és beállítja az ACK kódbitet. Ezután elküldi a szegmenst „A"-nak. Összefoglalva: „B" létrehozta az „A" fele irányuló kapcsolatra vonatkozó sorszámot (y) és nyugtázta az „A" által küldött sorszámot. Kövessük az ábrán!

Végül „A" állomás veszi a „B" állomástól érkező TCP szegmenst, és eltárolja a kapcsolatstruktúrájában a „B"-től vett y sorszámot. Létrehoz egy harmadik TCP szegmenst, a nyugta mezőbe beírja az y + 1 értéket, és beállítja az ACK kódbit értékét. Az így elkészített szegmenst elküldi „B"-nek, aki veszi azt. Ezzel a kapcsolat mindkét irányban felépült, a résztvevő állomások mindkét irányú adatfolyamra vonatkozólag egyeztették a sorszámokat, és a nyugtázás is megtörtént. Kezdődhet az adatforgalom.

Ezt a módszert *háromutas kézfogásnak* (three way handshake) nevezzük.

3.2.3. A TCP adatforgalom

A megbízható átvitel

Az **IP** nem megbízható átvitelére építve a TCP megbízható átvitelt nyújt mindkét irányban. Ehhez mindkét irányban sorszámozza az adatfolyam oktettjeit, valamint ellenőrző összeget használ.

Vegyük észre, hogy a TCP az egész szegmensre számítja az ellenőrző összeget, az IP pedig csak a fejrészre. így egy IP datagramba ágyazott TCP szegmens esetén az ellenőrző összegek (két részletben) *hiánytalanul* és *átfedés nélkül* fedik le a datagramot. Az is logikus, hogy melyik protokoll mely részre

[13] Ezzel tulajdonképpen azt állítja, hogy „A"-tól az adatokat x sorszámmal bezárólag vette, $x + 1$-től várja.

számítja az ellenőrző összeget, hiszen az IP feladata a célba juttatás („ha sikerül"), így az IP a fejrészt védi, a TCP feladata pedig a megbízhatóság biztosítása. (A mai hordozó hálózataink maguk is tartalmaznak hibavédelmet, de amikor a TCP/IP protokollcsalád létrejött, még sokkal kevésbé voltak megbízhatóak a hordozó hálózatok.)

Amikor megérkezik egy szegmens, a TCP protocol stack kiszámítja a szegmens ellenőrző összegét, és összeveti a benne szereplővel. Ha nem egyezik, akkor nem használja fel a benne szereplő adatokat. Ha az ellenőrző összeg helyes, akkor még meg kell vizsgálnia a sorszámot *(sequence number),* hogy vajon egyezik-e azzal, amit várt. Ha a vártnál kisebb a sorszám, akkor feltehetően valami „kóbor" szegmensről van szó (például korábban nem érkezett meg időben, újra kérték és most megjött, vagy esetleg a hordozó hálózat hibájából egy adategység megkettőződött), amit nem fogad el. Ha a sorszám a vártnál nagyobb, akkor sem fogadja el.[14] Tehát csak akkor fogadja el, ha a sorszám pontosan egyezik a várt értékkel. Ekkor *előbb vagy utóbb* nyugtát küld a megfelelő értékkel. A megfelelő érték kiszámítása nagyon egyszerű: ha x a sorszám és z oktett érkezett, akkor a nyugta értéke: $x + z$. De mikor is kell a nyugtát elküldeni? A hálózat hatékony kihasználása érdekében nem azonnal. Amennyiben van mit küldeni az ellenirányú adatfolyamban, akkor a nyugtát mintegy „ráültetik" arra az adatfolyamra, vagyis a másik irányú adatfolyam TCP szegmensének a nyugta mezőjét használják fel. Ezt angolul úgy hívják, hogy *piggy backing,* magyarul *ráültetett nyugta* a neve. Természetesen abban az esetben, ha egy meghatározott ideig nincs küldeni való az ellenirányú adatfolyamban, akkor elküldünk egy TCP szegmenst kizárólag a nyugtázás céljából, hiszen a küldő félnek adott (a kapcsolat során megfelelő algoritmussal dinamikusan változta-

[14] Természetesen tárolhatná is abban a reményben, hogy később hátha jó lesz, de ezt nem teszi.

tott) idő (timeout) eltelte után újra kell küldenie az adatot, ha addig nem kapott nyugtát.

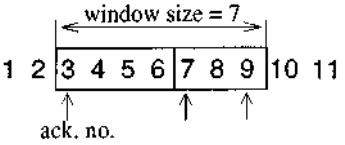
A forgalomszabályozás

Egyáltalán miért van szükség *forgalomszabályozásra* (flow control)? Ennek megvilágítására tekintsük a következő szituációt. Egy „A" állomás (mint forrás) nagy mennyiségű adatot szeretne küldeni egy tőle távoli „B" állomásnak (mint célnak). A kommunikáció sebessége függ mindkét állomás és a közöttük levő hálózat sebességétől. Most koncentráljunk csak a két gépre (a hálózati *torlódás* kezelése (congestion control) meghaladja e jegyzet kereteit). Amennyiben „A" minden újabb adategység küldése előtt megvárná az előzőre adott nyugtát, az nagymértékben lassíthatná a kommunikációt. Nézzünk erre egy számpéldát: legyen „A" és „B" távolsága 200 km, az őket összekötő üvegszál törésmutatója $n = 1{,}5$, a fénysebesség 300000 km/s (vagyis az üvegben 200km/ms), az átviteli sebesség 100 Mbit/s. Ha egyszerre 1000 bájt hasznos adatot küldünk, és a beágyazás során (bőkezűen számítva) még 100 bájt adódik hozzá, akkor a 8800 bit leadásához 88 us szükséges. Az oda-vissza út ideje csak a terjedési időt számítva is 2ms. Ez azt jelenti, hogy hiába lenne szabad az átviteli csatorna, még az idő (és így a csatornakapacitás) 1/20 részét sem tudjuk kihasználni! És ennél jóval nagyobb távolságok és átviteli sebességek is léteznek, ahol az arány sokkal rosszabb! Tehát egy állomás nem várhat az elküldött adategységének a nyugtájára, mielőtt újabbat elküldené. Akkor tehát nem fog várni. így azonban egy másik probléma merül fel: amennyiben „A" lényegesen gyorsabban küldi az adatokat (és a hálózat képes átvinni), mint ahogyan „B" képes azt feldolgozni, előbb-utóbb megtelnek „B" pufferei és az adatok elvesznek. Ekkor a „B"-nél elvesző forgalom a hálózat kapacitását fölöslegesen használja. Márpedig léteznek erősen különböző sebességű számítógépeink, tehát a probléma valós. Szükség van

tehát egy megoldásra! Ha „B" jelezni tudná, hogy vette ugyan „A" adását, de egy ideig többet nem tud fogadni, az még nem jelentene megoldást, hiszen mire a jelzés visszaér „A"-hoz, arra sok adat elindul feleslegesen. Jobb megoldás kell!

A TCP forgalomszabályozásra a *csúszó ablak* (sliding window) módszert használja. Az ablak értéke egy hitelkeretnek tekinthető, azt fejezi ki, hogy egy állomás ennyi adat elküldését engedélyezi a másik fél számára úgy, hogy a másik fél még nem kapott nyugtát róla.

A kapcsolat felépülése után az állomások megegyeznek a kezdő ablakméretben is, amit később (látni fogjuk milyen módon és feltétellel) meg is változtathatnak. A módszer működésének megértése érdekében nézzünk meg egy számpéldát kis számokkal!



3.11. ábra. A csúszóablak működése

A 3.11. ábrán az ablak bal széle az utolsó nyugta értékétől kezdődik (ez 3, ami azt jelenti, hogy 2-ig nyugtázva, a 3 következik) az ablak mérete pedig 7. Ez a küldő számára azt jelenti, hogy a 3-tól 9-ig terjedő sorszámú, összesen 7 darab oktett az, amit anélkül elküldhet, hogy nyugtát kapna. Például elküldi a 3-6 sorszámú oktetteket. Ekkor nem kell nyugtára várnia, hanem küldheti a 7-9 sorszámúakat is. Hogyan csökkentheti a fogadó fél az ablakméretet? Esetünkben a 3-6 sorszámú oktettek vétele után nyugtázza őket (nyugta értéke: 7) és az ablakméretnek legalább 3-nak kell lennie! Ugyanis ha csak 2 lenne, akkor a 9-es ezután nem lenne küldhető, pedig korábban az volt.

Legyen most az ablakméret 5, ekkor a 7-11 sorszámú oktettek küldhetők. Az ablak tehát a folyamat során csúszik előre, ezért hívják *csúszó ablaknak*.

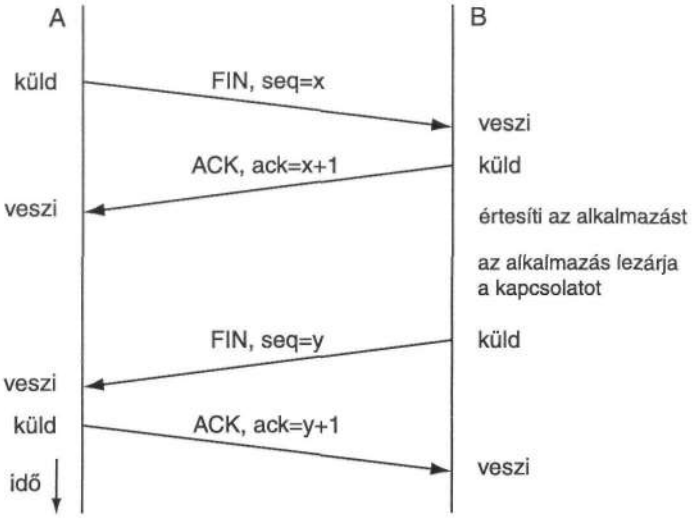
A bevezető számpélda után felmerül a kérdés, hogy vajon a TCP-ben használható legnagyobb ablakméret elég nagy-e? A 16 bites mező 64kB-os[15] ablakméretet tesz lehetővé, ami 512kbit Ennek az átvitele (a fejrészeket elhanyagolva) a példában említett lOOMbit/s-os hálózaton 5,12ms-ig tart, ami nagyobb a 2ms-nál, viszont ugyanabban a nagyságrendben van. Márpedig léteznek gigabites hálózatok, és a 2000 km sem tekinthető a valóságtól elrugaszkodottnak. Erre bizony azt kell mondanunk, hogy a jelenlegi TCP-vel ebben az esetben nem tudjuk egyetlen kapcsolattal kihasználni a hálózat kapacitását. Ez a mindennapi élet szempontjából azért nem okoz súlyos problémát, mert az ilyen nagy távolságú és sebességű hálózatok tipikusan gerinchálózatok, amit nem egy felhasználó forgalmával kell kitöltenünk. Azt azonban látnunk kell, hogy a csúcstechnológiát illetően a jelenleg használt TCP tartalékai kimerültek.

3.2.4. A TCP kapcsolat lebontása

A TCP kapcsolat lebontását a 3.12. ábra mutatja be. Ha az „A" állomásnak nincs több küldeni valója, a FIN kódbit beállításával jelzi a „B" állomás számára, hogy kéri a kapcsolat bontását. A „B" állomás a beállított FIN bit alapján értesíti az alkalmazást, hogy „A" a kapcsolat bontását kérte, valamint nyugtázza az „A"-tól származó sorszámig az adatok vételét. Amikor a „B" állomáson futó alkalmazás is úgy dönt, hogy lezárja a kapcsolatot, beállítja a FIN bitet az „A" felé küldött TCP szegmensben. Az "A" állomás ezt veszi, nyugtát küld, és bontja a kapcsolatot. A nyugta megérkezésekor a „B" is bontja a kapcsolatot. Ezt a megoldást nevezzük *négy utas kézfogásnak* (four way hand-

[15] Egészen pontosan ennél egy bájttal kisebbet, de most ez nem számít.

shake).



3.12. ábra. TCP kapcsolat bontása

3.3. User Datagram Protocol

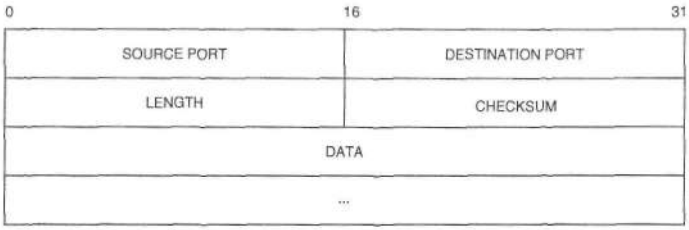
A *User Datagram Protocolt* az RFC 768-ban definiálták.

Vannak olyan esetek, amikor nincs szükségünk megbízható összeköttetésre, de az alkalmazások közötti kommunikációban akkor is szükség van a portokra az alkalmazások azonosításához.

A User Datagram Protocol (UDP) kapcsolatmentes szállítási protokoll: datagramok küldését teszi lehetővé a felhasználók számára összeköttetés létesítése nélkül. Egy olyan kliens-szerver alkalmazás (például DNS névfeloldás, lásd: 5.1.2) számára, amely egyetlen kérésre egyetlen választ küld, sokkal előnyösebb az UDP használata, mint az, hogy TCP kapcsolat felépítésével és lebontásával töltse az időt. Ezen kívül a valós idejű

forgalom számára is alkalmatlan a TCP az újraküldési mechanizmusa miatt.

Az UDP adategységét *user datagramnak* hívják, felépítése a 3.13. ábrán látható.



3.13. ábra. UDP adategységének felépítése

A user datagram az alábbi mezőket tartalmazza:

Source Port forrás port
A portszám azon a gépen, ahonnan a szegmenst küldték.

Destination Port cél port
A portszám azon a gépen, ahova a szegmenst küldték.

Length hossz
A user datagram teljes méretét adja meg.

Checksum ellenőrző összeg
A kiszámításához egy *pseudo headert* tesznek az UDP adategység elé, ami többek között a forrás és cél IP címeket is tartalmazza[16], az adategység végén pedig esetleg egy 0 értékű oktettel kiegészítik, ha anélkül páratlan oktettből állna (mert 16 biten történik az ellenőrző összeg számítás).

[16]így a hibavédelem a portszámon kívül az IP címre is kiterjed.

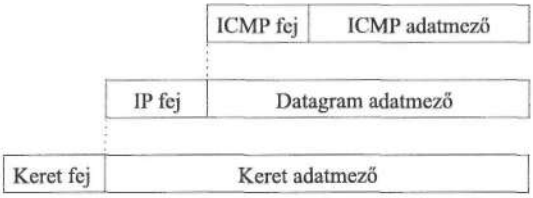
data adatok
: Itt található beágyazva az UDP fölötti protokoll adategysége.

Amint említettük, az UDP nem nyújt megbízható átvitelt. Amit nyújt, az egyrészt a végpontok azonosítása a portok segítségével és erre építve multiplexálás/demultiplexálás, másrészt a fent említett hibavédelem, amely az UDP adategységen túl az IP címekre is kiterjed.

3.4. Internet Control Message Protocol

Az *Internet Control Message Protocolt* az RFC 792-ben definiálták.

Bár az ICMP az IP rétegre építve szállítja a IP szolgálati közleményeit, mégsem tekinthető egy magasabb szintű (a TCP-vel és az UDP-vel egy szinten levő, azaz szállítási) protokollnak. Éppen a funkciója miatt kötelező része minden IP implementációnak. Üzenetei a tanult módon ágyazódnak be: 3.14. ábra.



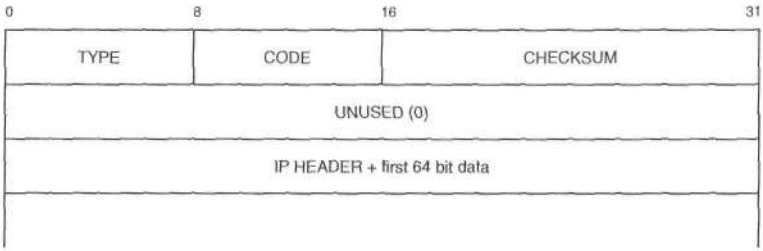
3.14. ábra. Az ICMP adategységének beágyazása.

A hálózat felhasználói számára az ICMP üzenetek általában észrevétlenek, a felhasználó csupán egy jól működő hálózatot lát. Van azonban néhány hálózat-karbantartással kapcsolatos feladat, amikor a felhasználók is igénybe vehetik az ICMP szolgáltatásait, ezekkel 3.4.3-ban fogunk találkozni, de előbb még

megismerjük az ICMP üzenetek formátumát és néhány fontosabb ICMP üzenetet.

3.4.1. ICMP üzenetformátum

Az ICMP üzenetek formátuma egyedi. Ami közös bennük, az az ICMP fejrész első 32 bitjén található 3 adatmező. A további rész kiosztása függ az üzenet típusától. Példaként nézzük meg egy tipikus hibaüzenetnek számító ICMP üzenet felépítését a 3.15. ábrán!



3.15. ábra. A *Destination Unreachable* ICMP üzenet felépítése

Az általános, minden ICMP üzenetre jellemző mezők:

Type típus
: Ez a 8 bites szám adja meg, hogy melyik ICMP üzenetről van szó.

Code kód
: Ennek a 8 bites számnak az értelmezése az ICMP üzenet típusától függ. Gyakran az adott típusú üzenet valamely altípusát jelenti, de az is lehet, hogy érdemben nem használjuk, ekkor az értéke 0.

Checksum ellenőrző összeg

Ugyanolyan algoritmussal képzett 16 bites ellenőrző összeg, mint az IP-nél.

A további mezők tehát üzenettípusonként eltérőek. Több esetben is előfordul kihasználatlan mező, ezt a forrásnak csupa 0 értékű bitekkel kell feltöltenie, a címzettnek pedig figyelmen kívül kell hagynia az értékét. Amennyiben hibaüzenetről van szó, az ICMP üzenet tartalmazza még a hibaüzenetet kiváltó IP datagram fejrészét és az adatrész első 64 bitjét. Vegyük észre, hogy az IP réteg nem tudja, hogy fölötte mi utazik, de akár TCP, akár UDP, ez a 64 bit tartalmazza a hiba kezeléséhez szükséges információt. Miért csak TCP-t és UDP-t említettünk? Azért, mert ICMP üzenettel kapcsolatos hiba következtében nem keletkezhet újabb hibaüzenet! Ez fontos védelem a hibaüzenetek öngerjesztő elszaporodása ellen, viszont ez azt is maga után vonja, hogy ICMP üzenetek nyom nélkül elveszhetnek. Ezt természetesen tolerálni tudjuk, hiszen az IP réteg amúgy sem megbízható!

3.4.2. Fontosabb ICMP üzenetek

Ismerjük meg a fontosabb ICMP üzeneteket! Az egyes üzeneteknél zárójelben megadjuk azok típusszámát is. A felsorolásban az RFC 792-beli sorrendet követjük.

Destination Unreachable (3) cél nem elérhető

Ennek több oka lehet, amiket a Code mező tartalma különböztet meg:

0 = net unreachable a célhálózat nem elérhető

1 = hőst unreachable a célgép nem elérhető

2 = protocol unreachable a kívánt (IP fölötti) protokoll nem elérhető

3 = port unreachable a kért port nem elérhető

4 = **fragmentation needed and** DF **set** a datagram tördelésére lenne szükség, de a DF (do not fragment) bit be van állítva

5 = **source route failed** a *forrás általi útvonalválasztás*[17] sikertelen

Time Exceeded (11) időtúllépés

Ha egy datagram TTL-je lejár (0-ra csökken), akkor az az útvonalválasztó, ahol éppen tartózkodik, köteles eldobni a datagramot. Ilyenkor ezzel az üzenettel jelezheti a forrás számára, hogy eldobta. Tördelt datagram esetén a cél állomás összerakáskor eldobja a töredéke(ke)t, ha a TTL lejár, ilyenkor ugyancsak ezzel az üzenettel jelezheti a forrás számára a hibát.

Paraméter Problem (12) érvénytelen fejrészmező

Ha egy útvonalválasztó vagy egy számítógép nem tudja értelmezni egy datagram fejrészének valamely mezőjét (például Type of Service vagy valamilyen opció), és ennek következtében eldobja a datagramot, ilyen üzenettel jelezheti azt a forrás számára. Ez az ICMP üzenet az első 32 bitje után tartalmaz egy 8 bites pointert. Ha a Code mező értéke 0, akkor ez a pointer az eredeti datagramnak arra az oktettjére mutat, amely a problémát okozta. Természetesen az eredeti datagram fejrésze és az első 64 adatbitje is része az üzenetnek (32 bites határra igazítottan elhelyezve).

Source Quench (4) forráslefojtás[18]

Ha egy útvonalválasztó nem tudja kezelni a túlságosan sok érkező datagramot (nem fér el a pufferében) és eldobja a datagramot vagy egy állomás nem győzi feldolgozni az

[17] Erről 4.1.2-ben fogunk tanulni.

[1] Ezek az üzenetek is tovább növel(het)ik egy hálózat túlterheltségét.

érkező datagramokat, ezzel az üzenettel jelezheti a forrás számára, hogy csökkentse a datagramok időegységenkénti számát. (Ezt a forrás köteles figyelembe venni és a Source Quench üzenetek megszűntéig csökkentenie kell az adás sebességét. Utána fokozatosan újra növelheti, amíg újabb ilyen üzenetet nem kap.)

Redirect (5) átirányítás

Egy router megadhat egy állomásnak egy rövidebb utat a datagramban szereplő *cél* felé, de ettől még továbbítja az eredeti datagramot. (A háttérben az van, hogy a routerekről feltételezzük, hogy pontosan tudják, merre vezet a rövidebb útvonal, míg az állomások nem, de így megtanulhatják.) A kód mező fejezi ki, hogy pontosan mit is értünk cél alatt:

0 = **Network** hálózat

1 = **Host** állomás

2 = **Type of Service and Network** szolgáltatástípus és hálózat

3 = **Type of Service and Host** szolgáltatástípus és állomás

Echo (8) visszhang kérés

A forrás arra kéri a címzettet, hogy küldje vissza az üzenetet. A minden ICMP üzenetben azonos funkciójú első 32 bit után ez az üzenet tartalmaz egy 16 bites azonosítót, és egy 16 bites sorszámot. Ez utóbbi az egymást követő üzenetekben egyesével nő.

Echo Reply (0) visszhang válasz

Az Echo üzenet címzettje ezzel az üzenettel válaszol. A válasz mezői általában megegyeznek a kérés mezőivel, de természetesen a típus mező nem, és az ellenőrző összeget is újra ki kell számítani.

Timestamp (17) időbélyeg kérés
Az Echo üzenetet további 3 mezővel bővíti:

Originate Timestamp küldési időbélyeg

Receive Timestamp vételi időbélyeg

Transmit Timestamp visszaküldési időbélyeg

Az időbélyegek az UTC szerint éjfél óta eltelt időt tartalmazzák ms-ban mérve.

Timestamp Reply (18) időbélyeg válasz
Válasz a Timestamp üzenetre. A válaszoló a mezőket az *Echo Replyhoz* hasonlóan másolja, illetve értelem szerint kitölti.

3.4.3. Felhasználók számára is elérhető ICMP üzenetek

Van két olyan hálózati tesztelésre szolgáló parancs (segédprogram), ami a legtöbb operációs rendszeren megtalálható (esetleg más néven).

A ping parancs

A *ping* parancs egy vagy több *visszhang kérés* ICMP üzenetet küld a felhasználó által megjelölt másik gépre. A címzett pedig *visszhang válasz* ICMP üzenetet küld annak a gépnek, ahonnan a kérés érkezett. A felhasználó ilyen módon tesztelheti, hogy egy másik gép elérhető-e a hálózaton keresztül.

A ping parancs célszerűen kiírja a visszaérkező üzenetből a TTL értékét, így azt is megtudhatjuk, hogy milyen távol van a vizsgált gép (útvonalválasztók számában mérve).

A ping parancs ezenkívül mérni szokta a *visszhang kérés* küldése és a vele azonos sorszámú *visszhang válasz* megérkezése

között eltelt időt, így megtudhatjuk a teljes oda-vissza út idejét (RTT, *round-trip time).*

A traceroute parancs

A *traceroute* parancs egy távoli géphez vezető útvonal során érintett útvonalválasztók válaszidejét[19] deríti ki.

