Roszik János

SZÁMÍTÓGÉP-HÁLÓZATOK GYAKORLATI SEGÉDANYAG

mobiDIÁK könyvtár

Roszik János

SZÁMÍTÓGÉP-HÁLÓZATOK GYAKORLATI SEGÉDANYAG

mobiDIÁK könyvtár

SOROZATSZERKESZTŐ

Fazekas István

Roszik János

SZÁMÍTÓGÉP-HÁLÓZATOK GYAKORLATI SEGÉDANYAG

Egyetemi jegyzet

mobiDIÁK könyvtár
Debreceni Egyetem
Informatikai Kar

Lektor

Dr. Almási Béla
Debreceni Egyetem

mobiDIÁK könyvtár
Debreceni Egyetem
Informatikai Kar
4010 Debrecen, Pf. 12
http://mobidiak.inf.unideb.hu

A mű *A mobiDIÁK önszervező mobil portál* (IKTA, OMFB-00373/2003) és a *GNU Iterátor, a legújabb generációs portál szoftver* (ITEM, 50/2003) projektek keretében készült.

Tartalomjegyzék

I. fejezet
Bevezető

A gyakorlat első felében a TCP/IP architektúrát a gyakorlati konfiguráció szempontjából tekintjük át.

A TCP/IP egy protokollkészlet, melyet heterogén rendszerek összekapcsolására hoztak létre. Két legismertebb tagja a Transmission Control Protocol (TCP) és az Internet Protocol (IP), melyek után a protokollcsaládot TCP/IP-nek nevezik.

Az ISO OSI modell hét rétegével szemben a TCP/IP hivatkozási modell a következő négy rétegből áll:

1. **Hoszt és hálózat közötti (gép a hálózathoz) réteg** – Ennek a rétegnek a feladata a hálózati rétegtől kapott bájtok (az aktuális hálózattól függő keretekbe foglalva történő) átvitelének megvalósítása a kommunikációs csatornán, illetve keretek fogadása és a belőlük kapott adatfolyam továbbítása a hálózati réteg felé.
2. **Hálózati (Internet) réteg** – Ez a réteg valósítja meg a hálózatok közötti kommunikációt. A gépek közötti kapcsolat összeköttetés mentes és nem megbízható.
3. **Szállítási (transzport) réteg** – A szállítási réteg felel a gépek közötti pont–pont kommunikációért. Ez a réteg foglalkozik a szolgáltatás minőségi kérdéseivel, legfontosabb feladata (az alkalmazási réteg igényeitől függően) az adatok sorrendben, megbízható módon történő továbbítása.
4. **Alkalmazási (applikációs) réteg** – Ennek a rétegnek a segítségével férnek hozzá az egyes alkalmazások a hálózathoz, ez tartalmazza az összes magasabb szintű protokollt, melyek lehetővé teszik például az elektronikus levelezést és állományátvitelt, vagy a távoli gépre történő bejelentkezést.

II. fejezet
Hoszt és hálózat közötti réteg

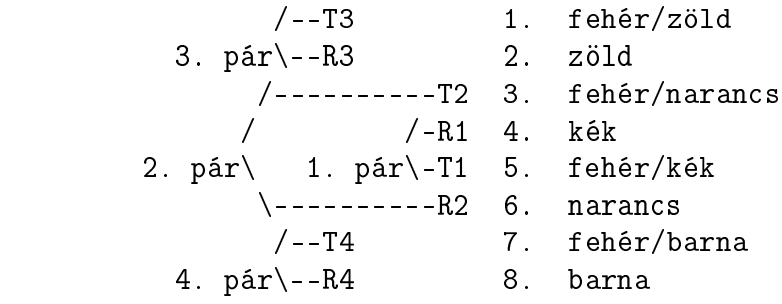
Ez a réteg 2 részből áll, a fizikai rétegből, melynek feladata a jelátvitel megvalósítása, valamint az adatkapcsolati réteből, mely a jelátvitel megbízhatóvá tételéért felelős.

1. Hálózati kábelek, csatlakozók

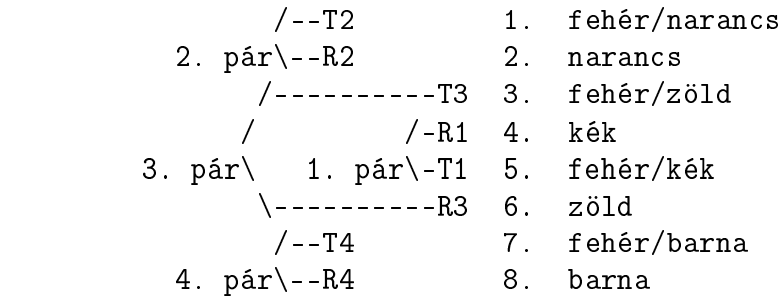
A fizikai réteghez kapcsolódóan, jelen környezetben **CAT5 UTP kábelt** használunk (Category 5 Unshielded Twisted Pair). Ez a kábetípus 100 Mbit/s adatátviteli sebességet tesz lehetővé. 8 vezetéket tartalmaz, melyek hogy az elektromos kisugárzást csökkentsék, páronként össze vannak csavarva, továbbá a 4 pár összecsavart vezeték egymással is össze van csavarva.