Ehhez többféle trükkös módszert is használhat, például egy elterjedt megoldást, hogy a vizsgált eszköz (router) *33434-es UDP portjára*[20] küld egy UDP csomagot. A beállított TTL-t 1-ről növeli. Amíg túl kicsi, addig *time exceeded* üzenetet kap vissza valamelyik közbenső routertől, amikor pedig már elég nagy, akkor *port unreachable* üzenet érkezik (feltéve, hogy a 33434-es porton nem figyel alkalmazás, ha mégis, akkor a felhasználó választhat más portot). Alternatívaként a felhasználó azt is megadhatja, hogy UDP datagram helyett *echo* ICMP üzenetet küldjön.

3.5. Kiegészítő protokollok

Ahhoz, hogy IP protokollt használhassunk valamilyen hálózati megvalósítás (például Ethernet) felett, szükség van még arra, hogy az általuk használt címek között kapcsolatot teremtsünk.

3.5.1. Address Resolution Protocol

Az **IP** négy oktett méretű címeket használ az állomások azonosítására, míg a hálózati megvalósításoknak is megvan a saját

[19]Az időben benne van a teljes oda-vissza út ideje (RTT - round-trip time), valamint a válasz előállításának az ideje, ami erősen függ a router terheltségétől.

[20]Egy várhatóan nem használt célport, szükség esetén lehet változtatni, BSD-ben lehet TCP-t is használni.

címzési rendszere. Például az Ethernet 6 bájtos címeket használ. Az IP négy oktettjéről a 6 bájtos MAC címre való leképzést hívjuk *címfeloldásnak,* amit az *Address Resolution Protocol* (ARP) segítségével valósítunk meg. Ennek a működéséhez fontos körülmény, hogy mindig csak olyan állomások IP címéhez kell kiderítenünk a hozzájuk tartozó MAC címet, ami velünk azonos fizikai hálózaton van.

Hogyan működik az ARP? Amikor egy számítógépnek vagy útvonalválasztónak szüksége van egy vele szomszédos, ismert IP című másik számítógép vagy útvonalválasztó MAC címére, akkor küld egy broadcast üzenetet, amiben megkérdezi, hogy kihez tartozik az adott ismert IP cím. A broadcast üzenetet mindenki veszi az adott hálózaton, és az IP cím gazdája válaszol: megmondja a saját MAC címét.

Az algoritmus hatékonysága cache-eléssel fokozható:

- amikor egy állomás ARP kérést küld, megadja a saját IP címét is, így a többiek eltárolhatják az *IP cím - MAC cím* párost
- a kapott választ is eltárolhatják a többiek is
- amikor egy állomást bekapcsolnak, akkor az állomás ARP kérést küld a saját IP címére, és utána válaszol is, ezt is tárolhatják a többiek

Ha két gép azonos IP címet szeretne használni, az hiba. Ez akkor derül ki, ha egy gép induláskor küld egy ARP kérést a saját IP címére, és más válaszol. Ilyenkor szabály, hogy az újonnan indulónak illik visszalépnie. Ezt nem mindenki tartja be.

3.5.2. Reverse Address Resolution Protocol

Előfordulhat olyan szituáció is, amikor éppen az ARP fordítottjára van szükség. Például egy merevlemez nélküli munkaállomás

esetén a hálózati kártya egyedi MAC címe alapján szeretnénk megkapni a hozzá tartozó IP címet. Erre is lehetőségünk van, de ehhez szükségünk van egy RARP szerverre. Ilyenkor a RARP kérdésre a RARP szerver válaszol. Miért jó ez? Mert elegendő azonos boot EPROM-ot tenni minden gépbe, amelyek a szükségszerűen különböző MAC cím alapján a RARP segítségével ki tudják deríteni a saját IP címüket, amivel aztán képesek kommunikálni. Az operációs rendszert aztán hálózaton keresztül fogják betölteni. Ma már ritkán használják erre a célra a RARP-ot, helyette van más protokoll, a DHCP, ami lényegesen többre is képes.

3.6. Az Internet Protocol 6-os verziója

Bár a „B" osztályú IP címek elfogyását követően a „C" osztályú címek CIDR segítségével való összevonása, valamint az „A" és „B" osztályú címek visszaadása, illetve kisebb tartományra való cseréje még átmeneti megoldásként alkalmas volt, de nyilvánvalóvá vált, hogy a címteret erősen ki kell bővíteni.[21]

A téma mélyebb megismerésére az irodalomjegyzékben megadott könyv [9] viszonylag jól használható, bár mivel 1998. decemberénél (az IPv6-ot definiáló RFC 2460 dátumánál) előbb írták, így előfordulhatnak eltérések. Az alábbiak írásához felhasználtuk, de sajnos részben elavult [17], az IPv6 fejrész hibásan szerepel benne. (Tanulság: egy forrást érdemes összevetni a vonatkozó RFC-vel!) Egy magyar nyelvű összefoglaló: [1].

[21] Különféle trükkökkel, mint a címfordítás (NAT) még mindig működik az IPv4 rendszer, így még mindig nem kényszer az IPv6-ra való áttérés.

3.6.1. Az IPv6 protokoll kialakításának főbb szempontjai

Sokan és sokféle igénnyel léptek fel, hogy milyen legyen az *új Internet Protocol*[22], ezeknek az érdeklődők magyar nyelven is utána olvashatnak [17] megfelelő fejezetében. A következőket tartjuk igazán fontosnak és előremutatónak:

- Legyen elegendő IP cím - még a címtartomány nem hatékony kihasználása esetén is. Felkészülni az előre nem látható igények kielégítésére is.
- Csökkenteni a forgalomirányító táblázatok méretét. A jelenlegi kétszintű címzés (hálózati prefix + gépcím) helyett valami jobbra lenne szükség, mivel már nagyon sok hálózat létezik.
- A protokoll egyszerűsítése a csomagok gyorsabb feldolgozása érdekében. (Mivel a hálózatok sebessége körülbelül két nagyságrenddel[23] megnőtt, a routereknek egyre kevesebb idő alatt kell dönteniük az útvonalak felől.)
- Biztonság javítása (titkosság, hitelesség). Ez szinte teljesen hiányzott az IPv4 tervezési szempontjai közül.
- Type of Service - szolgálat típusának jobb támogatása. Az IPv4-es DTR biteket és a prioritást nem is szokták figyelembe venni a routerek, most az alkamazások igénye miatt ennél kifinomultabb megoldásra lenne szükség.
- A *többbesküldés* (multicast) jobb támogatása. Például audió és videó műsorszórás gazdaságos megvalósításához.
- Mobilitás támogatása. Szintén új felhasználó igény.

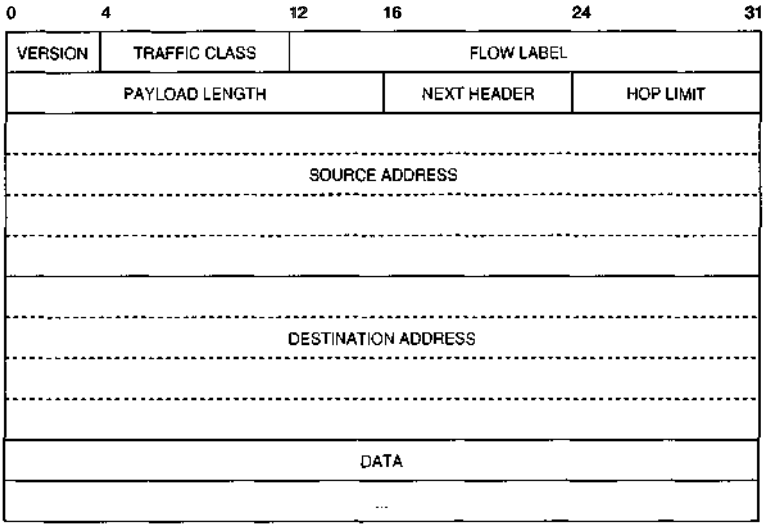
[22] Nevezték úgy is, hogy *next generation IP* vagy röviden: *ngIP*.
[23] például 10 Mbit/s-ról 1 Gbit/s-ra

- A protokoll fejleszthető legyen. Ha később előre nem látható változtatásra lesz szükség, az a protokoll lecserélése nélkül megvalósítható legyen.
- IPv4 és IPv6 együttélése lehetséges legyen. Az átállásnak fokozatosnak kell lennie. Először kisebb szigetek alakulhatnak, majd ezek olvadhatnak össze.

Ezen szempontok alapján - kellően sok megfontolás és vita után - az új protokoll az RFC 2460-ban leírt 6-os verzió lett.

3.6.2. Az IPv6 datagram felépítése

Az IPv6 adategységének felépítése a 3.16. ábrán látható.[24]



3.16. ábra. IPv6 datagram felépítése

Az IPv6 datagram mezőinek jelentése:

[24] Egyszerű, tömör leírás található róla például: [29]

Version (4 bit) verziószám
Értéke természetesen: 6.

Traffic Class (8 bit) forgalmi osztály
Az IPv4 *Type of Service* mező utódja. Az RFC 2474 szerinti *Differentiated Services* megoldást használjuk.

Flow Label (20 bit) folyam címke
Virtuális áramkör jellegű működést tesz lehetővé. Kísérletinek (experimentál) tekinthető, az értelmezése változhat.

Payload Length (16 bit) adatmező hossza
Mivel 16 bit hosszú, így értéke legfeljebb 65535 lehet. Az IPv4-gyel ellentétben itt a fejrész 40 oktettje nem számít bele, ezt fejezi ki a neve is: „hasznos teher hossza".

Next Header (8 bit) következő fejrész
Megmutatja, hogy mit hordoz az IP címek utáni rész. Ez a mező egyrészt tartalmazhatja az IPv6 fölötti protokoll típusának megadását (ilyen értelemben az IPv4 *Protocol* mezőjének utódja), másrészt ezzel ki lehet terjeszteni a fejrészt, ahova további mezők kerülhetnek (ilyen értelemben az IPv4 *Options* mezőjét is helyettesíti). Ez utóbbi esetben *a fő fejrész* (a címekkel befejeződő 40 oktett) után jön egy *következő fejrész*.

Hop Limit (8 bit) átugrás korlát
Funkciója megfelel az IPv4 *Time To Live* mezőjének, de az elnevezése jobban kifejezi a funkcióját, és a mértékegysége sem másodperc (hanem darab).

Source Address (128 bit) forrás IPv6-os címe
Az IPv6 címekkel külön foglalkozunk.

Destination Address (128 bit) cél IPv6-os címe

Megjegyzések

1. Elvileg egy router a *verziószámból* tudná, hogy az utána következő mezőket nem IPv4, hanem IPv6 mezőknek kell tekinteni. Gyakorlatilag nem így történik, mert már a hordozóhálózatban az IPv4-től eltérő protokoll azonosítót használnak az IPv6-ra.

2. A *virtuális áramkör* egy olyan átviteli mód, amelynek során minden adategység azonos útvonalon halad és azonos elbánásban részesül. (Mintha csak egy külön számukra készített áramkörkapcsolt csatornán haladnának. így ebben az esetben nem fordulhat elő például az adategységek sorrendjének felcserélődése.) Virtuális áramkörrel történő kommunikáció során az alábbi lépések játszódnak le:

   (a) útvonalfelépítés, ennek során minden csomópontban információ tárolása a kapcsolatról

   (b) minden csomag a felépített útvonalon halad

   (c) útvonal bontása, ennek során a csomópontokban törlődik a kapcsolatról szóló információ

   (Esetünkben a két állomás IP címe és a FLOW LABEL egyértelműen azonosítja a kommunikációt.)

3. Az *adatmező hosszát* alapesetben a 16 bites érték valóban erősen korlátozza, de lehetőségünk van úgynevezett *jumbogramm* kialakítására is. Ez 64kB-nál jóval nagyobb csomag is lehet, így a routereknek nem kell annyiszor a fejrészt feldolgozniuk, tehát gyorsul a rendszer.

3.6.3. IPv4 - IPv6 fejrészeinek összehasonlítása

Mindkettő verziószámmal kezdődik.

Utána az IPv4-ben a fejrész hossza következik, amiből tudhatjuk, hogy hol kezdődik az adatrész. Az IPv6-nál nincs fejrész hossz mező, mert mindig azonos a fő fejrész, ehhez jöhet hozzá a *Next Header* mezővel való kiterjesztés.

Már az IPv4 *Type of Service* mezőjét is a Differentiated Services célra használják (elvileg), így az IPv6 *Traffic Class* mezője ennek szerves utódja. Ezen kívül a *Next Header* mezőben van lehetőség a kapcsolatra vonatkozólag elrejteni szempontokat, és később a *Flow Label* mező segítségével használni.

A tördelést teljesen megszüntették, hogy ne terhelje a routereket. Helyette - szükség esetén - a teljes útvonalon előre meg kell határozni a maximális használható adategység méretet és azt kell használni. (így eltűntek az *Identification* és a *Fragment Offset* mezők valamint a *DF* és *MF* jelzőbitek.)

A *Time To Live* helyett - helyesebb névvel - *Hop Limitet* használunk.

Az IPv6 fejrészre nem képeznek ellenőrző összeget. Egyrészt a hordozó hálózatok megbízhatósága ezt már nem indokolja, másrészt meg ha - nagyon ritkán - egy-egy datagramot mégis hibásan (ezért potyára) továbbítunk a cél felé (ahol majd egy felsőbb rétegbeli protokoll észreveszi a hibát és eldobja), az sem jelent akkora kárt, mintha az összes routerrel kiszámoltatnánk az ellenőrző összeget.

Az opciókat pedig sikerült elrejteni a Next Header megoldással.

A protokoll tervezői valóban elérték azt a célkitűzést, hogy a fejrész egyszerűsödjön. A jobb megoldások mögött ott áll az IPv4 használata során szerzett nagy mennyiségű tapasztalat.

3.6.4. IPv6 címek

IPv6 címek írása

Az IPv6-os cím 128 bitből áll, 16 bites csoportonként 4 hexadecimális jeggyel írjuk, a csoportokat kettősponttal („:") választjuk el egymástól. A csoportok elején a nullák - egy csoportban legfeljebb háromszor - mindig elhagyhatók. így amennyiben egy 16 bites csoportban 4 nulla áll egymás után, azt mindig helyettesíthetjük egy nullával. Ha egymás utáni 16 bites csoportok csak nullákat tartalmaznak, az összes ilyen csoportot helyettesíthetjük dupla kettősponttal („::"), de ez egy címben csak egyszer alkalmazható, hogy a dekódolás egyértelmű legyen. Lássunk egy példát:

$$A00C:0000:0000:00AE:0000:0000:0000:ABCD$$

$$\Downarrow$$

$$A00C:0:0:AE:0:0:0:ABCD$$

$$\Downarrow$$

$$A00C:0:0:AE::ABCD$$

Az alábbi formában megtartható az IPv4-es formátum is:

$$::193.224.130.161$$

(Ez olyan IPv6 címet jelent, aminek az utolsó 32 bitjében egy IPv4 cím található, a többi bitje pedig 0.)

IPv6 címek csoportjai

Mivel az IPv6 címek 128 bitesek, kellően sok van belőlük. Tervezéskor nem látták előre, hogy milyen szempontok fognak felmerülni a címek struktúrájának kialakításával kapcsolatban. Két szempont kellően logikusnak tűnt:

- támogassuk a kettőnél több szintű struktúrát
- hagyjunk szabadságot arra, hogy mi legyen a strukturálási szempont

Természetesen a tervezőknek valamerre el kellett indulniuk, így eredetileg definiáltak például olyan címcsoportot, ahol földrajzi alapon és olyant is, ahol szolgáltatók szerint osztják ki a hálózatokat. Akkor úgy tűnt, hogy ez jól szolgálja az aggregálhatóságot, de aztán felülbírálták és megszüntették. Jelenleg inkább a *Regional Internet Registryknek* [37] osztanak ki nagyobb tartományokat és azok osztják szét saját hatáskörben a kérelmezőknek.

Az IPv4 címek osztályainak jelzéséhez hasonlóan az IPv6-ban *előtagokat* használnak a különböző címcsoportok megkülönböztetésére. A jelenleg érvényes IPv6 címzési architektúrát az RFC 4291 írja le. Most áttekintjük a fontosabb címcsoportokat.

0000:/8 Reserved Ebben a tartományban vannak a következők:

::/128 Unspecified Address meghatározatlan cím, akkor használjuk, amikor egy host inicializálja magát.

::1/128 Loopback Address A 127.0.0.1 IPv4 loopback címhez hasonlóan használt loopback cím.

::/96 IPv4-Compatible IPv6 Address Eredetileg az IPv4 —> IPv6 átmenethez való felhasználásra szánták, de már más megoldást (lásd következő) találtak ki, ezért érvénytelenítették.

::FFFF:/96 IPv4-Mapped IPv6 Address Ebben az új megoldásban egy tetszőleges x.y.z.w IPv4 címet a ::FFFF:x.y.z.w IPv6 címmel reprezentálnak.
(A korábbi ::x.y.z.w helyett azért kellett másik, mert

változott a módszer, amit az IPv4 —• IPv6 átmenethez szeretnének alkalmazni.)

FE80::/10 Link-Local IPv6 Unicast Addresses Olyan címek, amik csak egy adott fizikai linken érvényesek, például egy Ethernet szegmensen. Az ilyen forráscímmel rendelkező IPv6 csomagokat a routerek nem továbbítják. Olyan esetekben használjuk, amikor például egy IPv6 hálózatban nincs router, vagy *neighbour discovery* (lásd később) esetén.

FEC0::/10 Site-Local IPv6 Unicast Addresses Ez a tartomány ma már nem használt, érvénytelenített. Eredetileg az egy adott site körzetén belüli címzésre definiálták (mint IPv4-nél az RFC 1918 szerinti lokális címeket), de a site fogalmának nehéz meghatározhatósága miatt már nem használatos, lásd: RFC 3879.[25]

FF00::/8 Multicast Addresses Broadcasthoz és multicasthoz használt címek. A cím második bájtja két darab 4 bites mezőre oszlik. Az elsőben különböző flageket definiáltak aszerint, hogy *permanensen kiosztott* (well-known) vagy *tranziens* címről van szó, stb. A második mező a *hatály* vagy körzet (scope). Ez az adott cím érvényességi tartományát adja meg. Ez lehet - többek között - *csak adott linken érvényes* (link-local), *adminisztratív módon meghatározott* (admin-local), *adott szervezeten belüli* (organization-local) és globális.

A multicast cím további része a *csoportazonosító* (group ID). Ez a címzettek egy alcsoportját határozza meg. Előre

[25] Az RFC azt is leírja, hogy a másik probléma, ami IPv4-nél is előjött, az az, hogy a lokális(nak szánt) IP címek esetenként mégis kiszivárognak, ilyenkor aztán mindenféle gondot okoznak, ráadásul még azt sem lehet megtalálni, hogy honnan jöttek.

definiált csoportazonosító például az l-es és a 2-es. Ezek alapján az FF02::2 cím az adott linken levő összes routert címzi meg: a 2-es csoportazonosító az *összes router* (all-routers) cím, az FF02-ben a 2-es pedig link-local tartományt ad meg. Az FF05::2 pedig az összes site-on belüli routert címzi meg.

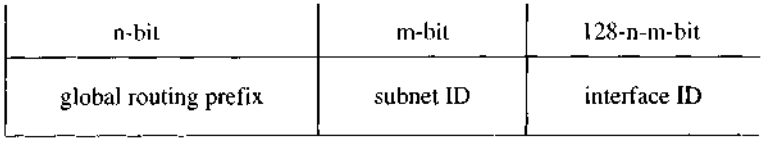
Az l-es csoportazonosító az *összes csomópont* (all-nodes) cím. így az FF02::1 a linken található összes hostot címzi meg, ami nem más, mint a broadcast cím. Vegyük észre, hogy az IPv6 ezen a ponton gyökeresen eltér az IPv4-ben megszokott broadcast címzéstől!

FC00::/7 Unique Local Unicast Addresses Az RFC 4193 definiálja. Ilyen címek használatával olyan helyeken is lehet IPv6-ot használni, ahol nincs hivatalosan kiosztott IPv6 prefix, azaz felhasználásuk célja teljesen hasonló az RFC 1918-ban meghatározott privát IPv4 címekhez. Nagy különbség azonban, hogy az ilyen címeknél a network ID meghatározásához egy *álvéletlen* (pseudo-random) címet használunk, amit az RFC-ben megadott módon lehet előállítani. Több szervezeti egység is generálhat így magának címet, és a pseudo-random algoritmusnak köszönhetően nagyon kicsi valószínűséggel fognak ezek a címek ütközni.

Global Unicast Addresses Az IPv6 címtartomány megmaradó része globálisan routolható unicast címzésre van fenntartva. A címek kiosztását az IANA végzi. Jelenleg a 2000::/3-on belül vannak kiosztva prefixek. Az ilyen címek szerkezete a 3.17. ábrán látható. A *global routing prefix* az IANA által kiosztott tartomány[26]. Az alhálózati címtartomány kiosztása hasonló az IPv4-nél megismerthez. Az *interface ID* a csomópont adott hálózati in-

[26]például a 2001:738:2001::/48 a BME tartománya

terfészének címe. Ez előállítható kézzel is, de lehetőség van automatikus konfigurációra is. Ilyenkor az adott interfész (tipikusan Ethernet) MAC címe alapján állítjuk elő az interface ID-t egy egyszerű eljárással (módosított EUI64 eljárás). Az eljárás az RFC 4291 „A" függelékében olvasható. Az egyes állomások a hálózati címüket is tudják automatikusan konfigurálni az RFC 2462-ben meghatározott módon (stateless autoconfiguration). Ebben az esetben a cím az adott hálózatban levő IPv6 router hirdetései és a megfelelő interface EUI64 címe alapján áll elő.



3.17. ábra. Globális unicast IPv6 címek felépítése

Az IPv4-ben az IPv4 cím és a MAC cím közti összerendelést az ARP protokoll segítségével végzi el egy host. Az IPv6-ban ennek az RFC 2461-ben leírt *neighhor discovery* eljárás felel meg. Ez sok más eljárást is tartalmaz. Számunkra jelenleg az IPv4-hez képest a legjelentősebb különbség az, hogy itt nem külön ARP-szerű protokoll van, hanem az ICMPv6 segítségével oldjuk meg a feladatot.

4. fejezet

Útvonalválasztás

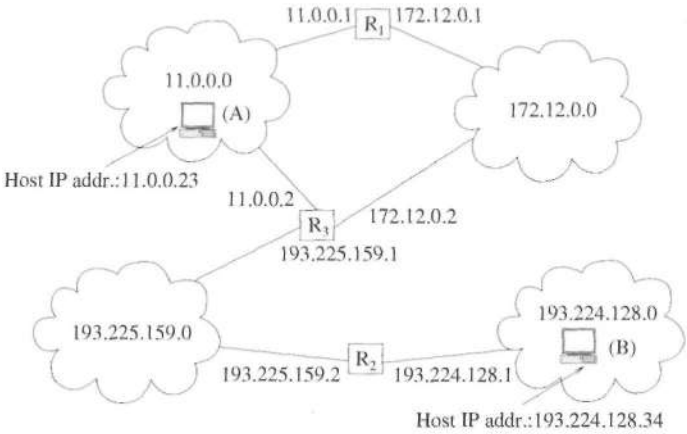
Adott nagy számú hálózat, amelyeket úgy kell összekötnünk egymással, hogy bármely hálózatból bármely hálózatba küldendő csomagjaink eljussanak a címzetthez. Az *útvonalválasztási*[1] (routing) feladat megoldása hálózati protokolltól függő'en más-más lehet. Mi csak *IP routinggal* foglalkozunk. Az egyes hálózatokat összekötő elemeket *útvonalválasztóknak* (router) nevezzük. A routereknek kell eldönteniük, hogy egy adott célcímmel rendelkező IP datagramot melyik irányba továbbítsanak. Minek alapján teszik ezt? A *táblázat alapú útvonalválasztás* esetén a routerek olyan táblázatokkal rendelkeznek, amik tartalmazzák az ehhez szükséges információt. A csomagok továbbításakor csupán felhasználják a táblázataikat a döntésekhez. Egy másik fontos kérdés az, hogy hogyan töltik fel a táblázataikat. Vizsgáljuk meg sorban ezeket, és a még közben felmerülő problémákat!

[1] Szokták még *forgalomirányításnak* is nevezni.

4.1. Datagramok továbbítása

4.1.1. Táblázat alapú útvonalválasztás

A táblázat alapú útvonalválasztást egy példán mutatjuk be.



4.1. ábra. IP hálózat routerekkel

A 4.1. ábrán négy hálózatot látunk, amelyeket az R_1, R_2, R_3 routerek (útvonalválasztók) kapcsolnak össze. Minden hálózathoz hozzárendeltünk egy-egy hálózatcímet. A hálózatokban számítógépek helyezkednek el, és a saját hálózatukból kapnak IP címet. A routerek hálózati interfészei is kapnak IP címet azokból a hálózatokból, ahova csatlakoznak. A routereknek nem kell tudniuk, hogy például a megjelölt „A" géptől a szintén megjelölt „B" gépig mi lesz a datagramok teljes útvonala, elég annyit tudniuk, hogy nekik melyik szomszédjuk felé kell továbbítaniuk a datagramot, ha az egy adott IP című gépnek szól (példánkban ez a „B" gép). A routerek az *útvonalválasztási táblázataik* (routing table) alapján tudják, hogy adott *célhálózatba* (destination network) merre vezet a *következő lépés,* azaz

melyik (milyen IP című) routernek (next hop address) kell továbbítaniuk a datagramokat.

Nagyon fontos, hogy a routing táblázatokban nem az összes lehetséges IP cím, csupán a hálózatcímek szerepelnek! Ez nagyságrendi különbség! A routerek útvonalválasztási táblázatai tehát két oszlopot (destination network address, next hop address) és annyi sort tartalmaznak, ahány hálózat van. A 4.1. ábrán látható hálózat routereinek útvonalválasztási táblázatait megtaláljuk a 4.1. - 4.3. táblázatokban.

destination network address	next hop address
11.0.0.0	direct delivery
172.12.0.0	direct delivery
193.225.159.0	11.0.0.2
193.224.128.0	11.0.0.2

4.1. táblázat. R_1 routing táblázata

destination network address	next hop address
11.0.0.0	193.225.159.1
172.12.0.0	193.225.159.1
193.225.159.0	direct delivery
193.224.128.0	direct delivery

4.2. táblázat. R_2 routing táblázata

A tábla alapú útvonalválasztás algoritmusa

Fontosnak tartjuk hangsúlyozni, hogy ma a routerek nem így működnek, az algoritmust csak didaktikai céllal ismertetjük.

Az útvonalválasztás lépései a következők:

destination network address	next hop address
11.0.0.0	direct delivery
172.12.0.0	direct delivery
193.225.159.0	direct delivery
193.224.128.0	193.225.159.2

4.3. táblázat. R_3 routing táblázata

- A cél IP címből a router annak osztálya alapján meghatározza a célhálózat címét.
- Ezután egy ciklust hajt végre a routing táblájának a soraira. (Az első során kezdi, és veszi a következőt, ha szükséges.)
  - A ciklus magjában összehasonlítja a datagram cél IP címéhez tartozó célhálózat címet a táblázat adott sorában levő *célhálózat címmel* (destination network address).
    * Ha egyezik, akkor a *következő ugrás címe* (next hop address) mező szerint jár el. Amennyiben itt egy IP címet talál, akkor az a következő router IP címét jelenti, tehát neki küldi a datagramot. Amennyiben itt azt találja, hogy *direct delivery,* az azt jelenti, hogy a célhálózat közvetlenül kapcsolódik ehhez a routerhez, tehát *közvetlenül kézbesíti* a cél IP cím alapján a címzettnek. Ezek után a ciklusból kilép, a továbbítás sikeres volt.
    * Ha nem egyezik, akkor a ciklust folytatja a táblázat következő során.

- Ha elfogytak a táblázat sorai, és nem sikerült a datagramot továbbítani, akkor a *destination unreachable* ICMP üzenetet küldi a datagram küldőjének (a forrás IP cím alapján).

Megjegyzés: a táblázat végén nagyon gyakran (például egy host, vagy egy olyan router esetén, amely néhány helyi hálózat és az Internet között routol) van egy *alapértelmezett átjáró* (default gateway), amelynek továbbítja a csomagot, ha másnak nem tudta.

Melegen javasoljuk az Olvasó számára, hogy kövesse nyomon az „A" állomás által a „B" állomás számára küldött datagram útját úgy, hogy végigjátssza az útvonalválasztást minden routeren, ahol áthalad. A megoldás során használja a megadott routing táblázatokat (4.1. - 4.3. táblázatok), és kövesse a datagram útját a 4.1. ábrán!

Gyakorlati probléma

A módszer elméletileg kiváló, a gyakorlatban azonban sajnos nem használható. Ugyanis az A, B, C osztályú IP címek koncepciója azt feltételezte, hogy egy intézménynek egy hálózata van. Ezzel szemben egy-egy intézmény hálózata több fizikai hálózatból áll. (Az semmiképpen sem járható út, hogy minden intézmény annyi hálózatcímet kapjon, ahány fizikai hálózattal rendelkezik, mert nincs annyi hálózatcím!)

Más megoldásra van tehát szükség. Hamarosan megvizsgálunk néhány módszert erre, de előbb tegyünk egy kis kitérőt!

4.1.2. Forrás által végzett útvonalválasztás

Természetesen akár azt is megtehetjük, hogy a forrásnál előre eldöntjük, hogy egy datagramnak milyen úton kell haladnia. Ezt úgy hívják, hogy *forrás által végzett útvonalválasztás* (source routing).

A módszer nyilvánvalóan nem versenyképes a táblázat alapú útvonalválasztással, hiszen jelentős hátránya, hogy az összes számítógépnek tudnia kell minden általa használt cél felé a pontos útvonalat. Tesztelési célokra azonban használják a módszert. A datagram opciójaként lehet megadni, hogy forrás által végzett útvonalválasztást szeretnénk, és itt adjuk meg az útvonalat is. Ráadásul két változata is van:

strict source routing esetén pontosan a megadott útvonalat kell a datagramnak bejárnia.

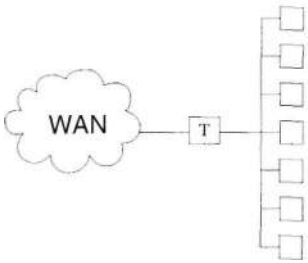
loose source routing esetén a datagramnak át kell haladnia a megadott routereken, de közben érinthet még másokat is.

4.1.3. Transparent router

Tekintsük a 4.2. ábra szerinti hálózatot. Az alaphálózat egy WAN, amelyhez egy speciális routeren (T) keresztül egy LAN-t kapcsolunk. A LAN nem kap saját IP hálózatcímet, hanem a LAN gépei a WAN-hoz rendelt IP hálózatból kapnak címeket. A T-vel jelölt *transparent router* tudja, hogy a WAN-hoz rendelt IP hálózatban szereplő címek közül melyeket osztották ki a LAN-on levő gépeknek. Ha a WAN felől ilyen IP címre érkezik egy csomag, akkor T továbbítja azt a megfelelő gépnek. A T router ugyancsak átveszi a LAN-ból a WAN-ban található gépek IP címeire küldött üzeneteket, és továbbítja őket.

Ilyen módon a WAN és LAN gépei nem szereznek tudomást arról, hogy fizikailag külön hálózaton vannak. Természetesen semmi akadálya annak, hogy több ilyen átlátszó routerrel több LAN-t is illesszünk a WAN-hoz.

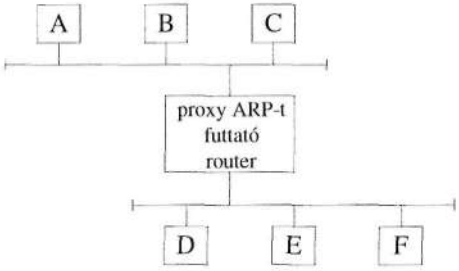
A módszer egyik hátránya, hogy mivel az átlátszó router az állomások számára ténylegesen láthatatlan, hiba esetén nem lehet ICMP üzenetekkel vizsgálni (például nem lehet ping-elni).



4.2. ábra. Transparent router

4.1.4. Proxy ARP

A módszer csak abban az esetben használható, ha a hálózat ARP-t használ az IP cím - MAC cím leképzésre. (Ez általában fennáll.) Adott egy hálózatcímünk és két fizikai hálózatunk, amelyeket egy proxy ARP-t futtató router kapcsol össze: 4.3. ábra.



4.3. ábra. Proxy ARP

Amikor egy gép (legyen most „A") egy olyan gép (legyen most „D") IP címéhez tartozó MAC címet kérdezi meg ARP-vel, amely gép nem vele azonos fizikai hálózaton van, az ARP broadcast üzenetre a keresett „D" gép helyett (amely ezt az

üzenetet nem kapja meg) a proxy ARP-t futtató router válaszol, méghozzá a kért MAC cím helyett a saját MAC címét adja meg. A becsapott „A" gép így a routernek továbbítja az üzenetet, amely továbbadja azt a másik fizikai hálózaton lévő „D" gép számára.

A két külön fizikai hálózaton lévő gépek most sem érzékelik a köztük lévő routert. A proxy ARP-t futtató router sem vizsgálható ICMP üzenetekkel.

Mindkét módszer (a transparent router és a proxy ARP) hátránya, hogy az egységes útvonalválasztási algoritmus helyett attól eltérő pótmegoldást használ, ami ráadásul nagyobb számú fizikai hálózatnál egyre bonyolultabbá válik.

4.1.5. Alhálózati útvonalválasztás

A módszer lényege

Láttuk, hogy bár eredetileg logikus volt az IP cím osztályok használata, ez korláttá vált, mivel egy intézmény hálózata több fizikai hálózatra bomlik. Kövesse akkor ezt az IP címek kezelése is! Bontsuk fel az IP címeknek eredetileg a hálózaton belül a gépek azonosítására használt *gépcím* részét két részre: a hálózatcím felé eső rész legyen az *alhálózat cím,* a jobb oldali része pedig legyen a *gépcím.* (4.4. ábra)

4.4. ábra. IP cím gépcím részének felbontása

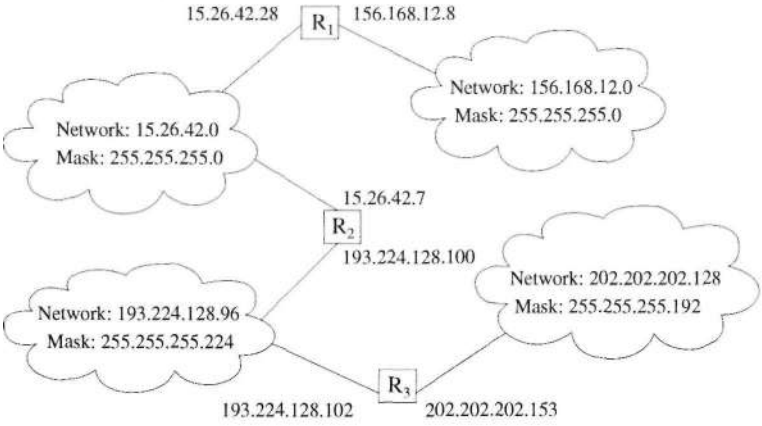
Korábban a hálózatcím és a gépcím határát az IP címek osztálya alapján tudtuk meghatározni. Emlékezzünk rá, hogy erre

feltétlenül szükségünk van az útvonalválasztásnál: a cél IP cím helyett annak hálózatcím részét hasonlítottuk a routing táblában levő hálózatcímhez. Erre most is szükségünk lesz. Mi alkotja most a hálózatcímet? Az osztály alapján is hálózatcímnek tekintett rész és az alhálózat cím együtt. Tehát az új értelemben vett hálózatcímet úgy kapjuk meg egy IP címből, ha kinullázzuk a most gépcímnek tekintett részt. Ehhez be kell vezetnünk egy további fogalmat, ez az *alhálózati maszk* (subnet mask). Ez ugyanolyan hosszú (32 bit) mint az IP cím, 1-eseket tartalmaz a (kibővített) hálózatcím megtartandó bitjeinek a helyén, és 0-kat tartalmaz a (lerövidített) gépcím bitjeinek a helyén. (Ez is látható a 4.4. ábrán.) így ha az IP cím és az alhálózati maszk között bitenkénti logikai ÉS műveletet végzünk, akkor az eredmény a kívánt hálózatcím lesz.

Természetesen módosítani kell a routing algoritmust, és ki kell bővítenünk a routing táblát is. Nézzünk meg egy példát erre! A 4.5. ábrán látható 4 fizikai hálózatot 3 router kapcsolja össze. Ezekhez a fizikai hálózatokhoz különböző „A", „B" és „C" osztályú hálózatokból képzett alhálózat címeket rendeltünk. Vegyük észre, hogy egy alhálózat címének megadásához hozzátartozik a subnet mask megadása is! A *subnet maskkal* kibővített routing táblák a 4.4. - 4.6. táblázatokban láthatók.

subnet mask	dest. netw. addr.	next hop address
255.255.255.0	15.26.42.0	direct delivery
255.255.255.0	156.168.12.0	direct delivery
255.255.255.224	193.224.128.96	15.26.42.7
255.255.255.192	202.202.202.128	15.26.42.7

4.4. táblázat. R_1 routing táblázata



193.224.128.102 202.202.202.153

4.5. ábra. Subneteket tartalmazó IP hálózat

subnet mask	dest. netw. addr.	next hop address
255.255.255.0	15.26.42.0	direct delivery
255.255.255.0	156.168.12.0	15.26.42.28
255.255.255.224	193.224.128.96	direct delivery
255.255.255.192	202.202.202.128	193.224.128.102

4.5. táblázat. R_2 routing táblázata

subnet mask	dest. netw. addr.	next hop address
255.255.255.0	15.26.42.0	193.224.128.100
255.255.255.0	156.168.12.0	193.224.128.100
255.255.255.224	193.224.128.96	direct delivery
255.255.255.192	202.202.202.128	direct delivery

4.6. táblázat. R_3 routing táblázata

A subnet routing algoritmusa

- datagram cél IP címének megállapítása
- ciklus a táblázat soraira
  - — bitenkénti logikai ÉS művelet a cél IP cím és a subnet mask között
  - — eredmény összehasonlítása a destination network címével
    - * egyezés esetén next hop address szerinti továbbítás, és a ciklusból kilépés
    - * ha nem egyezett, akkor a ciklus folytatása, ha van még sor a táblázatban
- hibajelzés, ha nem sikerült továbbítani

A korábbihoz hasonlóan az utolsó lépésben a router itt is közvetlenül a címzettnek kézbesít. Ez azért érdekes, mert minden más esetben a táblázat next hop address mezőjéből kiolvasott IP című routernek kell küldeni, itt viszont a datagramban szereplő cél IP címet birtokló gépnek.

Példa

Játsszunk el egy példát!

Forráscím: 156.168.12.31 Célcím: 193.224.128.102

Az R_1 router megkapja a datagramot és megállapítja a cél IP címet: 193.224.128.102

R_1 ciklust kezd a routing táblájának a soraira.

Az első sorban található subnet mask és a cél IP cím között bitenkénti logikai ÉS műveletet végez:

	193	224	128	102
	↓	↓	↓	↓
	11000001	11100000	10000000	01100110
	255	255	255	0
	↓	↓	↓	↓
&	11111111	11111111	11111111	00000000
	11000001	11100000	10000000	00000000
	↓	↓	↓	↓
	193	224	128	0

Az első bejegyzésben a célhálózat IP címe 15.26.42.0 nem azonos a számítottal, ezért vizsgálja a következő bejegyzést.

Mivel a mask itt azonos az előző bejegyzésnél találhatóval, a számítást nem ismételjük meg (a router természetesen elvégzi). Az eredmény (193.224.128.0) most sem egyezik meg a célhálózat IP címével (156.168.12.0), ezért a ciklus folytatódik.

R_1 bitenkénti logikai ÉS műveletet végez a cél IP cím és routing táblájának harmadik sorában található subnet mask között:

	193	224	128	102
	↓	↓	↓	↓
	11000001	11100000	10000000	01100110
	255	255	255	224
	↓	↓	↓	↓
&	11111111	11111111	11111111	11100000
	11000001	11100000	10000000	01100000
	↓	↓	↓	↓
	193	224	128	96

Az eredmény egyezik a routing tábla harmadik sorában található cél hálózat cím bejegyzéssel, ezért a next hop address szerinti 15.26.42.7-es IP című router felé továbbítja.

Az olvasót bátorítjuk a datagram útjának teljes lejátszására!

Hálózatcímek egyszerűsített írásmódja

Vegyük észre, hogy a *szubnet maszk* (subnet mask) balról jobbra egy határig csupa l-est, utána pedig csupa 0-t tartalmaz. Ezért ha megadjuk, hogy hány darab l-es van a maszkban, azzal a maszkot egyértelműen meghatároztuk. Amennyiben az egyesek számát az IP cím után írjuk egy per jellel („/") elválasztva, akkor az alhálózatot egyértelműen, de rövidebb írásmóddal tudjuk megjelölni.

Lássunk egy példát! Ahelyett, hogy:

IP: 152.66.148.0
mask: 255.255.252.0

azt írhatjuk, hogy:

```
152.66.148.0/22
```

4.1.6. Classless Inter-Domain Routing

Supernetting

Az *alhálózatokra bontásnál* (subnetting) az volt a célunk, hogy IP cím osztályok szerint tekintve egyetlen hálózatból több hálózatot hozzunk létre.

Most nézzük meg a másik irányt! Tegyük fel, hogy 1000 gép számára van szükségünk IP címre. Mivel a „B" osztályú IP címek már régen elfogytak, C osztályúakat kaphatunk. Ha ezeket megfelelően választjuk meg, akkor lehetőségünk van, hogy a subnettinghez teljesen hasonló technikával a kapott négy C osztályú hálózatot egyetlen hálózattá vonjuk össze: *supernetting.*

Nézzük meg egy számpéldán, hogyan is tehetjük meg ezt! Legyen a négy hálózat a következő:

193.224.128.0
193.224.129.0

193.224.130.0
193.224.131.0

Az összevont hálózat ekkor:

193.224.128.0/22

Vegyük észre, hogy ez nem tehető meg tetszőleges számú egymást követő hálózat esetében. Feltétel, hogy az összevonandó hálózatok együttesen folytonos, kettő hatvány méretű és kettő hatvány határra illesztett blokkot alkossanak. Viszont nem szükséges, hogy azonos méretűek legyenek! így például a fenti eredményt kapjuk az alábbi hálózatok összevonásával is:

193.224.128.0/24
193.224.129.0/26
193.224.129.64/26
193.224.129.128/26
193.224.129.192/26
193.224.130.0/25
193.224.130.128/25
193.224.131.0/24

CIDR

A két megoldás (subnetting és supernetting) során ugyanúgy használjuk a maszkolást, így a subnettingnél tanult útvonalválasztási algoritmus (subnet routing) módosítás nélkül használható a supernetting esetén is. Sőt, meg sem kell különböztetnünk, hogy a kettő közül melyikkel nyertük a hálózatcímet: teljesen elfeledkezhetünk az IP cím osztályokról. Osztály nélküli címzést (*classless addressing)* és osztály nélküli útvonalválasztást *(Classless Inter-Domain Routing* - CIDR, RFC-k: 1517 - 1520) használhatunk. Ennek megfelelően módosulnak a megnevezések is. Többé már nem beszélünk hálózatcímről (network

address) és alhálózatcímről (subnetwork address), hanem az IP címeink két részből állnak: a hálózati prefixből és a gépcímből (úgy mint eredetileg az RFC 791-ben, csak már nem az osztályok alapján választjuk szét őket):

```
IP-address ::= { <Network-prefix>, <Host-number> }
```

Bár a CIDR lehetővé tenné a teljes 32 bites címtartomány használatát, ettől még az IP címek kiosztásánál továbbra is csak az egykori „A", „B" és „C" osztályú tartományokat osztják ki felhasználásra, a „D" osztályba tartozó IP címek továbbra is multicast címek, az „E" pedig fenntartott (használaton kívüli).

A CIDR még azt is lehetővé teszi, hogy a routing táblában olyan célokat (hálózatcím+maszk) adjunk meg, amelyek között tartalmazási reláció van. Ezek közül a nagyobb hálózatot, tehát aminek rövidebb a hálózati prefixe (és ami több címet tartalmaz) nevezzük a *kevésbé specifikusnak,* és a kisebb hálózatot, tehát aminek hosszabb a hálózati prefixe (és ami kevesebb címet tartalmaz) nevezzük a *specifikusabbnak.* Az RFC 1812 előírja, hogy ilyen esetben a routereknek a legspecifikusabb illeszkedő cél felé kell a forgalmat irányítaniuk.[2]

A hálózatok közötti tartalmazási reláció lehetősége óriási jelentőségű. Ezzel sikerült az eredeti IP két szintű hierarchikus címzését több szintűvé tenni. így ami távolról egy hálózatnak látszik, az közelről több hálózatra válhat szét, és viszont: egy adott rendszeren belül található több kisebb hálózat kívülről egynek látszik. Ez középtávon megoldást jelent arra a problémára, amit a hálózatok számának növekedése okozott a routereknek.

[2] Az RFC 1812-ben a 2.2.5.2. szól a CIDR témáról, de érdemes tanulmányozni az egész 1812-es RFC-t html formázásban, így egy jól áttekinthető összefoglalót kapunk az útvonalválasztásról. Elérhető például: http://www.freesoft.org/CIE/RFC/1812/index.htm

Történeti áttekintés

Természetesen már régóta a CIDR szerint működnek a hálózataink. Egy rövid történeti összefoglaló a fejlődés lépéseiről, átvéve [33]-ből:

- 1980 - Classful Addressing, RFC 791
- 1985 - Subnetting, RFC 950
- 1993 - Classless Addressing (CIDR), RFCs 1517-1520

Privát IP címek - más jelöléssel

Most már fel tudjuk írni az RFC 1918-ban megadott *privát,* (más szóval *nem publikus* vagy *nem routolható*[3]*)* IP címek tartományát a CIDR jelöléssel:

- *10.0.0.0/8* vagy más néven: *10/8 prefix*
  sőt úgy is nevezik, hogy: *24 bites blokk*
- *172.16.0.0/12* vagy *172.16/12 prefix*
  ez pedig a *20 bites blokk*
- *192.168.0.0/16* vagy *192.168/16 prefix*
  ezt meg természetesen úgy hívják, hogy: *16 bites blokk*

[3]Természetesen a belső hálózatok routerei kezelik őket, csak az Internet gerinchálózati routerei dobják el az ilyen címekre küldött datagramokat.

4.2. Útvonalválasztási táblázatok kialakítása

Természetesen a routerek routing táblázatait nem a rendszergazdák töltik fel kézzel, hanem léteznek olyan protokollok, amik segítségével a routerek kicserélik egymással az információikat a hálózatokról, és „kitalálják", hogy melyik hálózat fele milyen irányba kell küldeni a csomagokat. Mielőtt ezekkel a protokollokkal megismerkednénk, meg kell értenünk néhány alapfogalmat.

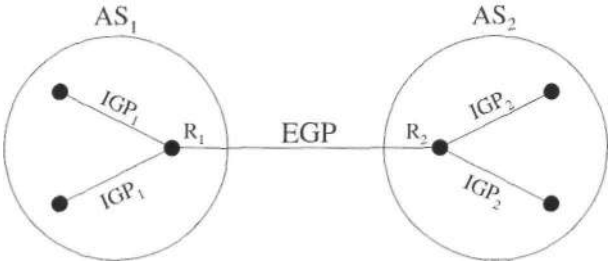
Először is azzal kezdjük, hogy az RFC-kben a routereket *gatewayeknek* nevezik. A 4.6. ábrán rövidítve szereplő fogalmak megnevezése és magyarázata a következő:

AS - Autonomous System (önálló rendszer)
Az egy *adminisztratív hatóság* (Administrative Authority) felügyelete alatt álló rendszer.

IGP — Interior Gateway Protocol (belső routerek közötti protokoll) Az azonos AS-ben lévő routerek egymás között ezzel a protokollal cserélnek útvonal-információt. Ezt az AS-t felügyelő adminisztratív hatóság választhatja meg.

EGP - Exterior Gateway Protocol (külső routerek közötti protokoll) Az AS-ek határán levő routerek ezzel a protokollal cserélnek útvonal-információt a velük szomszédos AS-ek *határ routereivel* (border gateway).

Az IGP és az EGP nem egy-egy protokollt, hanem egy-egy protokoll kategóriát jelölnek. IGP-ből kettőt (RIP, OSPF) EGP-ből egyet (BGP) fogunk megismerni. Azt is látni fogjuk, hogy az IGP és az EGP kategóriákba tartozó protokollokkal szemben eltérőek a követelmények.



4.6. ábra. Alapfogalmak útvonalválasztási protokollokhoz. Forrás: [4]

4.2.1. Routing Information Protocol

Amint a neve is kifejezi, a Routing Information Protocol célja az, hogy a routerek ennek segítségével átadhassák egymásnak az útvonalválasztással kapcsolatos információikat.