A kábelek végein RJ45-ös csatlakozó van. Az EIA/TIA 568-A és EIA/TIA 568-B (Electronic Industries Alliance / Telecommunications Industry Association) szabványok szerint egy csatlakozóban az egyes vezetékek sorrendje a következő:

- EIA/TIA 568-A szabvány



- EIA/TIA 568-B szabvány



Egy kábel két csatlakozója lehet ugyanazon szabvány szerinti, ekkor egyenes kötésű kábelről beszélünk. Ezzel köthető össze pl. egy számítógép hálózati kártyája egy hub-bal. A hub egy többportos jelismétlő (multiport repeater), azaz egy olyan fizikai rétegbeli kapcsolóelem, amely a bármely portján kapott jelet ismétli a többi portján.

Ha a két csatlakozó egyike A, másika B szabvány szerinti, akkor a kábel keresztkötésű. Keresztkötésű (cross–link) kábellel köthetünk össze pl. két számítógépet vagy két hub-ot közvetlenül.

PC esetén az első 2 csatlakozó pontokon történik az adatok küldése, a harmadikon és hatodikon pedig az adatok vétele.

A csavart érpáras kábelek másik két típusa az FTP (Folied Twisted Pair) kábel mely alufóliával árnyékolt (a 4 érpár együtt), valamint az STP (Shielded Twisted Pair) kábel amely érpáranként tartalmaz árnyékolást.

A kialakított kábelezéshez az egyes számítógépeket hálózati adapterek (másnéven hálózati interfész kártyák – NIC: Network Interface Card) segítségével csatlakoztathatjuk. Ilyen pl. az itt használt Ethernet kártya, melyen egy RJ45-ös aljzat van.

Gyakorlati feladat: 2 féle UTP kábel, hub megtekintése, kipróbálása.

2. Az Ethernet

Az **Ethernet** a Xerox cég által kifejlesztett helyi hálózati technológia, amely 10 Mbps adatátviteli sebességet biztosít koaxiális kábelen vagy árnyékolatlan csavart érpáron. Továbbfejlesztett változataival 100 Mbps (Fast Ethernet) és 1-10 Gbps (Gigabit Ethernet) sebesség is elérhető. Az Ethernet hálózatoknak többféle típusa ismert: 10Base2, 10Base5, 10Base-T, 100Base-T, 10Base-F, 100Base-F, 100Base-TX, 100Base-FX, 1000-Base-T. Itt az első szám a sebességet jelenti Mbps-ben, a Base az alapsávi átvitelre utal (nincs vivőfrekvencia), a következő szám pedig a maximális szegmenshosszt adja meg 100 méterekben. A 10Base2 és 10Base5 koax kábelt használ, a T a csavart érpárra, az F pedig az üvegszálra utal. A 100Base-TX és 100Base-FX a Fast Ethernet szabványt (IEEE 802.3u) jelöli. 100Base-TX esetén a maximális kábelhossz 100 méter.

Az Ethernet régebben sín, újabban kiterjesztett csillag topológiájú, és a CSMA/CD elérési rendszert használja a közeghozzáférés szabályozására.

Az adatkapcsolati rétegben azonosítani kell tudni az egyes kártyákat, mindegyiknek egyedi, 48 bites címe van (**Ethernet cím**, másnéven fizikai cím vagy hardver cím). Az első 24 bit a kártya gyártóját azonosítja, a második 24 bit pedig egy sorszám.

Gyakorlati feladat: Az Ethernet cím lekérdezése.
UNIX rendszerekben: /sbin/ifconfig -a vagy /usr/sbin/ifconfig -a
Windows rendszerekben: ipconfig /all esetleg winipcfg /all

3. Hálózati kapcsolóelemek az adatkapcsolati rétegben

A **híd (bridge)** két hálózat adatkapcsolati szintű összekapcsolását végzi. Értelmezi az adatkapcsolati réteg információit, ez alapján eldönti, hogy a keret címzettje melyik porton található, és ha szükséges továbbítja azt a másik portra.