A RIP egy IGP, azaz autonóm rendszeren belül működik, sőt alapvetően lokális hálózatokra tervezték, mivel a protokoll üzenetszórást használ az információcseréhez.

Az útvonal hosszát az átmenő routerek számával határozza meg (hop-count). Valamely hálózat felé irányuló útból mindig csak a következő router címét tárolja (amely router felé a datagramot majd továbbítania kell).

A továbbiakban a RIP 2-es verziójával foglalkozunk (RIPv2), amely a hálózatcímekkel együtt a *netmaskot* is továbbítja és kezeli (RFC 1388). (Az eredeti, RFC 1058-ban leírt RIP nem továbbított netmaskot.)

Egy RIP-es router működése

- A router üzenetszórás (broadcast) segítségével megosztja a többi routerrel, hogy mely hálózatokat lát közvetlenül (amelyek hozzá csatlakoznak).

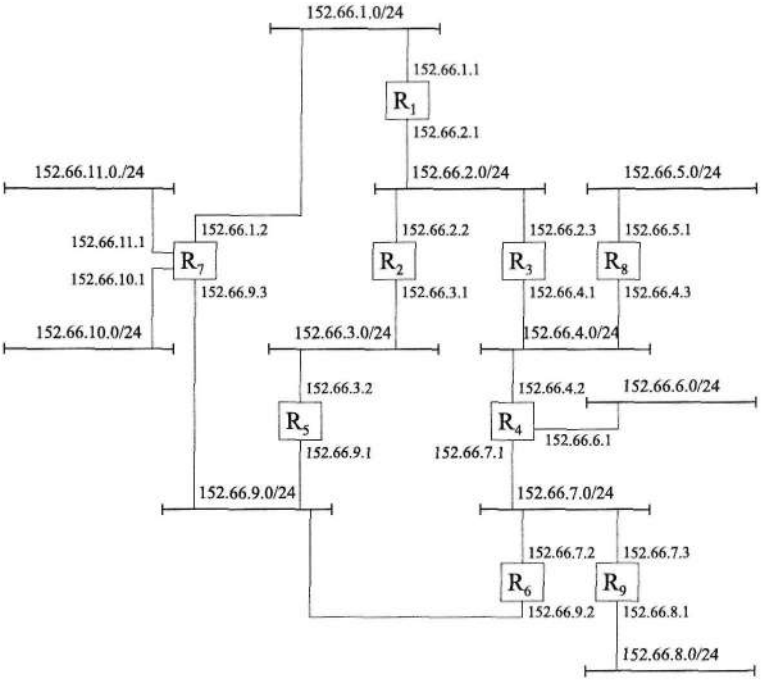
- Tanul, azaz:
  - ha (egy másik router broadcast üzenetéből) tudomást szerez egy eddig még ismeretlen hálózatról, akkor eltárolja azt (netmask + network prefix) az útvonal hosszával és kezdő lépésével (next hop address) együtt. (A kezdő lépés az a router, amely a felhasznált broadcast üzenetet küldte, azon keresztül fogja elérni. A hossz pedig 1-el nagyobb, mint amilyen távolságban az útvonalat hirdető router látta.)
  - ha tudomást szerez egy már ismert hálózat felé az eltároltnál *rövidebb* útról, akkor az eltároltat kicseréli erre.
- A saját információit broadcasttal továbbadja a többieknek: azt is, amit tanult.

Korlátai

- egy hálózat felé csak egy útvonal lehetséges
  Mivel csak a meglevőnél rövidebb új útvonal esetén cserél, az azonos hosszúságú újabb útvonalat nem tárolja el.
- csak a *hop countot* használja távolsági mérőszámként
  Ez az egy mérőszám nem jellemez elég jól: átviteli sebességet, fizikai távolságot nem veszi figyelembe.
- az üzenetszórás csak LAN esetén működik (hatékonyan)
  Természetesen egyenként elküldheti az útvonal információt minden szomszédjának, de az már kevésbé hatékony megoldás.
- nagy hálózat esetén nem használható
  Ha a hálózat mérete (fizikai hálózatok száma) nő, akkor a **RIP** hatékonysága csökken, bizonyos hálózatméret felett a hálózatot teljesen megtöltené a RIP forgalom.

Példa

Nézzünk meg egy példát a RIP működésére! A 4.7. ábrán látható hálózat RIP-et használ. Játsszuk le a RIP működését!



4.7. ábra. Példahálózat RIP-hez

Megoldás: Készítsünk egy táblázatot, aminek a vízszintes fejrészén feltüntetjük a routereket, a függőleges fejrészén pedig a hálózatokat. Első lépésben minden routerhez beírhatjuk, hogy közvetlenül látják azokat a hálózatokat, amelyekhez kapcsolódnak. Tehát minden router oszlopába a hozzájuk kapcsolódó hálózatok sorába beírjuk, hogy: ,,l(k)", ami azt jelenti, hogy az adott routeren keresztül 1 távolságban és közvetlen kézbesítéssel

elérhető az adott hálózat. A routerek időről időre üzenetszórással közzéteszik (broadcastolják), amit tudnak. Ha például R_1 megkapja R_2 adatait, akkor például R_2-*nek* a 152.66.3.0/24 hálózat felé irányuló ,,1(k)" bejegyzése alapján a saját táblázatába a 152.66.3.0/24 sorába beírhatja, hogy „2(2)", ahol az első 2-es azt jelenti, hogy 2 távolságban érhető el, a második 2-es pedig azt, hogy R_2-n keresztül. Természetesen egy valódi router az itt zárójelben szereplő szám helyett az illető router (megfelelő interfészének) IP címét írja be a táblázatába, esetünkben ez az R_2 router R_1 felé eső hálózati interfészének a címe: 152.66.2.2. Honnan tudja ezt? Az R_2-től kapott RIP üzenetben szerepel. (Mi most az IP cím helyett a router sorszámát használjuk, így sokkal kevesebbet kell írni.) Az Olvasót bátorítjuk arra, hogy papíron tényleg játsszon el a feladattal. Célszerű ceruzát használnia, hogy tudjon törölni. A 4.7. táblázatban egy már részlegesen kitöltött táblázat látható.

	R1	R2	R3	R4	R5	R6	R7	R8	R9
152.66.1.0/24	1(k)	2(1)	2(1)	3(3)	3(2)	4(4)	1(k)		
152.66.2.0/24	1(k)	1(k)	1(k)	2(3)	2(2)	3(4)	2(1)	2(3)	
152.66.3.0/24	2(2)	1(k)	2(2)	3(3)	1(k)	4(4)		3(3)	
152.66.4.0/24	2(3)		1(k)	1(k)		2(4)		1(k)	
152.66.5.0/24								1(k)	
152.66.6.0/24			2(4)	1(k)		2(4)			
152.66.7.0/24			2(4)	1(k)		1(k)			1(k)
152.66.8.0/24									1(k)
152.66.9.0/24					1(k)	1(k)	1(k)		
152.66.10.0/24							1(k)		
152.66.11.0/24							1(k)		

4.7. táblázat. RIP-es gyakorló feladat részleges megoldása

Felhívjuk a figyelmet néhány tényre:

- Amint a táblázatba bekerültek a routerek által közvetlenül elérhető hálózatok bejegyzései, többé már nem a hálózat rajza alapján kell dolgozni, hanem a routerek egymástól kapott információi alapján (a rajz természetesen ellenőrzésként használható).
- A táblázat a rendszerben sehol sem létezik, minden router csak a táblázatban a neve alatt szereplő oszlopot ismeri.
- Az Olvasó kaphat a 4.7. táblázattól eltérő részeredményt, ami nem feltétlenül jelent hibát. Még a végeredmény is függ attól, hogy az egyes routerek milyen sorrendben kapták meg mások információit. (Gondoljunk arra, hogy egyenlő távolságnál nem cserélünk.) Ez a valóságban is így van.

4.2.2. Open Shortest Path First

Az IGP egy másik fajtája az OSPF (Open Shortest Path First). Ez sokkal többre képes, de ugyanakkor lényegesen bonyolultabb protokoll, így nem fogjuk részletesen megismerni a működését. Először néhány fontosabb jellemzőjét tekintjük át:

- nyílt (Open), azaz szabadon, ingyenesen hozzáférhető
- nem csak egy, hanem több út is lehetséges adott cél hálózat felé, ezek között adott szempont szerint lehet választani (lásd következő kettő)
- Type of Service figyelembe vétele
- terhelés egyenletes elosztásának támogatása (load balancing)
- hierarchikus felépítés => méretnövekedés támogatása

- authentikáció (router azonosítás) => véd a routerek közötti hamis információcsere (távolságvektor) ellen!
- figyelembe veszi, hogy egy hálózat támogatja-e az üzenetszórást vagy nem (ha igen, használja, ha nem, akkor más megoldást alkalmaz)

Működése során a hálózatokat három kategóriába sorolja:

- Pont-pont közötti összeköttetés
- Többszörös hozzáférésű hálózatok adatszórási lehetőséggel (LAN)
- Többszörös hozzáférésű hálózatok adatszórási lehetőség nélkül

Most röviden (és leegyszerűsítve) összefoglaljuk hogyan működik az OSPF. Az AS-et *területekre* (area) osztja. Az area több együttesen kezelt hálózatot jelent. Minden area kapcsolódik a *backbone* areához (gerinchálózat). Az összes router érzékeli a hozzá kapcsolt vonalak állapotát (link state) és hirdeti a többiek számára, de csak az areán belül. így az egy areán belül található összes router ismeri az areát alkotó hálózatok topológiáját, és a topológiájukat leíró gráfon az ún. Dijkstra algoritmussal kiszámolja a legrövidebb utakat. A területeket a *terület határ routerek* (area border router) kapcsolják a gerinchálózathoz. Ezek a többi területhatár router elől elrejtik a saját területük belső felépítését (mintha az egyes területek pontszerűek lennének). A gerinchálózaton (backbone area) is teljesen hasonlóan határozzák meg a legrövidebb utakat. Lehetséges még külső útvonalak hirdetése is. A protokollt *kapcsolat állapot* (link state) alapú protokollnak is nevezik szemben a RIP-pel és a BGP-vel, amelyek *távolság vektor* (distance vector) alapú protokollok.

4.2.3. Border Gateway Protocol

Az EGP egyik fajtája a BGP (Border Gateway Protocol).

Egy EGP-nél mások a szempontok, mint egy IGP tervezésénél, itt figyelembe kell venni a topológián túl a gazdasági (költségkihatás), és politikai (ellenség felé ne forgalmazzon) nézőpontokat is. Nézzünk erre egy példát! Tekintsünk 3 autonóm rendszert: „A", „B" és „C", melyek mindegyike mindegyikkel össze van kötve, de „C"-nek a világ többi része felé való összeköttetését „A" biztosítja, a „B"-„C" link csak „B" és „C" közti forgalom direkt továbbítását szolgálja. Ha az „A"-„B" + „B"-„C" link összesen jobb, mint az „A"-„C", akkor egy közönséges routing protokoll azt választaná a „C" és a világ többi része közti forgalom továbbítására, ezzel szemben a BGP biztosítja, hogy ne úgy legyen, ha a felek nem azt kívánják: policy routing.

Az AS-ek bekötésük szempontjából három osztályba sorolhatók:

stub (csonk) csak egy bekötése van, végfelhasználók felé ad internet elérést

multi-connected több bekötése van, de nem enged átmenő forgalmat.

transit tranzit, azaz átmenő hálózat, éppen az a célja, hogy más hálózatok forgalmát szállítsa

5. fejezet

További témakörök

A továbbiakban még röviden áttekintünk néhány fontos és érdekes témakört, amelyek részletes tárgyalásával e jegyzet keretében már nem tudunk foglalkozni. Az érdeklődők ezeket alaposabban is megismerhetik az irodalomjegyzékben megadott forrásokból, illetve a választott szakiránytól függően elmélyedhetnek bennük a tanulmányaik során.

A *távközlés-informatika szakirányon* részletesen tárgyaljuk a hálózati alkalmazásokat és azok protokolljait a *Protokollok és szoftverek* tárgy keretében. A szolgáltatások nyújtásával, a szerverek beállításával a *Hálózati operációs rendszerek,* az aktív hálózati eszközök bekonfigurálásával a *Kommunikációs rendszerek programozása,* a biztonsági kérdésekkel pedig a *Hálózatok biztonsága* tárgy foglalkozik. Bővebb információ található a szakirány honlapján: [20].

Szakdolgozat készítés vagy tudományos diákkör keretében pedig hallgatóink önálló alkotó- és/vagy kutatómunkát végezhetnek azokon a területeken, amelyeket e jegyzet elején a számítógép-hálózatok céljaként, feladataként jelöltünk meg.

5.1. Hálózati alkalmazások

A megelőző fejezetekben tanultak feladata, hogy a hálózati alkalmazások számára biztosítsák a kommunikációt.

A hálózati alkalmazások többsége a kliens-szerver modell szerint működik, azaz egy szerver alkalmazás nyújt valamilyen szolgáltatást, amelyet a hálózaton keresztül hozzá kapcsolódó kliensek tudnak igénybe venni.

Hogyan találja meg egy kliens a szervert? Általában a kliensnek (illetve a kliens programot használó felhasználónak) tudnia kell, hogy az igénybe venni kívánt szerver program melyik számítógépen fut. A számítógépekre az IP címük helyett a szimbolikus nevükkel hivatkozhatunk: ezt részletesen tárgyaljuk az 5.1.2-ben. Egy gépen akár több szolgáltatás is futhat; a kliens program az általa kívánt szolgáltatást nyújtó szervert a *portszám* segítségével találja meg.

5.1.1. A portszámok kiosztása

A portszámok kiosztásáért a korábban már említett IANA [27] felelős. Honlapján a portszámok kiosztásán kívül nagyon sok más információ is megtalálható, javasoljuk a tanulmányozását. A honlapról a portszámok kiosztásának módja és a kiosztás aktuális állása is elérhető: [24].

A portszámokat három tartományra bontották:

Well Known Ports „jól ismert" portok: 0-1023
: Ezeket a portokat használják az alapvető, általánosan használt hálózati szolgáltatások. Ezekhez a portokhoz csak privilegizált (Unix alatt például root jogokkal elindított) szerver programok kapcsolódhatnak.

Registered Ports regisztrált portok: 1024-49151
Ezeken a portokon egyéb szolgáltatások érhetők el. Ha valaki kitalál egy szolgáltatást, igényelhet hozzá ilyen portszámot. Ezeken a portokon történő szolgáltatásnyújtáshoz nem szükséges privilegizált módban futtatni a szerver programot.

Dynamic Ports dinamikusan kiosztott portok: 49152-65535
Ebből a tartományból az operációs rendszer oszt ki portokat a kommunikációhoz a programoknak.

A továbbiakban csak olyan hálózati alkalmazásokkal foglalkozunk, amelyek szerver programja valamelyik jól ismert porton érhető el.

Az 5.1. táblázatban tájékoztatásul megadjuk néhány fontosabb hálózati alkalmazás portszámát, valamint az alkalmazások rövidített és teljes nevét.

port	alk. röv.	hálózati alkalmazás neve
20	FTP	File Transfer Protocol, data
21	FTP	File Transfer Protocol, control
22	SSH	Secure Shell
23	Telnet	Telnet
25	SMTP	Simple Mail Transfer Protocol
80	HTTP	HyperText Transfer Protocol
110	POP3	Post Office Protocol - Version 3
143	IMAP4	Internet Message Access Protocol - Version 4
443	HTTPS	HTTP over TLS/SSL
993	IMAPS	IMAP4 over TLS/SSL
995	POP3S	POP3 over TLS/SSL

5.1. táblázat. Néhány fontosabb hálózati alkalmazás portszáma

A továbbiakban a hálózati alkalmazások leírásához felhasznált forrás: [15]

5.1.2. Domain Name System

Mivel az IP címek az emberek számára nehezen megjegyezhető'k, ezért a számítógépeket az IP cím helyett a sokkal könnyebben megjegyezhető *szimbolikus névvel* azonosítjuk.

A szimbolikus nevek

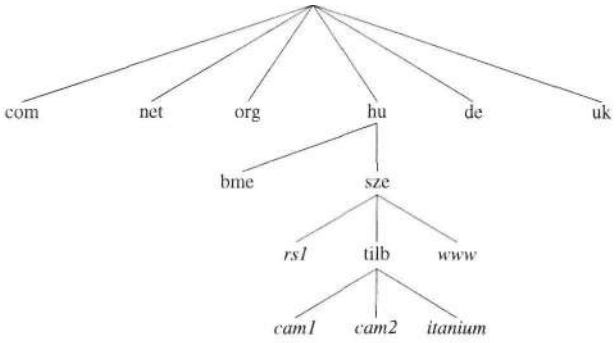
Hogyan épülnek fel a szimbolikus nevek? Példaként tekintsük a következő't:

itanium.tilb.sze.hu

A név mezőit pontok választják el. Az első mező jelenti a *gép* (host) nevét (itanium), a többi a *tartomány* (domain) neve. A tartományok felépítése hierarchikus, fa struktúrát követ. Az 5.1. ábrán látható, hogy a legfelső szinten egy pont („.") van, amit a nevek írásakor elhagyunk. Alatta vannak a *legfelső szintű domain-ek* (TLD: Top Level Domain). Ezek további domain-ekre bomlanak, ami tetszőleges mélységig folytatható. Az ábrán értelemszerűen a „hu" Magyarországot, a „sze" a Széchenyi István Egyetemet, a „tilb" pedig a Távközlés-informatika Labort jelenti. Egy számítógép *teljes nevét*[1] (FQDN: Fully Qualified Domain Name) tehát a gép nevétől a fa struktúrában felfele haladva olvassuk ki, az egyes mezők közé pontot teszünk, és a végét is ponttal zárjuk! A végén álló pont teszi formálisan egyértelművé, hogy egy FQDN-nel állunk szemben. A gyakorlatban a pontot el szoktuk hagyni.

A TLD neveknek két fajtája van. A *generic TLD* (gTLD) nevek a szervezetek fajtája szerinti csoportosítást használják,

[1] Nehéz rá jó magyar kifejezést találni, lehetne például *teljesen kifejtett névnek* hívni, a legbiztosabb, ha FQDN-nek hívjuk.



5.1. ábra. A DNS névhierarchia egy részlete

erre bemutatunk néhány példát az 5.2. táblázatban. Mivel az Internet eredetileg az USA-ban indult, ezek eredetileg ottani kategóriákat jelentettek, aztán bizonyosak az egész világon elterjedtté váltak, de van amelyik (például gov) ma is csak az USA-ban használatos. A *country code TLD* (ccTLD) nevek az ITU (International Telecommunication Union) illetve az ISO 3166 szabvány szerinti két betűs ország megjelölést követik (kivétel: gb helyett inkább uk-t használnak), az 5.1. ábrán ilyenek: hu, de, uk.

gTLD	szervezet fajtája
com	üzleti vállalkozás
org	non-profit szervezet
net	hálózattal kapcsolatos
edu	oktatási intézmény
gov	USA közigazgatás

5.2. táblázat. Példák *generic TLD-re*

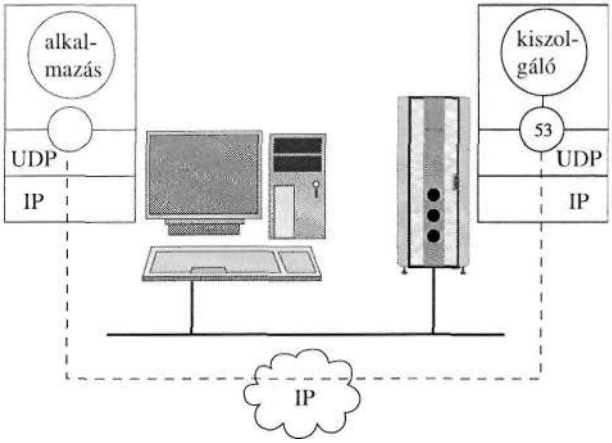
A névfeloldás

Hogyan történik a névfeloldás? Amikor egy programnak szüksége van valamilyen szimbolikus névhez tartozó IP címre, akkor meghívja a megfelelő könyvtári függvényt.[2] Operációs rendszertől függően a részletekben eltérés lehet, mi most a Linux példáján mutatjuk be a továbbiakat. A *névfeloldó* (name resolver) a megfelelő konfigurációs fájl[3] alapján eldönti, hogy milyen sorrendben próbálkozzon a különböző lehetőségekkel. A tipikus beállítás szerinti sorrend esetén először helyi fájlból[4] kísérli meg feloldani, ha nem sikerül, akkor a /etc/resolv.conf fájlban beállított névkiszolgálót kérdezi meg. A gép a továbbiakból csak annyit érzékel, hogy a névkiszolgáló válaszol: megadja a keresett IP címet, vagy jelzi, hogy az adott szimbolikus névhez nem található IP cím.

Mivel a hálózati alkalmazások kliens-szerver kommunikációjának alapvető működése nagyon hasonló, ezért a DNS példáján keresztül a többihez is közelebb kerülünk. Kövessük tehát nyomon a kliens és a szerver kommunikációját! Az 5.2. ábrán látható, hogy a névkiszolgáló az 53-as porton érhető el. A kliens tehát megkérdezi a szervert, amire a szerver válaszol. A kommunikáció a többi hálózati alkalmazás esetén is hasonlóan történik, de ott természetesen más a szerver portszáma, és más a kliens-szerver kommunikáció protokollja is. További különbség, hogy a DNS-sel ellentétben az alkalmazások többsége TCP-t használ, mert megbízható átvitelre van szüksége. Itt azért célszerűbb az UDP, mert egy rövid (egy adategységben átvihető) kérdésre egy rövid választ várunk, és ha esetleg valamelyik elveszik is, akkor majd megkérdezzük újra. A DNS tehát általában UDP-t használ a name resolver és a name server között, mert a *TCP overhead* (a kapcsolat felépítése és bontása) túl nagy lenne, de

Unix esetén ez a gethostbyname() függvény.
régebben: /etc/host.conf, újabban /etc/nsswitch.conf
/etc/hosts

olyan esetekben, amikor a name resolver tudja, hogy több kérést is fog küldeni, akkor általában TCP-t használ. (Például a `netstat`[5] parancs esetén.)



5.2. ábra. DNS kliens-szerver kapcsolat

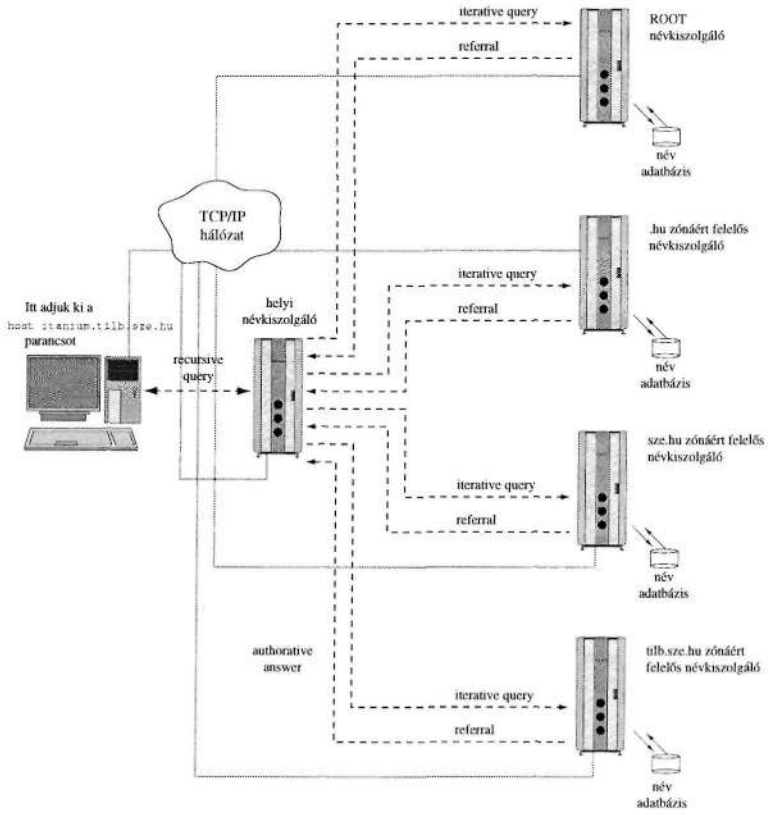
Nézzük meg most azt, hogy a szerver hogyan tudja megadni a választ a kliens kérdésére. Ehhez tudnunk kell, hogy a DNS egy világméretű elosztott adatbázis. A névkiszolgálók egy része[6] felelős egy vagy több zónába tartozó nevek feloldásáért. A *zóna* nem egyezik meg a *domainnel.* A különbséget egy példával világítjuk meg. Az 5.1. ábrán a sze.hu *domainhe* tartozik mindaz, aminek a neve sze.hu-ra végződik:

```
rs1.sze.hu
cam1.tilb.sze.hu
cam2.tilb.sze.hu
itanium.tilb.sze.hu
```

[5]lásd: A.3. függelék
[6]Nem mindegyik, lásd a 150. oldalon: *cacheing only name serverek.*

www.sze.hu

A sze . hu *zónába* azonban csak az rs1 . sze . hu és a www.sze .hu tartozik bele, ezeket a sze.hu zónáért felelős névkiszolgáló képes feloldani, míg a másik hármat nem ismeri, hanem delegálja őket a tilb.sze.hu zónáért felelős névkiszolgálónak.



5.3. ábra. A névfeloldás folyamata

Kövessük végig az 5.3. ábrán, hogyan alakul a névfeloldás folyamata, ha valahol egy számítógép megkérdezi a helyi névkiszolgálójától az itanium.tilb.sze.hu gép IP címét! A gép a helyi névkiszolgáló felé *recursive queryt* küld, ami azt jelenti, hogy a névkiszolgáló feladata, hogy teljes egészében elvégezze a feloldást. A helyi névkiszolgáló a többi névkiszolgálót mindig *iterative queryvel* szólítja meg, ami azt jelenti, hogy azok feladata csak annyi, hogy egy lépéssel közelebb vigyék a kérdezőt a feladat megoldásához a rendelkezésükre álló információ alapján, de nem kell más névkiszolgálóhoz fordulniuk. Tehát a helyi névkiszolgáló megszólítja valamelyik root névkiszolgálót (ilyenből több is van), amely válaszában megadja, hogy melyik névkiszolgáló felelős a hu zónáért. Az ilyen választ úgy hívják, hogy *referral*. A helyi névkiszolgáló a következő kérdést a hu zónáért felelős névkiszolgálóhoz intézi, amelynek válaszából megtudja, hogy melyik névkiszolgáló felelős a sze.hu zónáért. Most már a sze.hu zónáért felelős névkiszolgálót kérdezi meg, amely megadja, a tilb.sze.hu zónáért felelős névkiszolgáló IP címét. Ezt megkérdezve pedig már *authoritative answert* kap: ez a keresett IP cím. Ha mondjuk a nincsilyen.tilb.sze.hu gép IP címének kiderítését kíséreljük meg, a folyamat akkor is ugyanígy zajlik le, de a végső *authoritative answer* az lesz, hogy nincs ilyen bejegyzés.

Néhány megjegyzés:

1. Bár a névkiszolgálók is rendelkeznek szimbolikus névvel, a használat során mindig az IP címükre van szükségünk. A hallgatók gondolják végig, hogy mire mennénk a szimbolikus nevükkel!

2. A névfeloldás hatékonyságát *cache-eléssel* fokozzák: a névkiszolgálók a kiderített információt az authoritative forrás által megadott TTL lejártáig tárolják; legközelebb már nem kell megkérdezniük - ez gyorsítja a névfeloldást,

valamint lényegesen csökkenti a névkiszolgálók és a hálózat terhelését. Létezik *negatíve cache* is: a nem létezést is tárolják, ezzel is csökkentik az authoritative name server terhelését.

3. Amennyiben egy telephelyen nem foglalkoznak DNS adminisztrációval (mert például a cég központi telephelyén látják el, vagy az internet szolgáltatóra bízzák) akkor is üzemeltethetnek helyben névkiszolgálót a névfeloldás gyorsítására, ezt a saját zónával nem rendelkező névkiszolgálót úgy hívják, hogy *cacheing only name server.*

4. A DNS a bemutatottnál lényegesen bonyolultabb, például rendelkezik tartalékolással: minden zónát legalább két névkiszolgáló képes feloldani.

5. A DNS iránt mélyebben érdeklődőknek javasoljuk a következő angol nyelvű szakkönyvet: **[3]**

Reverse mapping

Bizonyos esetekben arra is szükség van, hogy az IP cím alapján határozzuk meg a szimbolikus nevet, amelyhez az IP cím tartozik. Ezt fordított irányú leképzésnek, angolul *reverse mappingnek* nevezzük. A leképzés ugyanazt az infrastruktúrát használja, amit a szimbolikus névről IP címre való leképzés is használ, méghozzá olyan módon, hogy a leképzéshez szükséges struktúrát a DNS névfában az `in-addr.arpa` alatt hozták létre. Ennek részleteivel már nem tudunk foglalkozni.

Domain regisztráció

Ismerjünk meg néhány fontos fogalmat a regisztrációval kapcsolatban!

registry Egy TLD adminisztrálásának felelőse, delegálja az adott TLD subdomainjeit, de az ügyfelek közvetlenül nem fordulhatnak hozzá.

registrar Magyarul regisztrátor, általában egy profit orientált cég (de akár egyesület is lehet), akihez az ügyfelek fordulhatnak domain név regisztrációért.

registration A domain név regisztráció folyamata.

A hu domain adminisztrálásának felelőse az Internet Szolgáltatók Tanácsa. Honlapján [28] megtalálható a hivatalos regisztrátorok listája, valamint a regisztrálással kapcsolatos szabályok. Az általuk közölt *Domainregisztrációs szabályzat* világos fogalom magyarázattal kezdődik, megtalálhatók az igénylés, a regisztrálás, a vitás esetek kezelésének szabályai, valamint egy domain igénylő lap minta is, amin látható, hogy mit kell megadni domain igényléskor (például: kért domain név, névkiszolgáló, többféle kapcsolattartó, számlázási cím). A honlapon található egy kereső is, amivel ellenőrizhetjük, hogy egy név szabad-e még, illetve ha már nem, akkor megtudhatjuk, hogy ki regisztrálta.

5.1.3. További alkalmazások

Nagyon röviden tárgyalunk még néhány fontosabb alkalmazást. Ezekre általában jellemző a már megismert kliens-szerver architektúra és a TCP szállítási protokoll használata.

Simple Mail Transfer Protocol

A levelezéshez szükség van az e-mail címekre. Nézzünk meg erre egy egyszerű példát:

lencse@rs1.sze.hu

Ebben az esetben a „@" után az rsl.sze.hu gép FQDN-je áll, előtte pedig egy érvényes postafiók az említett gépen.[7]

A felhasználó egy *levelező kliens* (Mail User Agent) program segítségével megírja a levelet, ahol természetesen meg kell adnia a címzett e-mail címét is. Amikor a levelet elküldi, a levelező kliens az SMTP protokoll segítségével átadja a levelet a programban beállított *kimenő SMTP szervernek* (outgoing SMTP server). Ez egy *üzenetátviteli ügynök* (Message Transfer Agent) amely ugyancsak az SMTP protokoll segítségével eljuttatja a levelet a címzett leveleit kezelő levelező szervernek, amely elhelyezi azt a címzett felhasználó postafiókjába. A legegyszerűbb esetben a címzett belép arra a gépre, ahol a postafiókja található, és a levelező kliens programja segítségével elolvassa a levelet.

SMTP-nél nem garantált az átvitel, sikertelenség esetén általában hibaüzenetet kapunk.

POP3 és IMAP4

A Post Office Protocol (3-as verziója) arra nyújt lehetőséget, hogy egy felhasználó hozzáférjen a leveleihez (elolvashassa, törölhesse őket) anélkül, hogy be kellene lépnie a leveleit kezelő gépre (ahol közvetlenül elérhetné őket). Ehelyett a POP3 szerver fér hozzá közvetlenül a felhasználó postafiókjához, a felhasználó levelező programja pedig POP3 kliensként a POP3 protokoll segítségével (felhasználói névvel és jelszóval azonosítva magát) letölti a felhasználó számára a leveleket, amelyeket ezután a felhasználó helyben tud kezelni. A részleteket most mellőzzük.

Az Internet Message Access Protocol (4-es verziója) a POP3-hoz hasonló funkciójú azzal az eltéréssel, hogy lehetővé teszi, hogy távolról elérjük, kezeljük a szerveren tárolt leveleinket

[7] A helyzet ennél lényegesen bonyolultabb is lehetne, de most nem célunk mélyebbre ásni.

és azok részeit (például csatolt fájlok), sőt arra is lehetőséget nyújt, hogy az egész postafiókot letöltsük és off-line módosítsuk, majd utána újra szikronizálhassuk a helyi és a távoli postafiókot. Ezért IMAP4-et használunk akkor, ha a leveleinket több helyről is el szeretnénk érni.

Használati szempontból fontos még megemlíteni, hogy beállítástól függően a POP3 esetleg nyílt szövegként (titkosítás nélkül) átviszi a felhasználó jelszavát a hálózaton, illetve mindkét protokoll titkosítás nélkül viszi át a leveleket.

File Transfer Protocol

Az FTP segítségével fájlokat tudunk egy távoli gépről letölteni vagy oda feltölteni.

A kliens-szerver kommunikációt illetően abban különbözik a többi tárgyalt protokolltól, hogy a kliens és szerver között két TCP kapcsolatot használ. A *vezérlő kapcsolat* a szerver 21-es portjára csatlakozik, és a műveletek során mindvégig fennáll. Az *adat kapcsolat* portszáma a szerveren a 20-as, és csak az adatátvitel idejére (fájl le- vagy feltöltés, könyvtárlista letöltése) jön létre, utána lebomlik.

A felhasználó szempontjából jelentős tulajdonság, hogy az adatkapcsolat kiépítési iránya eldönthető. *Aktív módban* a kliens a vezérlőcsatornán keresztül megadja a szervernek, hogy milyen IP címen és milyen porton várja a szerver csatlakozását, és a kapcsolatot a szerver építi ki a kliens felé. *Passzív módban* éppen fordítva történik: a szerver adja meg az IP címet és portszámot, és a kliens kezdeményezi a TCP kapcsolat felépítését. Amennyiben a kliens oldalon NAT-ot vagy tűzfalat használunk, csak passzív módban tudjuk az FTP-t használni!

Az FTP is nyíltan visz át minden információt a kliens és a szerver között!

Az FTP-t használhatjuk úgy is, hogy felhasználói névvel és jelszóval bejelentkezünk, de használhatjuk úgy is, hogy az

Anonymous felhasználói névvel jelenkezünk be. Ilyenkor jelszóként az e-mail címünket kéri, amit évtizedekkel korábban még bátran megadtunk, ma azonban a kéretlen levelek miatt nem szívesen adjuk meg, anélkül is beenged, legfeljebb a „@" karakter meglétét ellenőrzi. Természetesen anonymous FTP-vel csak ahhoz férünk hozzá, amit a nyilvánosságnak szántak.

Telnet

A Telnet programmal egy távoli gépre tudunk bejelentkezni, majdnem úgy, mintha előtte ülnénk. Problémák:

- A teljes kliens-szerver kommunikáció (a jelszavas authentikáció is) titkosítás nélkül zajlik.
- Csak karakteres felület áll rendelkezésünkre, grafikus nem.

HyperText Transfer Protocol

A weblapok átviteléért felelős protokoll. Igen elterjedten használják, problémát okoz itt is a titkosítás hiánya.

SSH, SCP és SFTP

Az **ssh** elsődleges funkciója a biztonságos távoli bejelentkezés. Működése során nyilvános kulcsú kriptográfiát használva először egy biztonságos csatornát hoz létre a kliens és a szerver között, majd ezen keresztül jelszó vagy kulcs alapú authentikációt használ. Ha megfelelő körültekintéssel használjuk, akkor biztonságot nyújt, egyébként nem.[8] Lehetőség van *port továbbítás* (port forwarding) használatával X session vagy fájlok átvitelére is.

[8]Például egy szerver kulcsának elfogadása végzetes lehet, ha az nem a szervertől, hanem a támadótól származik. Jelszó alapú authentikáció esetén ilyenkor jelszavunk a támadó birtokába kerülhet.

Az scp parancs a Unix cp parancsának távoli gépekre történő kiterjesztése, ahol az átvitel biztonságosan, az ssh segítségével történik. (Természetesen az ssh-hoz teljesen hasonlóan csak akkor biztonságos, ha körültekintően használjuk.)

Az sf tp is az ssh-t használja fel az átvitelre, de funkcionalitásában több az scp-nél, mert nem csupán fájl átvitelre, hanem például könyvtár listázásra és fájl vagy könyvtár törlésére is használható. Ne tévesszük össze az FTPS-sel, ami SSL/TLS feletti FTP!

HTTPS

A HTTP biztonságos változata, TLS (Transport Layer Security) vagy SSL (Secure Socket Layer) fölött működő HTTP. Jelentősége igen nagy, hiszen a mindennapi élet során egyre több tranzakciót hajtunk végre webes felületen keresztül. (Például: banki műveletek, rendszerek távoli adminisztrációja.) Biztonsági kockázatot jelent, hogy az SSL 2.0-s verziója biztonsági rést tartalmaz. Ezért kizárólag SSL v3.0-t vagy TLS 1.0-t használjunk!

Az ssh/scp-hez hasonlóan ez is nyilvános kulcsú kriptográfián alapul, itt egy (potenciálisan hamis) tanúsítvány elfogadása jelenti a legnagyobb veszélyt! További veszélyforrás, ha a böngészőnk nem ellenőrzi a CRL-eket (Certificate Revocation List - a visszavont tanúsítványok listája).

A biztonsági megfontolások ismertetése sajnos meghaladja e jegyzet kereteit, de szeretnénk felhívni mindenkinek a figyelmét, hogy a mai felhasználói szokások (tanúsítványok automatikus elfogadása) mellett egy sikeres *DNS elleni támadás*[9] esetén a felhasználók gyakorlatilag védtelenek. A hálózati támadásokról az ellenük való védekezésről ajánlott olvasmány a *Hálózatok*

[9] A támadó eléri, hogy a kliens programok (például web böngészők) az általa hamisan megadott IP című gépre csatlakozzanak.

biztonsága tárgy jegyzete, amely jelenleg bárki számára elérhető [20].

POP3S, IMAP4S és FTPS

A POP3, az IMAP4 és az FTP protokollok biztonságos változatai, a biztonságot az SSL/TLS réteg nyújtja.

5.2. TCP/IP socket interface

A témához javasolt irodalom: [4]. Ezen kívül hasznos a Unix elektronikus kézikönyve (man), kiindulásként érdemes megnézni az IP, TCP, UDP, ICMP protokollok implementációjáról szóló leírást. Linux alatt[10]:

```
man 7 ip | tcp | udp | icmp
```

Unix alatt a processzek közötti kommunikációra (inter-process communication) több megoldás is létezik[11]:

- *Szignálok* küldése (erősen korlátozottak a lehetőségek)
- *Osztott memória* (shared memory) használata
- *Socketek* használata

Ezek közül csak az utolsó használható, ha a processzek eltérő gépeken futnak.

Mi is az a socket? Nem más, mint kommunikációs végpont. Első (kicsit sánta) közelítésként tekintsük a fájl általánosításának. Ami a hasonlóság a kettő között, az főleg a kezelésükkel

[10] A „|" jel a megszokott módon vagylagosságot jelent, tehát: man 7 ip, man 7 tcp, stb. Más Unix verzió alatt lehet, hogy másik kötetben találhatók, például MAC OS X alatt a 4-es kötetben vannak.

[11] IPC még a message queue / semaphore megoldás is.

kapcsolatos: használat előtt meg kell nyitni, aztán lehet bele írni, lehet belőle olvasni és a végén le kell zárni. Az eltérés azonban nagyon jelentős. A fájl egy olyan adathalmaz, amelyet valahol tárolnak. A socket pedig az információ csere egyik végpontja. Ha két ilyen végpontot megfelelő módon egymáshoz rendelünk, akkor amit az egyiken küldünk, a másikon azt fogadhatjuk; a kommunikáció természetesen mindkét irányban működik. A *socket interfész* (socket interface) pedig az a programozási felület, amelyen keresztül a hálózati alkalmazások elérhetik a TCP/IP protocol stack szolgáltatásait.

5.2.1. A TCP/IP socket interface programozása

A továbbiakban megismerjük a socket alapú kommunikáció elemeit. A TCP/IP socket interfészének programozását Unix környezetben, C programozási nyelven mutatjuk be.[12]

Socket létrehozása

Mielőtt bármit is tennénk, létre kell hoznunk egy socket-et. Erre szolgál az alábbi függvény hívás:

```
int socket(int domain, int type, int protocol);
```

domain: a protokollcsaládot fejezi ki.
Az 5.3. táblázatban megadtunk néhány példát a protokollcsaládra.

type: azt fejezi ki, hogy milyen jellegű a kommunikáció.
Számunkra a következők érdekesek:

SOCK_STREAM megbízható kétirányú kapcsolat
Megvalósítása TCP-vel történik.

[12]Figyelem! Linux alatt a C programozás a man 2-es kötetében található, ha ezt nem írjuk ki, akkor esetleg más azonos nevű programok, függvények, stb. leírását kapjuk!

SOCK_DGRAM datagram szolgálat
(kapcsolatmentes, nem megbízható, a maximális csomaghossz korlátozott)
Megvalósítása UDP-vel történik.

SOCK_RAW hozzáférés a hálózatai szintű protokollhoz

protocol: megadja, hogy milyen protokollt kell használni a sockethez. Általában egy adott socket típushoz csak egy protokoll létezik (egy protokollcsaládban), de ha esetleg több is, akkor ezek közül lehet vele választani. Ha csak egy protokoll van, akkor a protokollt kifejező szimbolikus konstans értéke 0, így ezt be is írhatjuk.

return: descriptor - egy egész szám, ami azonosítja a socketet, hasonlóan, mint az open()-nel történő fájl megnyitás esetén. Hiba esetén a visszatérési érték: -1.

szimbolikus konstans	jelentés	man page
AF_UNIX, AF_LOCAL	Local communication	unix(7)
AF_INET	IPv4 Internet protocols	ip(7)
AF_INET6	IPv6 Internet protocols	
AF_IPX	IPX - Novell protocols	
AF_X25	ITU-T X.25 / ISO-8208 protocol	x25(7)

5.3. táblázat. Példák protokollcsaládra

A *protokollcsalád* (protocol family) (más néven *címcsalád:* address family) megnevezésében az 5.3. táblázatban a szimbolikus konstansoknál az „AF_" (address family) kezdetet használtuk. A régebbi a BSD stílusú rendszerekre jellemző a „PF_"

előtag, de a modern BSD-ben és más rendszereknél is az „AF_" (address family) előtagot használják.

Helyi IP cím és portszám megadása.

Ahhoz, hogy a socketünket használni tudjuk, hozzá kell kötnünk egy helyi IP címet és portot. Egy sockethez az alábbi függvényhívással rendelhetünk helyi IP címet és portot:

```
int bind(int sockfd, struct sockaddr *my_addr,
         socklen_t addrlen);
```

sockfd: socket descriptor.

my addr: mutató egy sockaddr típusú címstruktúrára, ami a beállítani kívánt helyi IP címet és portszámot tartalmazza. AF_INET esetén ez sockaddr_in típusú. (A struktúra definícióját megtaláljuk a paraméterlista után.)

addrlen: megadja a címstruktúra méretét bájtokban.

return: hiba esetén -1, egyébként 0.

```
struct sockaddr_in {
  sa_family_t sin_family; /*address family: AF_INET*/
  u_intl6_t  sin_port; Aport in network byte order*/
  struct in_addr sin_addr;  /* internet address */
};

/* Internet address: */
struct in_addr {
  u_int32_t s_addr; /*address in network byte order*/
};
```

Amint látjuk, a socket címe egy IP cím és egy port szám kombinációja. Az IP nem rendelkezik port számmal, azt csak

a fölötte levő TCP vagy UDP használja. Amint a megjegyzés mutatja, mind az IP cím, mind a portszám tárolása a *hálózati bájt sorrendet* követi, ami azt jelenti, hogy a legnagyobb helyi értékű bájt van elöl (MSB). Erre még visszatérünk 5.2.2-ben.

Socket csatlakoztatása célcímhez

Mivel a TCP kapcsolat orientált protokoll, nyilvánvalóan létre kell hozni a kapcsolatot, mielőtt kommunikálni szeretnénk. Az UDP kapcsolatmentes protokoll, de ekkor is beállítható egy célcím, amit így már később nem kell megadni a kommunikáció során. Mi a továbbiakban csak a kapcsolat orientált esettel (TCP) fogunk foglalkozni.

```
int connect(int sockfd, const struct sockaddr *
            serv_addr, socklen_t addrlen);
```

sockfd: socket descriptor.

serv_addr: mutató egy sockaddr típusú címstruktúrára, ami a kívánt cél IP címet és portszámot tartalmazza. AF_INET esetén ez is sockaddr_in típusú, lásd: `bind()`.

addrlen: megadja a címstruktúra méretét bájtokban.

return: ha sikerült a kapcsolódás, akkor 0, hiba esetén **-1.**

Várakozás az ellenoldal csatlakozására.

A kapcsolat létrehozása nem szimmetrikus. Mindig az egyik fél, a *szerver* várakozik a másik félre *(kliens),* a kliens pedig kapcsolódik a szerverhez a fent megismert `connect()` függvényhívással.

A szerver részéről ez két függvényhívást jelent:

```
int listen(int sockfd, int backlog);
```

sockfd: socket descriptor.

backlog: hagyományosan a létrehozás alatt álló kapcsolatok (TCP 3 utas kézfogás) számának felső korlátja. Ha ezt kimerítették, a szerver a további kapcsolatokat vissza fogja utasítani.

A 2.2-es Linux kernelben ennek jelentését megváltoztatták, a már létrejött, de még accept()-tel el nem fogadott kapcsolatok maximális számát jelenti. Bővebben: man **listen.**

return: hiba esetén -1, egyébként 0.

```
int accept(int sockfd, struct sockaddr *addr,
           socklen_t *addrlen);
```

sockfd: socket descriptor.

addr: mutató egy címstruktúrára. Ide kerül be a kapcsolódó fél IP címe és portszáma.

addrlen: mutató egy kétirányú paraméterre: híváskor megadja a címstruktúra méretét, visszatéréskor pedig a visszaadott cím tényleges méretét bájtokban.

return: siker esetén a létrejött kapcsolat eléréséhez a socket descriptor, hiba esetén -1.

Adatok küldése

A fájlba való írás függvény segítségével végezhető: A legegyszerűbb:

```
ssize_t write(int fd, const void *buf, size_t count);
```

Ez a függvény ugyanaz, amit (alacsony szintű) fájlba írásra is használunk.

fd: socket descriptor.

buf: mutató arra a memóriaterületre, ahol az adatok találhatók. Ezek, úgy ahogy vannak, átvitelre kerülnek.

count: az átvinni kívánt bájtok száma.

return: hiba esetén -1, egyébként az elküldött bájtok száma.

További lehető'ségeket nyújt még a **writev, send, sendto, sendmsg** függvények használata, lásd **man.**

Adatok fogadása.

```
ssize_t read(int fd, void *buf, size_t count);
```

Ez a függvény ugyanaz, amit (alacsony szintű) fájlból való olvasásra is használunk.

fd: socket descriptor.

buf: mutató arra a memóriaterületre, ahova az adatok kerülnek.

count: a fogadni kívánt bájtok száma.

return: hiba esetén -1, egyébként a fogadott bájtok száma.

További lehető'ségeket **nyújt** még a **readv, recv, recvfrom, recvmsg** függvények használata, lásd man.

A kapcsolat bontása

Minden socket () függvényhívással megnyitott, vagy **accept** ()-tel kapott socketet le kell zárni:

int close(int fd);

fd: socket descriptor.

return: hiba esetén -1, egyébként 0.

Fontos megjegyzés: a függvények visszatérési értékét mindig ellenőrizzük!

A hálózati kommunikációval kapcsolatban példaprogramokat mutatunk be, amit a hallgatók a tárgy weblapjáról letöltenek és gyakorlaton kipróbálnak.

5.2.2. A hálózati bájtsorrend figyelembe vétele

Több oktettből álló adatok (például egész számok) esetén számítógépünk architektúrájától függően eltérő lehet az adatoknak a memóriában való tárolása. A bájtok sorrendjét illetően két lehetőség van:

LSB Least Significant Byte first[13]
A legkisebb helyiértékű bájt van elöl (az alacsonyabb memória címen), így van ez például az Intel gyártmányú processzorok esetében.