A kettőnél több csatlakozóval (porttal) rendelkező hidat **kapcsolónak (switch–nek)** nevezzük. A switch képes az egyes portok között egymástól függetlenül kereteket továbbítani, azaz két különböző portpár között egyidőben folyhat maximális sebességgel az adattovábbítás.

III. fejezet
Hálózati réteg

A hálózati réteg feladata bármely két csomópont közötti kommunikáció lehetőségének megteremtése.

1. IP címek, címzési osztályok

Az Internet Protokoll (jelenleg még szinte mindenütt használt 4–es verziója) azonosítóként 32 bites értékeket (**IP címeket**) használ, ez alapján történik a csomagtovábbítás és az útvonal kiválasztás. (Az IPv6 címek 128 bitesek.) Az IP cím nem a gépet, hanem annak egy hálózati adapterét azonosítja, így egy gépnek több IP címe is lehet. A teljes TCP/IP hálózat routerekkel összekapcsolt alhálózatokból épül fel. A **router** (forgalomirányító) működése hasonlít a kapcsolóéhoz. Minden beérkező csomagot a megfelelő szegmensre, vagy annak irányba továbbítja, de a hálózati réteg információit is kiolvassa a csomagokból, továbbításukhoz az IP címet használja. A számítógépeket és egyéb címmel rendelkező hálózati eszközöket úgy tudjuk azonosítani, hogy megadjuk melyik hálózat melyik gépe. Így az IP cím két részből áll:

- hálózat azonosítóból és
- (az adott hálózaton belüli) csomópont azonosítóból.

Azt, hogy az IP címben melyik rész a hálózat és melyik a hoszt azonosító a **netmask** határozza meg. A netmask szintén 32 bites, és (balról folytonosan) azon bitjei 1-esek, ameddig a címben a hálózat azonosító tart.

Az IPv4 címeket pontozott decimális alakban írjuk le, azaz a 4 bájt értékét különkülön írjuk fel decimálisan, ponttal elválasztva. Pl.: 193.6.135.99. A netmask–ot is hasonlóan szokás felírni, pl.: 255.255.255.0, de gyakran használják azt az írásmódot, amikor a cím után a hozzá tartozó netmask 1–eseinek darabszámát (a prefix hosszát) írják ki, pl.: 193.6.135.99/24.

Gyakorlati feladat: IP cím és netmask lekérdezése.

Mivel az egyes hálózatok mérete igen különböző lehet, ezért több **címzési osztályt** vezettek be. Ezen címosztályok jellemzői az 1. táblázatban láthatók.

Ezeket az osztályokat belső hálózatunkon a netmask segítségével további alhálózatokra is bonthatjuk.

A 224.0.0.0 – 239.255.255.255 tartományban lévő címek a D osztályú, úgynevezett **multicast címek**. A multicast címek interfészek egy csoportját azonosítják, az ilyen címre küldött csomagot minden, az adott csoportba tartozó interfész megkapja.

osztály	hálózat azon. bitek	csp. azon. bitek	1. bájt értéktart.	default netmask
A	8 darab	24 darab	0–127	255.0.0.0
B	16 darab	16 darab	128–191	255.255.0.0
C	24 darab	8 darab	192–223	255.255.255.0

1. táblázat. Címzési osztályok

A 240.0.0.0 – 255.255.255.255 tartományban lévők pedig az E osztályú, foglalt, nem használt címek.

Gyakorlati feladat: Írjuk fel a netmask-ot pontozott decimális alakban 16, 17, 22 és 26 hálózat azonosító bitek esetén!
Hány bit azonosítja a hosztokat azokban a hálózatokban, ahol a netmask: 255.255.224.0, 255.255.248.0 és 255.255.255.128?

2. Speciális IP címek

Egyes IP címeknek speciális jelentésük van, ezeket a 2. táblázat foglalja össze.

0 ... 0	az aktuális gép (nem lehet célcím)
0 ... 0 hoszt	az aktuális hálózaton a hoszt (nem lehet célcím)
hálózat 0 ... 0	a hálózat azonosítója
hálózat 1 ... 1	üzenetszórás (broadcast) az adott hálózaton
1 ... 1	üzenetszórás a saját hálózaton
127.bármi	visszacsatolási (loopback) cím

2. táblázat. Speciális IP címek

A **loopback** címre érkező csomagot a rendszer saját magának küldi vissza a loopback interfész segítségével, mintha az hálózaton keresztül érkezett volna. Így olyan számítógépen is képesek vagyunk hálózati kommunikációra (a saját gépünkkel) amelyben nincs semmilyen fizikai hálózati eszköz.