MSB Most Significant Byte first[14]
A legnagyobb helyiértékű bájt van elöl (az alacsonyabb memória címen), így van ez például a Motorola gyártmányú processzorok esetében.

A *hálózati bájtsorrend* (network byte order) az MSB-t követi, így amennyiben az általunk használt architektúra nem ilyen, ügyelnünk kell a konverzióra. Hordozhatónak szánt programok esetén úgy kell több bájtos adatokat kezelnünk, hogy mindkét fajta architektúrán helyesen működjön a programunk. Az egész számok eltérő hosszára is figyelnünk kell!

[13]Angolul gyakran hívják *little-endiannak,* magyarul néha *alvégnek.*
[14]Angolul gyakran hívják *big-endiannak,* magyarul néha *felvégnek.*

A témához kapcsolódóan felmerül egy másik érdekes kérdés is; ez a *bájtok bitsorrendje*[15] (például az átvitel közben):

lsb least significant bit first
A legkisebb helyiértékű bit van elöl.

msb most significant bit first
A legnagyobb helyiértékű bit van elöl.

A bájt és bitsorrend jelzése között ezt a nagy- és kisbetűs, rendkívül logikus megkülönböztetést [11] használja. Az alvég/felvég problémáról bővebben: [34]

5.3. Teljesítőképesség-vizsgálat

Számítógép-hálózatok, sőt általában kommunikációs rendszerek hardver és szoftver elemei, protokolljai esetén fontos azok teljesítőképessége.

Most rövid betekintést nyújtunk azokba a módszerekbe, amiket ilyen célra használni szoktak. A téma iránt mélyebben érdeklődőknek ajánljuk az alábbiakban is felhasznált forrást: [32] valamint: [10].

5.3.1. Célok, alapfogalmak

Vizsgált rendszerként gondolhatunk akár egyetlen Ethernet hálózatra vagy szerverre, de akár egy országos méretű hálózatra is. Nézzünk meg néhány kérdést, ami felmerülhet:

- Mire képes a rendszer?
- Milyen a kihasználtsága (tartaléka)?
- El fogja-e bírni a forgalmat, ha … ?

[15] angolul: *bit endianness*

- Milyen késleltetések, illetve késleltetésingadozások lesznek, ha ... ?
- Mekkora és milyen forgalmat engedhetünk még a rendszerre, hogy adott minőségi paramétereket tartani tudjunk?
- Hol vannak a szűk keresztmetszetek?
- Mit és hogyan kellene bővíteni?
- Adott bővítés után hogyan alakulnak a jellemzők?

Ilyen, és hasonló kérdésekre kereshetjük a választ, ha teljesítőképesség vizsgálattal foglalkozunk.

Vezessünk be néhány fogalmat:

Felajánlott forgalom Azon csomagok összessége (időbeli eloszlásukat is beleértve), melyeknek az átvitelét a forgalomforrások kérik.

Puffer ideiglenes tár (memória) a hálózati eszközökben. Ebben tárolják a csomagokat, amíg továbbítani nem tudják őket. Mivel a kapacitásuk véges, ezért adatvesztés történhet.

Átvitt forgalom Azon csomagok összessége (időbeli eloszlásukat is beleértve), melyeknek az átvitele megtörtént.

Az átvitt forgalom nem egyezik meg a felajánlott forgalommal, mert a rendszer nem végtelen kapacitású. Az átvitel során csomagvesztés lehetséges, de ha nincs is, a várakozások miatt a forgalom időbeli eloszlása megváltozik.

A forgalom jellemzése

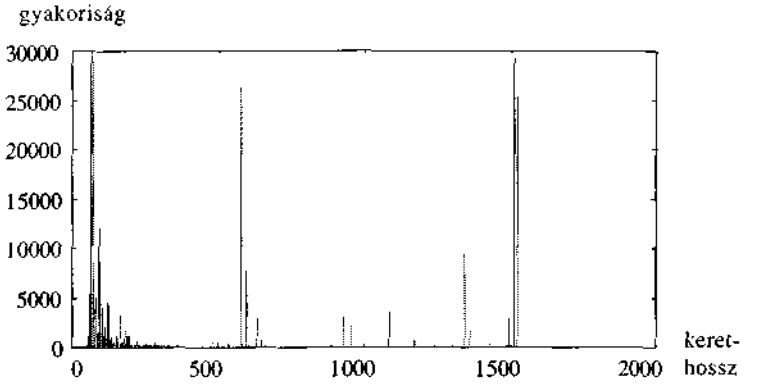
A forgalmat általában nem tudjuk egyetlen számértékkel jellemezni. Különböző szempontokból más-más jellemzői fontosak. A forgalom néhány fontos jellemzője:

- időegység alatt érkező / átvitt csomagok száma (főként az útvonalválasztók terhelését jellemzi)
- időegység alatt érkező / átvitt bitek száma (főként az átviteli csatornák / vonalak terhelését jellemzi)
- érkezési időköz eloszlás (inter-arrival time), két egymást követő csomag érkezési időpontjának különbsége. Egyszerű modellekben például exponenciális eloszlásúnak tekintik.
- csomaghossz eloszlás
- *burstösség* - a forgalom nem független az előző időszaktól (illetve annak forgalmától), bizonyos „csomósodás" figyelhető meg.

Forgalom jellegének leírása, jellemzőinek mérése olyan kérdéseket vet fel, amelyek messze meghaladják e jegyzet kereteit. Most nézzünk meg két példát mért forgalmi jellemzőkre: az 5.4. ábrán egy *kerethossz eloszlás*[16], az 5.5. ábrán pedig egy *érkezési időköz eloszlás* látható.

Figyeljük meg, hol vannak a kiugró értékek az 5.4. ábrán látható kerethossz eloszlásban, és mi ezek oka:

[16]Mivel szimulációs vizsgálatainkat tipikusan hálózati szinten szoktuk végezni, ezért általában csomagokról beszélünk - néha még akkor is, ha nem hálózati szinten dolgozunk. Itt most a precizitás kedvéért említünk kereteket, a mérést ugyanis adatkapcsolati szinten végeztük.

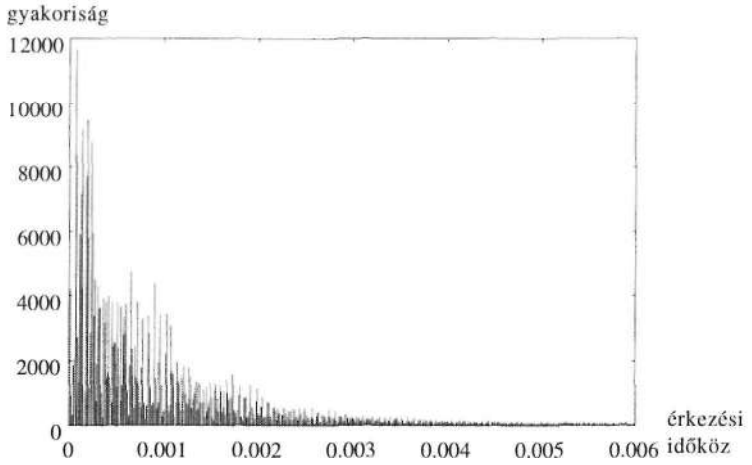


5.4. ábra. A BME FDDI gerinchálózatán 1996-ban mért forgalom kerethossz statisztikája (A 67-es hossz értéknél a gyakoriság értéke 130000 (!) körül van, ezért csonkolt.) [2]

A nagyon rövid adategységek okozói azok az alkalmazások, amelyek 1 karakternyi információ átvitelét igénylik, valamint a TCP nyugták.

A közepes hossz kiugró aránya a hálózatos szabványokban keresendő: RFC 791 szerint minden IP hosztnak képesnek kell lennie legalább 576 oktett méretű datagramok fogadására; ennél nagyobb datagramokat csak akkor szabad küldeni, ha a forrás hoszt meggyőződött róla, hogy a cél hoszt képes azt fogadni.

A hosszú keretek pedig az Ethernet megengedett kerethossza miatt akkorák, amekkorák. (Az FDDI keretmérete jóval nagyobb, de mivel gerinchálózatról van szó, a forgalom döntő része a reá kapcsolt Ethernet hálózatokból származik.



5.5. ábra. A BME FDDI gerinchálózatán 1996-ban mért érkezési időköz gyakoriságok statisztikája [2]

Az 5.5. ábrán látható érkezési időköz eloszlásnál megfigyelhető az exponenciálisra emlékeztető burkológörbe, ami azt mutatja, hogy a források Poisson folyamattal való modellezése nem rossz közelítés, de jól látható, hogy nem is pontos!

Mielőtt a teljesítőképesség-vizsgálati módszerekre rátérünk, definiáljunk egy fontos fogalmat!

Modell alkotás

Definíció: A **modellezés** (vagy modell alkotás) olyan emberi tevékenység, amely során egy valóságos (létező vagy elképzelt) rendszernek egy valamilyen eszközkészlettel kezelhető, általában egyszerűsített változatát hozzuk létre. A modell valamilyen számunkra fontos tulajdonságban hasonlít a valós rendszerre.

Például az áruházak kirakatában található ember alakú bábuk a testi felépítés szempontjából hasonlítanak az emberre: hasonlóan áll rajtuk a ruha, mint az embereken. Az egér pedig bizonyos gyógyszerek hatása szempontjából tekinthető az ember modelljének.

A modellezésnek a továbbiakban nagyon fontos szerepe lesz, két módszerünk is használni fogja.

Kommunikációs rendszerek teljesítőképesség-vizsgálatára alkalmazható módszerek:

1. Mérés (a valóságos rendszeren való mérés)
2. Analitikus módszer (matematikai számítás)
3. Szimuláció (a rendszer modelljén végrehajtott kísérlet)

Ebbe a három módszerbe tekintünk most bele.

5.3.2. Mérés

A legkézenfekvőbb megoldás, hogy mérjük meg, amit tudni szeretnénk. Egy jól végzett méréssel (mérés sorozattal) pontos, hiteles adatokhoz juthatunk. Gyakran a másik két módszer esetén is szükségünk van arra, hogy a bizonyos adatokat méréssel szerezzünk meg.

Azonban a mérésen kívül gyakran szükségünk van a másik két módszerre is. Miért? Milyen hátrányai, akadályai lehetnek a mérésnek? Nézzünk meg néhányat:

- a mérés beavatkozást jelent a rendszerbe
  Ez néha problémát okozhat. Például forgalmat okoz, erőforrásokat foglal, stb.
- a méréshez a rendszernek léteznie kell
  A rendszer megépítése drága lehet, vagy a rendszert esetleg nem is lehet megvalósítani, mert például még a tervezett rendszer elemei sem léteznek.

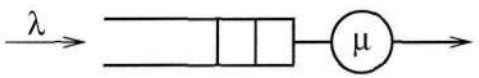
- a mérésnek jogi akadályai vannak
- a mérésnek biztonsági akadályai vannak
  A mérés a mérést végzőre, a mért rendszerre, vagy a rendszer adataira veszélyt jelenthetne.
- a mérést időben nem lehet kivitelezni
- a mérés túl drága lenne
  Emberi erőforrások, műszerek ára miatt.

Ilyen esetekre van két másik módszerünk.

5.3.3. Analitikus módszer

Az *analitikus módszer* azt jelenti, hogy a vizsgált rendszer *matematikai modelljének* segítségével végzünk számításokat.

Példaként nézzük meg, hogyan lehet egy várakozás sort *Markov lánccal* modellezni. A várakozási sor maga is modellje lehet valamilyen hálózati eszköznek, például egy routernek! Egy érkezési-kiszolgálási folyamatot láthatunk az 5.6. ábrán.

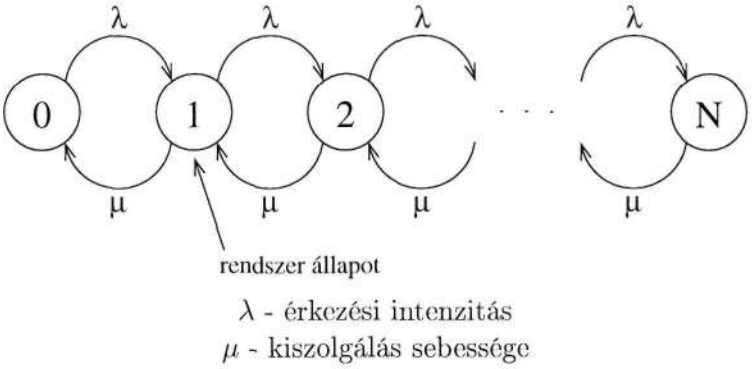


5.6. ábra. Érkezési-kiszolgálási feladat

Az érkezés (például egy forrás alkalmazástól a csomagok érkezése) A paraméterű *Poisson folyamat* szerinti. A kiszolgálás ideje exponenciális eloszlás szerinti u paraméterrel. A rendszer *állapotát* a sorban álló feladatok (csomagok) száma jelenti. Ez minden érkezéskor eggyel nő, és minden kiszolgálás befejezésekor eggyel csökken. Kérdés lehet például, hogy állandósult állapotban mekkora a sorhossz várható értéke, vagy a csomagok hány százaléka veszik el adott (véges) méretű sor esetén.

A *Markov lánc* egy matematikai objektum, itt most elég annyit tudnunk róla, hogy állapotai vannak, amelyeket bizonyos valószínűséggel vesz fel, illetve egyik állapotából egy másik állapotába bizonyos valószínűséggel kerül át.

A feladatunkhoz az 5.7. ábrán látható Markov láncot rendeljük.



5.7. ábra. Markov lánc rendelése a feladathoz

Jelölje p_i annak a valószínűségét, hogy a rendszer az *í.* állapotban van (a sorban várakozó csomagok száma: *i),* formálisan:

$$\Pr\{\textit{a rendszer az i. állapotban van}\} = p_i$$

Állandósult állapotban ugyanakkora valószínűséggel megy át a rendszer a 0-s állapotból az l-es állapotba, mint fordítva:

$$p_0 \cdot \lambda = p_1 \cdot \mu$$

$$\Downarrow$$

$$p_1 = p_0 \frac{\lambda}{\mu}$$

[17] Érdeklődők bővebben olvashatnak róla: [8]

A 1 és 2-es állapot közötti állapot átmenetekre hasonlót felírva kapjuk, hogy:

$$p_2 = p_1 \cdot \frac{\lambda}{\mu} = p_0 \cdot \left(\frac{\lambda}{\mu}\right)^2$$

Általános kifejezéssel:

$$p_n = p_0 \cdot \left(\frac{\lambda}{\mu}\right)^n$$

Elvileg a sorhossz lehet végtelen, ekkor a valószínűségek összegére felírhatjuk, hogy:

$$\sum_{i=0}^{\infty} p_i = 1$$

$$\Downarrow$$

p_0 kiszámítható.

Gyakorlati esetben persze a sorhossz mindig véges, jelölje N. Ha a puffer tele van, és új igény (csomag) érkezik, akkor az el fog veszni. Hasonlóképpen a valószínűségek összege:

$$\sum_{i=0}^{N} p_i = 1$$

$$\Downarrow$$

p_0 ekkor is kiszámítható.

A módszer segítségével megválaszolható kérdések például:

- Mekkora a sorhossz várható értéke?
- Mekkora puffer kell, hogy a veszteség 1 % alatt legyen?

Természetesen az analitikus módszer alkalmazásának is vannak komoly korlátai:

- az analitikus módszer pontatlan
  Mert a modell a rendszerhez képest túl sok leegyszerűsítést, elhanyagolást tartalmaz.

- az analitikus módszer nem számolható végig
  Mert nem létezik zárt alakú megoldás, vagy mert a közelítő megoldás számításigénye sem elégíthető ki.

Kellően nagy komplexitású rendszerek esetén az analitikus módszer alkalmatlan a rendszer viselkedésének kielégítő vizsgálatára.

5.3.4. Szimuláció

Fogalmak

A szimulációt gyakran összetévesztik az emulációval, ezért most mindkettőt definiáljuk:

Definíció: A **szimuláció** számítógép által végrehajtható modellen végzett *kísérlet.*

Definíció: **Emulációról** beszélünk, amikor valamilyen hardvert vagy szoftvert más hardverrel vagy szoftverrel *helyettesítünk.* Fekete dobozként (kívülről nézve) ugyanúgy működik, mint az eredeti, belső működése lehet teljesen más.

Röviden úgy fogalmazhatjuk meg, hogy:

- szimuláció: kísérletezés

- emuláció: üzemszerű használat

Lássunk mindegyikre egy-egy példát is:

- Repülőgép szimulátor: nem tudunk vele repülni, csak a pilóta gyakorolhat rajta.

- Koprocesszor emuláció: a program is kiszámítja a szám négyzetgyökét, csak kicsit lassabban, esetleg más algoritmussal.

Van a szimulációval kapcsolatos másik két fogalom is, amit szintén össze szoktak keverni.

A *verifikáció* a szimulációs modell vagy az eredmények helyességének ellenőrzését jelenti. Tipikus kérdés: „Ez a modell jó?"

A *validáció* annak a vizsgálata, hogy a modell, illetve az eredmények valóban leírják-e a valóságos rendszert, amit vizsgálunk. Tipikus kérdés: „Ez a jó modell?"

Az eredmények érvényességének meghatározása is fontos. Milyen feltételek mellett jellemzik a rendszert az eredmények.

A szimuláció fajtái

Ismerjünk meg néhány fogalmat!

Folytonos idejű rendszer állapota időben folyamatosan változik (például: víz áramlása egy csőben)

Diszkrét idejű rendszer állapota *diszkrét* (egymástól elkülöníthető) időpontokban változik

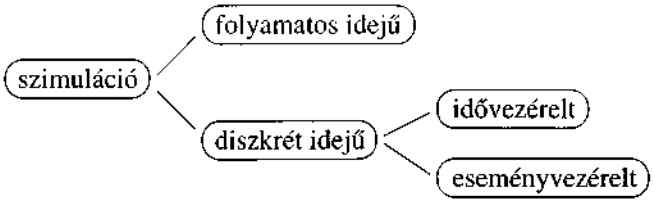
Diszkrét idejű szimuláció A rendszer állapotváltozásai egymástól elkülöníthető időpontokban történnek (vagy úgy vesszük figyelembe őket).

Eseményvezérelt szimuláció A szimuláció végrehajtása során csak azokkal az időpontokkal foglalkozunk, amelyekben a rendszer állapotát megváltoztató *esemény* történik.

Idővezérelt szimuláció A modellben az idő mindig valamilyen előre meghatározott At értékkel nő, az így adódó

időpontokban meg kell vizsgálni, hogy az előző időpont óta történt-e valami a rendszerben.

A diszkrét idejű szimulációt angolul úgy nevezik, hogy: *Discrete Event Simulatíon* (rövidítve: DES).



5.8. ábra. Szimuláció fajtái

Szimuláció esetén különféle idő elnevezéseket szoktak használni:

- A **virtuális idő** *(virtual time)* a vizsgált rendszer modelljében használt idő, éppen ezért kifejezőbb (de ritkábban használt) a *modellidő.*

  A **végrehajtási idő** a szimulációt végrehajtó gép valós idő órája által mért idő, ezt angolul úgy nevezik, hogy *watt clock time* (falióra idő).

Eseményvezérelt diszkrét idejű szimuláció

Az eseményvezérelt diszkrét idejű szimuláció működésében fontos szerepet játszik a *jövőbeli események halmaza* (Future Event Set - FES). Amint a neve is jelzi, a szimuláció során ebben tároljuk azokat az eseményeket, amelyekről már tudjuk, hogy be fognak következni. Az eseményeket időbélyeggel látjuk el; az időbélyeg megmutatja, hogy mely virtuális időpontban fog az esemény bekövetkezni. Például:

időbélyeg	esemény leírása
0.012	keret adása befejeződik
0.018	keret eleje a vevőhöz megérkezik

A szimuláció kezdetén bekerül néhány esemény a FES-be, majd a szimuláció során mindig a legkisebb időbélyegű eseményt vesszük ki (és ki is töröljük onnan). Az esemény kivétele után először is beállítjuk a modell óráját az esemény időbélyegének az értékére, majd „eljátsszuk" az esemény bekövetkezését. Ennek során (az esemény típusától függően) különböző állapotváltozók értéke megváltozhat, illetve újabb események kerülhetnek be a FES-be. Az események feldolgozásának ciklusa természetesen szükségszerűen befejeződik, ha a FES kiürül, de más feltételek miatt is megállhatunk. Az eseményvezérelt diszkrét idejű szimuláció algoritmusát az alábbiakban formálisan is megadjuk, az algoritmusban a „MOST" a modell óráját jelenti.

```
Inicializálás, azaz bizonyos események felidőzítése
  a FES-be;
REPEAT
  Legkisebb időbélyegű esemény kivétele (és törlése)
    a FES-ből;
  MOST := a kivett esemény időbélyege;
  Esemény feldolgozása, eközben esetleg újabb
    esemény(ek) generálása (és behelyezése a
    FES-be);
UNTIL (elfogytak az események) v (MOST > mint egy
      beolvasott határ) v (egyéb ok miatt
      meg kell állni);
```

Az algoritmus végrehajtása során az új események természetesen csak az aktuális modell időnél (MOST) nem kisebb időbélyeget kaphatnak.

Ezt az algoritmust használják a kommunikációs rendszerek teljesítőképesség-vizsgálatára alkalmazott eseményvezérelt szimulátorok, mint például az OPNET Modeler [31], az OMNeT++ [30] és az ImiNet [26].

Teljesítőképesség-vizsgálat szimulációval

Mit kell tennünk, ha egy kommunikációs rendszert szimulációval szeretnénk vizsgálni?

Először is elkészítjük a rendszer modelljét az alkalmazott szimulátor eszközkészletének felhasználásával. Lehet, hogy a rendszerünk elemeinek a modelljét megtaláljuk a szimulátor elemkönyvtárában, de lehet, hogy magunknak kell megírnunk. A szimulátor által nyújtott módon leírjuk a rendszer topológiáját. (Lehet, hogy grafikus felületen összerakhatjuk a hálózatot, de az is lehet, hogy valamilyen szöveges topológia leíró nyelvet kell használnunk.)

Modelleznünk kell a hálózatot használó alkalmazásokat, pontosabban azok forgalmát is!

Meghatározzuk, hogy milyen feltételek mellett szeretnénk elvégezni a vizsgálatot, és ennek megfelelően állítjuk be a modell paramétereit.

Meghatározzuk, hogy milyen jellemzők érdekesek számunkra, és ennek megfelelően állítjuk be a statisztikagyűjtést.

Végrehajtjuk a szimulációt, közben a szimulátor statisztikát gyűjt az általunk kívánt jellemzőkről.

Kiértékeljük a gyűjtött statisztikákat. Ennek alapján dönthetünk úgy, hogy a modellt vagy a terhelést módosítjuk, illetve szükség lehet további jellemzők megfigyelésére is. Ilyen esetekben újabb szimulációt végzünk.

Amikor végre választ kaptunk a kérdéseinkre, akkor az eredményeinket olyan formára hozzuk, hogy az mások számára is emészthető legyen. Ennek részleteivel mélyebben is foglalkozunk!

5.3.5. Mérési eredmények kezelése

Függetlenül attól, hogy a mérési eredményeket valóságos rendszeren vagy egy működtetett modellen való méréssel nyertük, fontos azok helyes kezelése. Először is meg kell fontolnunk a szükséges mérések számát. Alapelv, hogy *egy mérés, nem mérési* A szükséges mérések száma természetesen függ az adott feladattól.

Szórás feltüntetésének fontossága

Mérési eredményeinkből átlagot és szórást számolunk. Mivel a szórás számításakor nem ismerjük az elméleti várható értéket, ezért a tapasztalati szórás képletét használjuk, ahol a nevezőben a mérések számánál 1-gyel kisebb szám áll.

Nézzünk meg egy egyszerű számpéldát arra, hogy miért fontos a szórás feltüntetése! Legyen két mennyiség: „A" és „B". Tegyük fel, hogy mindkettőt megmértük mondjuk 10-szer, és az átlagra „A" esetén azt kaptuk, hogy 8,2; „B" esetén pedig azt, hogy 8,7. Mit mondhatunk „A" és „B" viszonyáról? Nem igazán sokat! Ha viszont tudjuk, hogy például a szórás mindkét mennyiségnél 0,1 volt, akkor már mondhatjuk, hogy elég nagy valószínűséggel: $A < B$. Ellenben, ha például a szórás mindkét mennyiségnél 2 volt, akkor az esetek jelentős részében nem áll fenn, hogy $A < B$.

Eredmények pontossága

Az eredmények túlzott pontossággal való feltüntetése helytelen. Tegyük fel, hogy egy mérés eredményeként azt kaptuk, hogy egy mennyiség mért értékeinek átlaga: 8,4786; szórása pedig: 1,2345. Ebben az esetben bátran kerekítsük a várható értéket 8,5-re, a szórást pedig 1,2-re!

5.4. Eredmények megjelenítése

Függetlenül attól, hogy az eredményeket milyen módon kaptuk (például méréssel, számítással vagy akár szimulációval), gyakran szükségünk van arra, hogy eredményeinket másoknak bemutassuk. Az alábbiakban közölt szempontok hasznosak lehetnek mind papír alapú művek, mind szóbeli elő'adásokat támogató, projektorral vetített anyagok készítésénél.

A következő'kben anyagok teljes átvételével is támaszkodunk [32] forrásra, amely viszont felhasználta a következőt: [10].

5.4.1. Általános szempontok

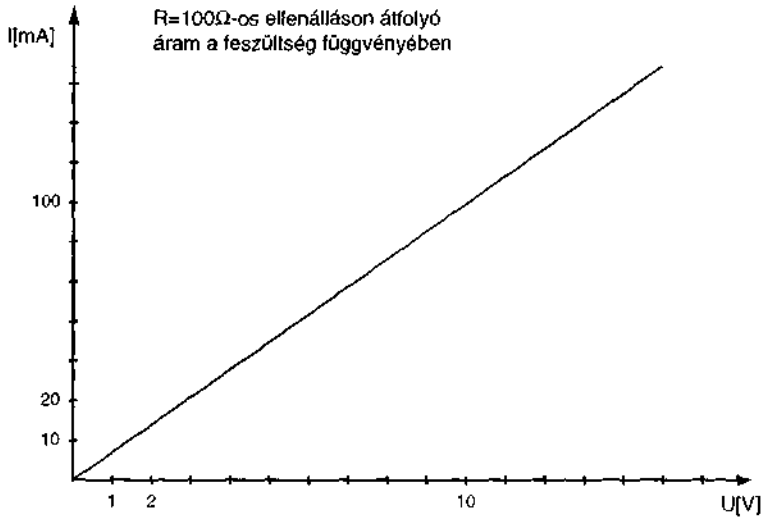
Egy ábrának vannak *alapvető alaki kellékei,* mint például grafikonok esetén:

- cím (képaláírás): mit látunk az ábrán
- tengelyfeliratok, mértékegységek
- tengelyeken egységek bejelölése; derüljön ki, hogy lineáris vagy logaritmikus skálát használunk
- ha több jellemzőt is ábrázolunk: melyik grafikon mit jelent
- hasznos a *nevezetes értékek* (min/max hely, inflexiós pont, stb.) koordinátáinak feltüntetése.

Példaként nézzünk meg egy grafikont, ahol szerepelnek a szükséges alaki kellékek: 5.9. ábra.

Néhány további általános szempont, jó tanács:

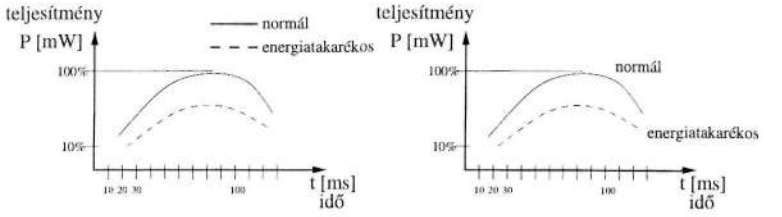
- A grafikus megjelenítés jobban áttekinthető, mint a táblázatos, de nem helyettesíti azt.



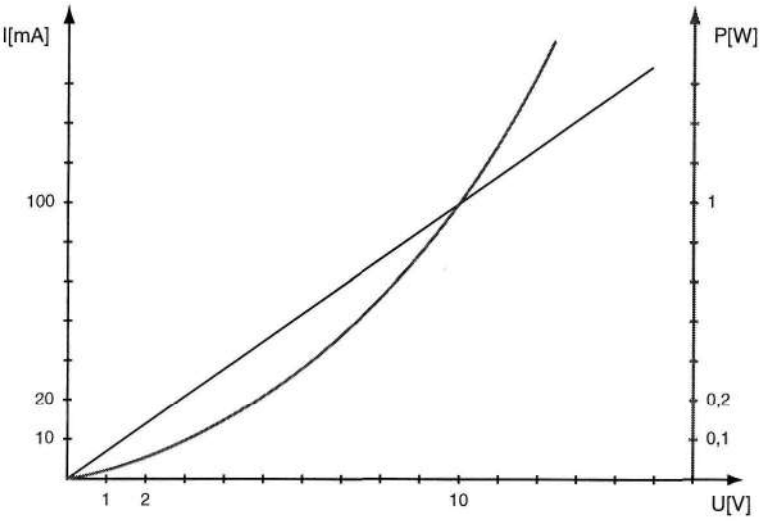
5.9. ábra. Szemléltető ábra

- Az ábrába a maximális mennyiségű információt vigyük be!
- Ugyanakkor az olvasótól minimális erőfeszítést követeljen meg az eredmények áttekintése!
- Az eredmény érvényességére vonatkozó adatok is ott szerepeljenek (az ábrán, vagy az ábra címében)!
- A grafikonok jelentését külön magyarázzuk meg, vagy még jobb, ha az ábrába írjuk: 5.10. ábra!
- Adott független változó függvényében akár eltérő függő változókat is ábrázolhatunk egy ábrában, de ne legyen az y tengelyen sok skála, legfeljebb kettő! (5.11. ábra)

  Ne kössük össze a nem folytonos dolgokat! (5.12. ábra)



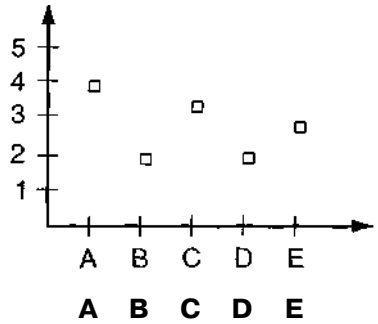
5.10. ábra. Jelmagyarázat és ami még jobb



5.11. ábra. Az y tengelyen legfeljebb két skála legyen!

Érdemes színeket használni, ha erre lehetőség van!

Színválasztás: háttér és szöveg (rajz) fényereje erősen eltérő legyen! Például sötét háttéren világos ábra vagy fordítva. (Különben projektoros kivetítésnél nem lesz jól látható.)



5.12. ábra. Ne kössünk össze nem folytonos dolgokat!

- Összeillő színeket válasszunk! (Színkörbe szabályos sokszöget **írunk.)**
- Ügyeljünk arra, hogy fekete-fehér nyomtatás / fénymásolás után is értelmezhető maradjon a munkánk! (Például eltérő vonalvastagság, szaggatott vonal, stb.)

5.4.2. Ábrázolási módok

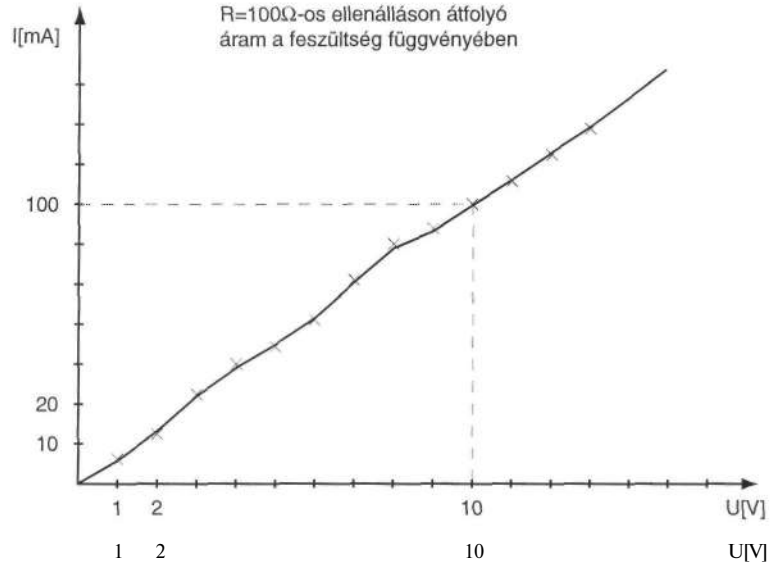
Az ábrázolandó mennyiség jellegétől függően más-más módot választunk a megjelenítésre. Ezek közül mutatunk be néhányat.

Függvény grafikonja

Amennyiben az ábrázolni kívánt függvény értelmezési tartománya és értékkészlete (a vizsgált tartományban) *folytonos,* akkor az 5.9. ábra szerint ábrázolhatjuk. Ha a függvény képlet formájában adott, az ábrázolás nem okoz gondot. Ha viszont mérési eredményeket ábrázolunk, akkor csak véges, esetleg kis számú (például 10db) független változó értékre van mért értékünk. Ebben az esetben alkalmazhatunk törtvonalas közelítést, vagy illeszthetünk rá görbét. Ilyenkor hasznos lehet a mérési

pontokban kapott értékek megkülönböztetése a többi értéktől. Erre látunk példát az 5.13. ábrán.

További hasznos eszköz lehet, ha a mért értékeknél feltüntetünk valamilyen (például 90%-os) konfidencia intervallumot is, illetve amennyiben (például az eloszlás ismeretének hiányában) ezt nem tudjuk megtenni, akkor értelmes lehet a mérési pontokba olyan függőleges szakaszt rajzolni, aminek a közepe a mért (átlag)értékre illeszkedik, alsó végénél a függő változó értéke átlag-szórás, felső végénél pedig átlag+szórás. Ehhez természetesen minden mérési pontban (adott független változó érték mellett) megfelelő számú mérést kell végeznünk.



1 2 10 U[V]

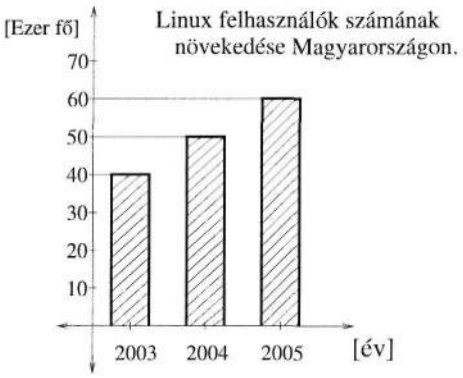
5.13. ábra. Folytonos függvény közelítése mért értékek alapján

Amennyiben a függvény értelmezési tartománya nem folytonos, akkor a mért értékeket ábrázoló pontokat nem szabad összekötni, mert nincs valóságos tartalmuk, amint az 5.12. ábrán láttuk. Erre mutatunk az alábbiakban egy másik megoldást

is, ami egy speciális esetben használható.

Oszlopdiagram

Az oszlopdiagramot olyan függvényeknél célszerű alkalmazni, amikor az értelmezési tartomány nem folytonos, a független változó egymástól egyenlő távolságban található, viszonylag kis számú értéket vesz fel. (5.14. ábra)



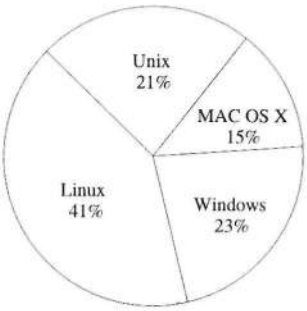
5.14. ábra. Példa oszlopdiagramra (nem valós adatok)

Kördiagram

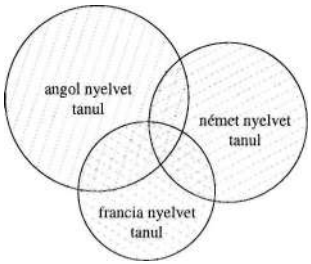
A kördiagramot százalékos megoszlás ábrázolására használhatjuk. Az egész kör jelenti a 100%-ot. (5.15.)

Venn-diagram

Halmazok és a köztük levő relációk ábrázolására használhatjuk a Venn-diagramot. (5.16. ábra)



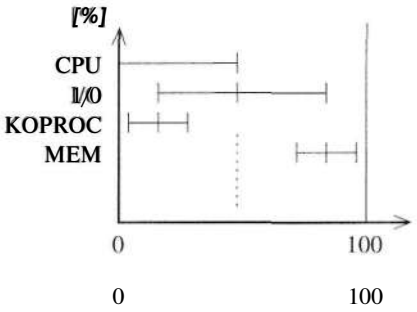
5.15. ábra. A feltelepített operációs rendszerek megoszlása 2015-ben Magyarországon, (nem valós adatok)



5.16. ábra. Példa halmazok közötti relációk ábrázolására (nem valós adatok)

Gantt-ábra

A Gantt-ábra segítségével ábrázolhatjuk erőforrások kihasználtságát az időbeli átlapolódások figyelembe vételével. (5.17. ábra)



5.17. ábra. Gantt-ábra

Kiviat-gráf

A Kiviat-gráfot[18] százalékos értékek ábrázolására szokták alkalmazni. Az ábrát úgy készítjük el, hogy egy körbe (egyenletes elosztásban) berajzolunk a vizsgált jellemzők számának megfelelő számú sugarat, majd ezekre a kör középpontjától kiindulva bejelöljük az egyes jellemzők százalékos értékét (0%: kör középpontja, 100%: körvonal), majd a bejelölt pontokat összekötjük a szomszédaikkal. Az összekötés célja folthatás elérése.

Megtehetjük, hogy a kör mentén váltakozva HB (Higher is Better - minél nagyobb annál jobb) és LB (Lower is Better - minél kisebb, annál jobb) értékeket helyezünk el. Ekkor az optimális ábra egy sokágú csillag lesz.

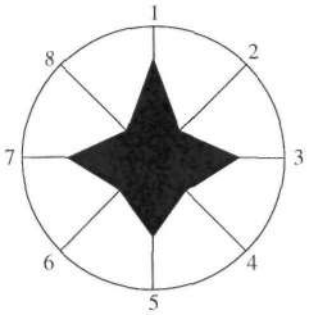
Egymás mellé helyezhetünk korrelált, illetve nem korrelált mennyiségeket is.

Az 5.18. ábrán egy Kiviat-gráfot látunk, ahol az egyes mennyiségek jelentése a következő:

1. CPU foglalt

2. csak a CPU foglalt

[18] A Kiviat-gráfot radar-gráfnak is szokták nevezni.



5.18. ábra. Példa Kiviat-gráfra

3. CPU és csatorna átlapolt

4. csak csatorna foglalt

5. bármely csatorna foglalt

6. CPU vár

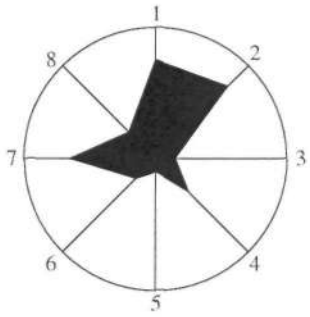
7. CPU felhasználói programot hajt végre

8. CPU supervisor állapotban

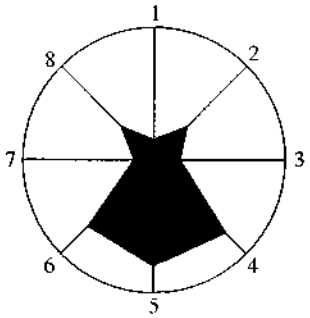
A folt alakjából nagyon gyorsan megállapíthatók a rendszer bizonyos tulajdonságai. Esetünkben például van néhány tipikus alak, például:

- főleg a CPU-t használja (5.19. ábra)
- főleg a csatornát használja (5.20. ábra)

A diagramot több szintűre is lehet rajzolni. Például memória méretet növelem.(5.21. ábra)



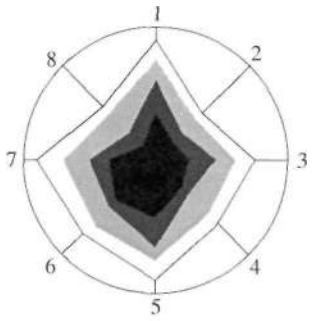
5.19. ábra. A CPU használata dominál



5.20. ábra. A csatorna használata dominál

Hisztogram

Eloszlásokat szemléletesen jellemezhetünk a valószínűségi sűrűségfüggvényükkel. Amennyiben az eredményeink megfigyelésből (teljesen mindegy, hogy egy valóságos rendszeren való mérésekkel vagy szimulációból) származnak, tipikus, hogy az eloszlás jellemzésére egy adott számosságú minta áll rendelkezésünkre. A minta elemeinek egyenkénti megjelenítésénél sok esetben jobb megoldás, ha a mintákból hisztogramot készítünk.



5.21. ábra. Többszintű Kiviat-gráf.

Hisztogram készítéséhez a megfigyelések lehetséges értékkészletét diszjunkt, és az értékkészletet teljesen lefedő intervallumokra bontjuk. Ezeket az intervallumokat *celláknak* nevezzük. A legegyszerűbb esetben a cellák szélessége azonos. Ez nem mindig célszerű választás, sokszor jobb (az eloszlást pontosabban jellemző) eredményt kaphatunk, ha az eloszlás közelítő ismerete alapján a cellahatárokat úgy választjuk meg, hogy az egyes cellákba eső megfigyelések száma közel egyenlő legyen. A hisztogram celláit oszlopokkal (téglalapokkal) ábrázoljuk úgy, hogy az oszlop alapja a hisztogram celláját jelentő szakasz (a cella szélességét jelölje *d),* magassága pedig *h.* Az *i.* cella esetén:

$$h_i \equiv \frac{n_i}{d_i}$$

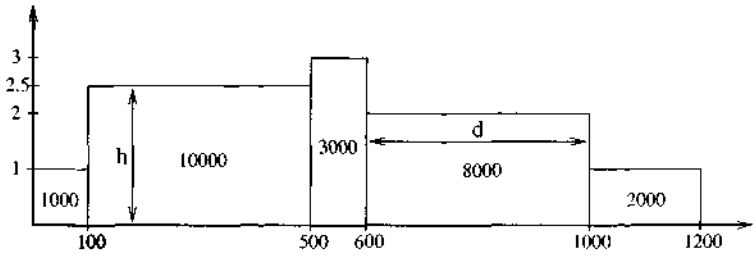
ahol:

$h_i \equiv$ az i. cellát ábrázoló oszlop magassága

$n_i \equiv$ az i. cellába eső elemek száma

$d_i \equiv$ az i. cella szélessége

Lássunk egy példát: 5.22. ábra.



5.22. ábra. Példa hisztogramra

Ha egy hisztogramot *normálunk,* azaz a cellák magasságát elosztjuk az összes megfigyelés számával, akkor hisztogram területe 1 lesz: megkapjuk a valószínűségi sűrűségfüggvény[19] közelítését.

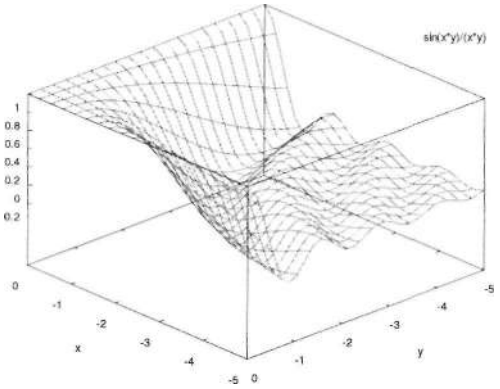
Két változós függvény ábrázolása felülettel

Amennyiben egy két változós függvényt szeretnénk ábrázolni, azt megtehetjük például úgy, hogy a független változóit egy négyzetháló mentén tekintjük, és ezekre ábrázoljuk a függvény értékét: Ilyenre látunk példát az 5.23. ábrán.

5.4.3. Trükkök

Az alábbiakban bemutatott trükkök egyrészt használhatók a mondanivalónk hangsúlyozására, másrészt viszont vissza is lehet velük élni; alkalmasak az olvasó megtévesztésére is! Ez utóbbit semmiképpen sem javasoljuk, hanem inkább arra bátorítjuk e jegyzet olvasóit, hogy mérnöki munkájuk eredményének bemutatása során legyenek becsületesek! Mégis hasznos ezeknek

[19] *f(x),* angolul: PDF - Probability Density Function, az integrálja pedig megadja az eloszlás függvényt *F(x),* angolul: CDF - Cumulative Density Function



5.23. ábra. Példa felület ábrázolására

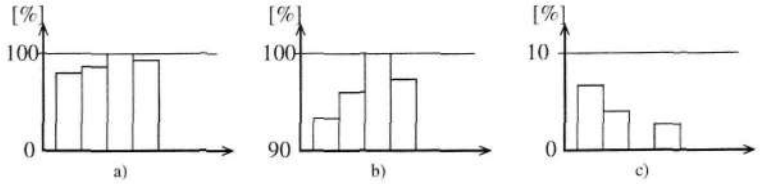
a trükköknek az ismerete, mert sajnos mind a mérnöki munka során, mind a hétköznapi életben (például reklámok, politikai hirdetések) találkozhatunk ezekkel a trükkökkel; ismerjük fel ó'ket, és ne hagyjuk magunkat becsapni!

- Ha azt szeretnénk hangsúlyozni, hogy minden majdnem 100%-os, akkor az 5.24. ábra „a" részében látható módon a skála 0%-tól 100%-ig terjedjen, ha a különbséget kívánjuk hangsúlyozni, akkor az ábra „b" részében látható módon a skálát nem 0%-tól célszerű indítani. Így a különbségek a valóságosnál nagyobbnak tűnnek![20] A különbséget hangsúlyozza az 5.24. ábra „c" része is, ahol azt ábrázoltuk, hogy mennyi kellene a 100%-hoz.[21]

- Ha összehasonításnál egyenlő szélességű oszlopok helyett

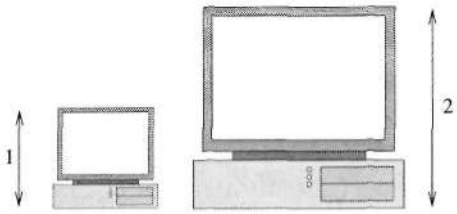
[20] Ez teljesen becsületes is lehet: alkalmas kicsi, de jelentős különbségek kimutatására, de felhasználható lényegtelen különbségek felnagyítására is. Azt, hogy a különbség lényeges vagy sem, a műszaki életben sok esetben el lehet dönteni, a hétköznapi életben ez sokkal nehezebb kérdés ...

[21] Ennek is lehet érdemi jelentése, például: mennyi a szabad kapacitás.



5.24. ábra. Különbségek elfedése vagy kinagyítása

(amelyeknek a magassága lenne arányos az ábrázolandó mennyiséggel) az ábrázolandó mennyiséggel lineáris méretben arányos, két dimenziós ábrákat használunk; az már kifejezetten becsapás, ugyanis a szemünk területet érzékel! így például egy kétszeres arány négyszeresnek látszik: 5.25 ábra.

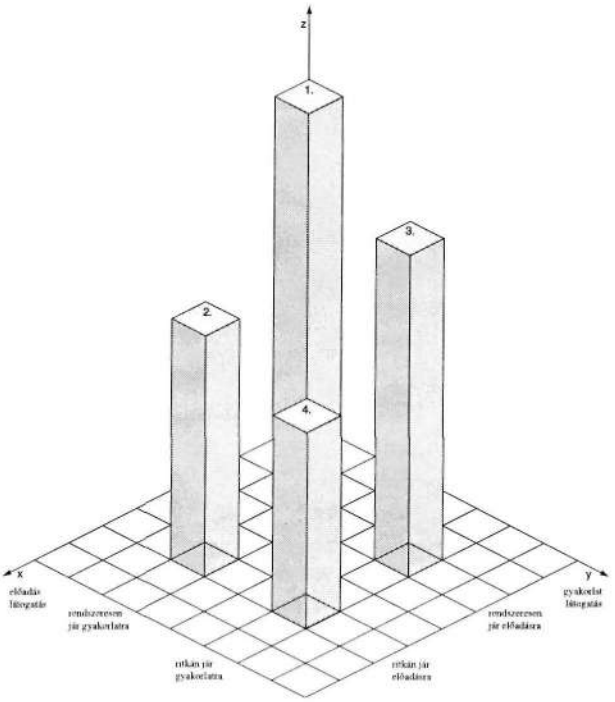


5.25. ábra. A szemünk területet érzékel

A relatív jellemzők használata kifejezetten alkalmas a becsapásra. (Mottó: „A statisztika a számok segítségével elkövetett hazudozás.") Például:

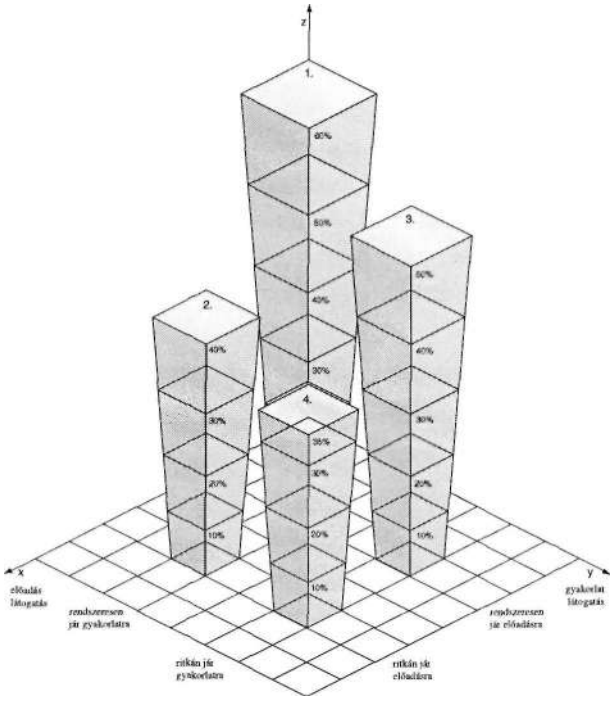
- kerekek száma / ütemek száma —> a kétütemű Trabant a legjobb autó
- egy főre jutó űrhajósok száma —> Magyarország az űrhajózásban az élen áll

- Többdimenziós ábrán (két független és egy függő változó esetén) például az oszlopok elhelyezkedése is kihasználható: 5.26. ábra. Az ábrán a Számítógép-hálózatok tantárgyból tett első vizsga sikerességének valószínűsége látható a hallgatók előadás és gyakorlat látogatási hajlandóságának függvényében. Első ránézésre azok a hallgatók, akik mindkét helyre becsülettel járnak, toronymagasan kiugranak a többiek közül. Trükk: az ő oszlopuk eleve magasabban kezdődik!



5.26. ábra. Az első vizsga sikerességének valószínűsége a rendszeres előadásra és a gyakorlatra járás függvényében.

• Az 5.27. ábrán a perspektivikus ábrázolást használjuk a (túl)hangsúlyozásra. A magasabb oszlopok így szélesebbek is (és a szemünk területet érzékel), ráadásul a függő változó tengelyének a skálája nem lineáris: a magasabb oszlopok még magasabbnak látszanak!



5.27. ábra. Perspektíva kihasználása

A. Függelék

Unix bevezető

A tárgy gyakorlatain Linux operációs rendszert használunk, ehhez szükséges az alapvető Unix[1] parancsok ismerete. A gyakorlatokon használt Debian GNU/Linux egy GPL licensz [22] alatt használható *szabad*[2] operációs rendszer.

Bővebb ismertetésre e jegyzet keretei nem nyújtanak lehetőséget, ezért az érdeklődő hallgatóknak javasoljuk, a távközlés-informatika szakirányos hallgatók pedig tekintsék kötelezőnek az alapvető Unix felhasználói ismereteket nyújtó [16] vagy más hasonló forrás tanulmányozását. További, kifejezetten Linux specifikus ismereteket nyújtó ajánlott forrás: [12].

A.1. Alapismeretek

A Unix egy *többfeladatos* (multitasking) operációs rendszer, ami azt jelenti, hogy egyszerre több programot is futtathatunk.[3]

[1] A UNIX (csupa nagy betűvel) bejegyzett márkanév, *Unixnak* nevezzük az összes UNIX-szerű operációs rendszert.

[2] A szabad szoftverekről bővebben: [21]

[3] A programok nem biztos, hogy tényleg egyidejűleg futnak, lehet, hogy csupán az operációs rendszer váltakozva futtatja őket.

Emiatt szükséges a programok esetleges káros (hibás vagy rossz szándékú) működésétől védenünk a többi programot, az operációs rendszert, sőt a hardvert is. A programok csak a rendelkezésükre bocsátott memória területet használhatják, minden szolgáltatást operációs rendszer (kernel) függvényhívásokon keresztül vehetnek igénybe, közvetlenül a hardverhez nem férhetnek hozzá. Ehhez természetesen hardver támogatás szükséges, a processzornak legalább két privilegizációs szintet kell támogatnia: a kernel privilegizált módban fut, míg a felhasználók programjai az alap szinten futnak.

A Unix *többfelhasználós* (multi-user) operációs rendszer is, ami azt jelenti, hogy egy gépen több felhasználó is dolgozhat. Az egyes felhasználók adatait is védeni kell egymástól, erre szolgálnak a fájlrendszerrel kapcsolatos védelmi mechanizmusok. A rendszer adminisztrálásához szükség van olyan felhasználóra (super user) aki mindenhez hozzáfér. Unix alatt a super usert rootnak hívjuk.

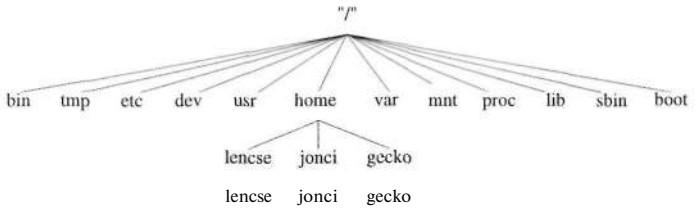
Ahhoz, hogy valaki egy Unix rendszeren dolgozhasson, először be kell jelentkeznie a rendszerbe. Bejelentkezni valamely előzőleg létrehozott *felhasználói néven* (login name) lehet, a felhasználó a *jelszó* (password) megadásával bizonyítja a személyazonosságát.

A.2. Fájlrendszer

A Unix a DOS/Windows rendszerekkel ellentétben nem használ meghajtó jelöléseket (mint „A:" vagy „C:"), hanem egyetlen fájlrendszer megfelelő pontjaira (alkönyvtár nevek alá) kapcsolja fel az egyes köteteket.

A. 2.1. Könyvtárszerkezet

A könyvtár rendszer kiindulási pontját *(root* vagy gyökér könyvtár) a "/" jellel jelöljük. Az A.l. ábra egy *tipikus Unix könyvtárszerkezetet*[4] mutat.



A.l. ábra. Tipikus Unix könyvtárszerkezet

A szokásosan alkalmazott alkönyvtárak közül megadjuk néhánynak a funkcióját.

/bin Az alapvető', minden felhasználó számára fontos parancsok (például: ls, cp, cat) találhatók ebben.

/boot Az operációs rendszer indulásával kapcsolatos fájlok találhatók benne.

/dev Az eszközfájlokat tartalmazza.

/etc A rendszer adminisztrálásával kapcsolatos dolgokat (például konfigurációs fájlokat) tartalmazza.

/home A rendszerre felvett felhasználók könyvtárait tartalmazza.

/lib A dinamikusan szerkesztett programkönyvtárak helye.

[4] A különféle Unix rendszerekben hasonló alap könyvtárszerkezetet használnak, de vannak eltérések is. További információ: [35]

/mnt Ide szokás felkapcsolni az ideiglenesen felcsatolt köteteket.

/proc Virtuális fájlrendszer: fájlként láthatjuk a kernel belső dolgait.

/root A rendszergazda home könyvtára.

/sbin A rendszer adminisztrálására szolgáló parancsokat (például: fdisk, init) tartalmaz.

/tmp Bárki által írható könyvtár, az ideiglenes fájlok tárolására szolgál.

/usr További alkönyvtárakban a felhasználói programokat tartalmazza.

/var További alkönyvtárakban olyan dolgokat tartalmaz, amelyek tipikusan változni szoktak, például nyomtatási sorok, levelesládák, naplófájlok, stb.

A különböző eltávolítható média eszközöket (például: optikai vagy USB flash drive) újabban nem a /mnt, hanem a /media könyvtár alá szokták felkapcsolni.

Hétféle könyvtárbejegyzés lehetséges:

directory könyvtár, amely a könyvtárbejegyzés típusok bármelyikéből tartalmazhat bejegyzéseket

file normál állomány/fájl, amely adatoknak egy névvel azonosított halmaza

block device speciális eszközfájl, a Unix minden hardver eszközt (perifériát) device-ként azonosít, ezek között vannak blokkos átvitelt használók

character device karakterenkénti átvitelt használó periféria azonosítására szolgáló speciális eszközfájl

symbolic link szimbolikus link, amely egy másik könyvtárbejegyzésre mutat

socket socket, lásd: 5.2.

pipe névvel rendelkező csővezeték (named pipe)

Néhány jellegzetes Linux periféria név :

/dev/hda Az első (primary master) IDE-s merevlemez egységet jelöli, a továbbiak: hdb, hdc, stb.

/dev/fdO Az első hajlékonylemez-meghajtót jelöli. A következő: fd1.

/dev/sda Az első SCSI merevlemez egységet jelöli, a továbbiak: sdb, sdc, stb. SCSI emuláció esetén ilyen nevet kapnak például az USB flash drive-ok, SATA merevlemezek is.

/dev/ttyO ttyl, tty2, stb. a billentyűzet neve.

/dev/psaux A ps2-es egér neve.

/dev/null Végtelen kapacitású nyelő: nyom nélkül eltünteti a bele irányított kimenetet.

A.2.2. Fájlrendszerrel kapcsolatos parancsok

Az alapvető Linux felhasználói jártasság megszerzése a laborgyakorlatokon történik, itt csak egy rövid összefoglalást adunk. A leírásban a „<" és „>" jelek úgynevezett metanyelvi zárójelek, a közöttük levő szöveg azt jelenti, hogy oda mit kell **írni** a parancs kiadásakor. A „<" és „>" jeleket természetesen nem kell, sőt nem szabad begépelni, mert akkor egészen mást jelentenek a parancsok!

[5]Felhívjuk az olvasó figyelmét, hogy ezek kifejezetten a Linuxra jellemzőek, más Unix rendszerek más konvenciót használnak!

Is könyvtár listázása

Is -a könyvtár listázása a rejtett fájlokkal együtt

Is -l könyvtár listázása, az attribútumokat is kiírja

cd <directory> belépés a megadott könyvtárba

cd A parancsot argumentum nélkül kiadva a felhasználó saját home könyvtárába jut.

pwd aktuális *munkakönyvtár* (working directory) nevének kiírása

mkdir <directory> könyvtár létrehozása

rmdir <directory> könyvtár törlése, csak üres könyvtár esetén használható

rm <fájlnév vagy fájlnevek> fájl(ok) törlése

rm -r <directory> könyvtár rekurzív törlése a benne lévő fájlokkal, alkönyvtárakkal együtt

cat fájl tartalmának kiírása

cp <forrás fájl> <cél fájl> fájl másolása

cp <fbrrás fájlok> <cél könyvtár> fájlok másolása

mv <forrás> <cél> fájl(ok), könyvtár(ak) mozgatása

du lemezhasználat: könyvtár és alkönyvtárainak helyfoglalását adja meg

df lemezhasználat: egyes kötetek kihasználtságát (teljes, szabad, foglalt terület és *i-node-ok)* adja meg

quota a felhasználó által használható és felhasznált terület kijelzése

A.2.3. Jogosultságok és kezelésük

Minden fájlra és könyvtárra a jogok három csoportja vonatkozik. A jogokat kilenc biten tároljuk.[6] Az első három bit adja meg a *tulajdonosra* (owner) vonatkozókat, a negyediktől a hatodikig a fájl *csoporttulajdonosaként* beállított *csoportba*[7] (group) tartozó felhasználók jogait, az utolsó három pedig az összes többi felhasználó (world) jogait adja meg. Egy fájlnál a bithármasok rendre meghatározzák az írásra, olvasásra és a végrehajtásra való jogokat (read, write, execute: "rwx"). Egy könyvtárnál pedig meghatározzák a listázásra, módosításra (benne fájlok, könyvtárak létrehozása, törlése) és a navigálásra (a benne lévő fájlok, könyvtárak elérése) való jogokat. Tekintsük például egy fájl esetén az A.1. táblázatban megadott jogokat. Ezen jogok alapján a tulajdonos jogosult a fájlt írni és olvasni, a csoporttulajdonosként megadott csoportba tartozó felhasználók jogosultak a fájlt olvasni, ezen kívül senki másnak semmi más joga nincs rá.

szimbolikus	r	w	x	r	w	x	r	w	x
bináris	1	1	0	1	0	0	0	0	0
oktális		6			4			0	

A.1. táblázat. Példa jogosultságokra

A jogosultságok beállításával kapcsolatos alapvető Unix parancsok:

chmod <jogok> <fájlnév> jogok beállítása (A jogokat oktálisan adjuk meg.)

[6] Valójában ennél többön, de az meghaladja e bevezető kereteit.

[7] A Unixban a felhasználókat csoportokba soroljuk, minden felhasználónak van egy elsődleges csoportja, ezen kívül tagja lehet még további csoportoknak is.

chown <username> <fájl vagy könyvtár neve> tulajdonos megadása.

chgrp <group> <fájl vagy könyvtár neve> csoporttulajdonos megadása

A.3. Hálózat kezelése

Alapvető' Unix parancsok:

ifconfig Argumentum nélkül az aktív hálózati interfészeket listázza ki a legfontosabb adataikkal együtt. Paraméterezését egy példával mutatjuk be (külön).

route Argumentum nélkül kiírja a kernel routing táblázatának bejegyzéseit.

route add default gw <default gateway> Beállítja az alapértelmezett átjárót.[8]

netstat Részletes információt ad a fennálló (létrehozás vagy lebontás alatt álló) hálózati kapcsolatokról.

A **ping** és a **traceroute** parancsokat már korábban (3.4.3) ismertettük.

A laborgyakorlatokon használt gépekben két-két Ethernet hálózati kártya található, ezeket a Linux ethO-val és ethl-gyel jelöli. Most példaként az ethO interfészt fogjuk beállítani. Legyen az IP címe: 192.168.100.41, a netmask: 255.255.255.224, az alapértelmezett átjáró: 192.168.100.33

```
ifconfig ethO 192.168.100.41 \
netmask 255.255.255.224 \
```

[8] Az add <net> gw <gateway> forma Linux specifikus, máshol a gw szócska nem kell!

```
broadcast 192.168.100.63 up
route add default gw 192.168.100.33
```

Figyelem! A fentiekben összesen két (!) parancsot adtunk ki, ugyanis a sor végi backslash („\") azt jelenti, hogy a parancs a következő sorban folytatódik! Az első paranccsal értelemszerűen beállítottuk a gépünk IP címét és hálózati maszkját, valamit megadtuk a broadcast címet (az ifconfig igényli), végül aktivizáltuk az interfészt az up-pal. A második paranccsal beállítottuk az alapértelmezett átjárót.

Ezek után máris tesztelhetjük gépünk hálózati kapcsolatát a ping paranccsal.

Teljes értékű használathoz szükségünk van még legalább egy névkiszolgáló IP címének beállításához. Ha a névkiszolgáló a 193.224.130.161 IP címen érhető el, ezt beállíthatjuk az alábbi paranccsal:

```
echo "nameserver 193.224.130.161" » /etc/resolv.conf
```

A.4. Programfejlesztés

Alapvető Unix parancsok:

gcc <forrás fájl neve> C vagy C++ forráskód lefordítása, az eredmény out" fájlba kerül.

gcc <forrás fájl neve> -o <program neve> C / C++ forráskód fordítása úgy, hogy a program neve a megadott név legyen.

gcc <forrás fájl neve> -g C / C++ forráskód fordítása úgy, hogy a program nyomon követhető legyen gdb-vel, az eredmény az „a.out" fájlba kerül, de kombinálható az előzővel.

gdb <program neve> Program nyomkövetése gdb-vel.

Nyomkövetés *gdb-vel*

l <sor száma> forráskód sorainak listázása

b <sor száma> töréspont megadása

d <töréspont száma> töréspont törlése

r futtatás

c futás folytatása

n egy lépés végrehajtása (a függvényeket egy utasításnak tekinti, azaz egy lépésben végrehajtja)

s egy lépés végrehajtása (függvényben csak egyet lép)

p <változó neve> a változó értékének kiírása

q kilépés

set args <1. argumentum> <2. argumentum> a program parancssorbeli paramétereinek megadása

A.5. Egyéb szükséges Unix parancsok

echo <argumentum> Kiírja az argumentumát, és az esetleg benne szereplő' dollár jelet követő' környezeti változó értékét. (például: echo $TERM)

ps aux Kilistázza a futó programokat. Unix alatt minden elindított program egy processzként fut, és kap egy azonosító számot (process ID), a programra ezzel számmal tudunk hivatkozni a későbbiekben.

kill <process ID> Leállítja az adott programot, pontosabban egy TERM szignált küld, amit a program kezel, esetleg nem hajlandó a futást befejezni.

kill -9 <process ID> Mindenképpen leállítja a programot. (KILL szignál)

kill -SEGV <process ID> Leállítja a programot és egy core fájlban eltárolja annak memóriaképét, amely segítségével később folytatható a futtatás (például gdb-vel).

more <fájl név> Kiírja a képernyőre a fájl tartalmát, amíg kifér a képernyőre és vár egy billentyű leütésig, majd folytatja a kiírást.

less <fájl név> Kiírja a képernyőre a fájl tartalmát, amíg kifér a képenyőre, majd fel és le is lehet görgetni a dokumentumot. (Kilépni a *q* billentyű leütésével lehet.)

<program neve> > <ki-fájl> < <be-fájl> Átirányítás. A program kimenetét a megadott fájlba írja, bemenetét pedig a másik megadott fájlból veszi. A „>" jel használatával az esetlegesen már létező fájl tartalmát felülírja, a „ » " jel esetén viszont a fájlhoz hozzáír (append).

< 1. program neve> | <2. program neve> Csővezeték. Az első program kimenetét a második program bemenetére irányítja.

sort Lexikografikus rendezés a bemenet soraira. Argumentum nélkül a standard inputról dolgozik, de megadható fájlnév is. A „-r" opcióval csökkenő sorrendbe rendez.

difi* <1. fájl neve> <2. fájl neve> Fájlok összehasonlítása. Szöveges fájloknál a különbséget írja ki. Bináris fájloknál csak az eltérés tényét állapítja meg.

man <Unix parancs, C függvény, stb.> On-line Unix kézikönyv. Leírást ad arról, amiről kértük. Mivel a kézikönyv kötetekből áll, esetleg szükséges lehet a kötet számának a megadása is, például: **man 2 printf**, ellenkező

esetben előfordulhat, hogy egy másik, ugyanolyan nevű parancs leírását kapjuk!

Érdeklődő hallgatóinknak javasoljuk valamelyik ajánlott forrás ([16] vagy [12]) olvasását és a benne szereplők kipróbálását. Jó munkát!

Irodalomjegyzék

Folyóirat- és konferenciacikkek

[1] Farkas Károly: *IPv6 - A jövő Internet-protokollja?,* Híradástechnika, 2005./10. 2-6.

[2] Lencse, Gábor: *Statistics Collection for the Statistical Synchronisation Method,* Conference Paper, appeared in the *Proceedings of the 10th European Simulation Symposium* (ESS'98, Nottingham, England, October 26-28. 1998.), SCS-Europe, 46-51.

Könyvek

[3] Albitz, Paul and Cricket Liu: *DNS and BIND,* 4th ed., O'Reilly, Sebastopol, CA, 2001.

[4] Comer, Douglas E.: *Internetworking with TCP/IP,* Vol. I: Priciples, Protocols and Architectures, 3rd ed., Prentice Hall, Inc., Upper Saddle River, NJ, 1995.

[5] Cooklev, Todor: *Wireless Communicaton Standards,* IEEE Press, New York, 2004.

[6] Ferrero, Alexis: *Az örök Ethernet,* Szak Kiadó Kft. Bicske, 2001.

[7] Ferrero, Alexis: *The Eternal Ethernet,* 2nd ed. Addison Wesley Longman, Harlow, England 1999.

[8] Györfi László és Páli István: *Tömegkiszolgálás informatikai rendszerekben,* Műegyetemi Kiadó, 2001.

[9] Huitema, Christian: *IPv6 - The new internet protocol,* 2nd ed., Prentice Hall PTR, Upper Saddle River, NJ, 1998.

[10] Jain, Ray: *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling,* Wiley- Interscience, New York, NY, April 1991.

[11] Jain, Raj: *FDDI Handbook: High Speed Networking Using Fiber and Other Media,* Addison-Wesley Publishing Company, Reading MA, April 1994.

[12] Pere László: *Linux felhasználói ismeretek L,* Kiskapu Kft. Pécs, 2002.

[13] Quinn, Liam B. and Richard G. Russell: *Fast Ethernet,* Wiley *k* Sons, Inc., 1997.

[14] Roberts, Dave: *Internet Protocols Handbook,* The Coriolis Group, Inc., Scottsdale, AZ, 1996.

[15] Siyan, Karanjit S.: *Inside TCP/IP,* Third Edition, New Riders Publishing, Indiana, 1997.

[16] Szeberényi Imre: *Bevezetés a UNIX operációs rendszerbe,* oktatási segédlet, 4. bővített kiadás, BME Informatika és Irányítástechnika Tanszék, Budapest, 1998.

[17] Tanenbaum, Andrew S.: *Számítógép-hálózatok,* 3. kiadás, Panem Könyvkiadó Kft. Budapest. 1999.

[18] Thomas, Stephen A.: *IP kapcsolás és útvonalválasztás,* Kiskapu Kft, Budapest, 2002.

[19] Stevens, W. Richard: *TCP/IP Illustrated,* Vol. 1. The Protocols, Addison Wesley Longman, Reading, MA, 1994.

Webes források

[20] *A távközlés-informatika szakirány honlapja*
http://www.tilb.sze.hu

[21] FSF: *The Free Software Definition*
http://www.fsf.org/licensing/essays/free-sw.html

[22] FSF: *GNU General Public License*
http://www.fsf.org/licensing/licenses/gpl.html

[23] IANA: *IP Address Services*
http://www.iana.org/ipaddress/ip-addresses.htm

[24] IANA: *Port Numbers*
http://www.iana.org/assignments/port-numbers

[25] IEEE Standards Association: *IEEE OUI and Company_id Assignments*
http://standards.ieee.org/regauth/oui/index.shtml

[26] *ImiNet Network Expert System,* Elassys Consulting Kft.
www.elassys.hu

[27] *Internet Assigned Numbers Authority*
http://www.iana.org

[28] *Internet Szolgáltatók Tanácsa*
http://www.domain.hu

[29] *IPv6 Datagram Main Header Format* in „The TCP/IP Guide"
http://www.tcpipguide.com/free/t_IPv6DatagramMainHeaderFormat.htm

[30] *OMNeT++ Discrete Event Simulation System,* OMNeT++ Community Site
http://www.omnetpp.org

[31] *OPNET Modeler,* OPNET Technologies, Inc.
http://www.opnet.com

[32] Pongor György: *Kommunikációs hálózatok szimulációja,* dr. Pongor György egyetemi előadásainak anyaga, szerkesztették: Fellegi István, Czopf Ákos, Ladányi József, BME, VIK, 1993-94.
http://www.hit.bme.hu/anonftp/pongor/szimjegy/szimjegy.zip

[33] *Subnetting and CIDR,* in „Connected: An Internet Encyclopedia"
http://www.freesoft.org/CIE/Course/Subnet/index.htm

[34] Wikipedia: *Endianness*
http://en.wikipedia.org/wiki/Endianness

[35] Wikipedia: *Filesystem Hierarchy Standard*
http://en.wikipedia.org/wiki/Filesystem__Hierarchy_Standard

[36] Wikipedia: *Fresnel zone*
http://en.wikipedia.org/wiki/Fresnel_zone

[37] Wikipedia: *Regional Internet Registry*
http://en.wikipedia.org/wiki/Regional_Internet_Registry

Más, nehezen hozzáférhető anyagok

[38] Kárpáti László: *KRONE - A passzív hálózat,* PrimusNet Kft elő'adásanyaga, elhangzott: Győr, Széchenyi István Egyetem, Számítógéphálózatok tantárgy keretében: 2005. 05. 02.

[39] Lugosi Zoltán: *A Széchenyi István Egyetem Távközlési Tanszék WLAN lefedésének megtervezése,* szakdolgozat, Széchenyi István Egyetem, Távközlési Tanszék, 2002.

[40] Nagy Zoltán: *A SZIF kollégiumban működő számítógépes hálózat vizsgálata és átalakításának megtervezése,* szakdolgozat, Széchenyi István Főiskola, Távközlési Tanszék, 2000.