A belső, magán hálózatok számára az IP címek közül elkülönítettek több, különböző méretű részt. Az ezekbe a tartományokba eső belső címek bármely helyi hálózatban szabadon használhatók, nem egyediek az egész világon. Egyrészt akkor érdemes őket alkalmazni, ha a hálózatunk nem kapcsolódik más hálózatokhoz. Ha igen, akkor a külvilág felé a Network Address Translator (NAT) fordítja őket más, gyakran külső, regisztrált címekké, az érkező válasz csomagok esetén pedig vissza. Így sokkal kevesebb IP cím regisztrálására van szükség, egy hálózathoz egy regisztrált IP cím is elegendő. Ekkor a külvilág felé az egész hálózat egy címen látszik, és így kívülről nem lehet megcímezni az egyes hosztokat.

A 3. táblázatban ezeknek a tartományoknak a jellemzői láthatók.

hálózat azonosító	netmask	tartomány
10.0.0.0	255.0.0.0	10.0.0.0 – 10.255.255.255
172.16.0.0	255.240.0.0	172.16.0.0 – 172.31.255.255
192.168.0.0	255.255.0.0	192.168.0.0 – 192.168.255.255

3. táblázat. Privát IP cím tartományok

Gyakorlati feladat: Mi lesz a hálózat azonosítója és a broadcast cím 193.6.231.232 IP cím és 16, 17, 22 valamint 26 hálózatazonosító bit esetén?

3. Az ARP és RARP protokollok

A hálózati kártyák a 48 bites fizikai címekre alapozva küldenek és fogadnak kereteket. A hálózati réteg viszont csak az IP címet kapja meg, ezért ezt valahogyan le kell képezni a hálózati kártyákhoz tartozó címekre. Erre szolgál az **ARP** (Address Resolution Protocol – címfeloldási protokoll). Minden csomópont egy táblázatban (ARP táblázat) tartja nyilván a hálózati címekhez tartozó fizikai címeket. Ebbe a táblázatba a következő módon kerülnek bejegyzések:

1. a csomópont üzenetszórással feltesz egy ARP kérdést: Ki tudja az X IP címhez tartozó fizikai címet? Ehhez a kérdéshez a csupa egyesekből álló ff:ff:ff:ff:ff:ff címet használják (Ethernet broadcast).
2. ha van X IP című csomópont, akkor a saját fizikai címével megválaszolja a kérdést. A kérést küldő pedig a kapott információt bejegyzi az ARP táblázatába.

A hálózati és fizikai címpárok egy előre definiált ideig tárolódnak a táblázatban.

Gyakorlati feladat: Az ARP táblázat lekérdezése az /sbin/arp (vagy /usr/sbin/arp) paranccsal.

A **ping** egy olyan ICMP–n (Internet Control Message Protocol) alapuló szolgaltatás, amellyel egy csomópont elérhetőségét lehet vizsgálni (ICMP ECHO_REQUEST csomagokat próbál küldeni neki).

Gyakorlati feladat: a ping paranccsal „pingeljünk” meg egy másik gépet a teremben. Vizsgáljuk meg, hogy bekerült–e az ARP táblázatunkba!
Próbáljuk ezt ki egy más alhálózatba tartozó géppel is, mi történik és miért?
Megjelenik–e valami a táblázatban, ha a két gép között egy hub/switch van?

A Reverse Address Resolution Protocol–ra (**RARP**) olyan eszközöknek van szükségük, amelyek nem ismerik a saját IP címüket (pl. lemez nélküli munkaállomások). Ezek üzenetszórással elküldik a fizikai címüket, amire a RARP szerver az IP címmel válaszol.

4. Routing

A TCP/IP hálózatokat tehát routerek kapcsolják össze, ezek az IP címmel rendelkező eszközök irányítják a hálózatok közötti csomagforgalmat. A **routing** az a folyamat, amikor egy eszköz (a routing táblája alapján) eldönti, hogy a kapott IP csomagot merre továbbítsa.

Gyakorlati feladat: A routing tábla lekérdezése a /sbin/route [-n] vagy /bin/netstat -r[n] paranccsal.

A routing tábla minden sora tartalmazza:

- a cél címét és a hozzá tartozó netmask–ot,
- az interfészt amelyen a csomagot továbbítja,
- valamint a következő router címét, ha router szükséges a csomag továbbításához.

A 4. és 5. táblázatban egy hoszt routing táblájának legfontosabb oszlopai láthatók normál és numerikus (-n opció) nézetben.

Destination	Gateway	Genmask	Iface
192.168.1.0	*	255.255.255.0	eth0
default	192.168.1.19	0.0.0.0	eth0

4. táblázat. Routing tábla normál nézetben

Destination	Gateway	Genmask	Iface
192.168.1.0	0.0.0.0	255.255.255.0	eth0
0.0.0.0	192.168.1.19	0.0.0.0	eth0

5. táblázat. Routing tábla numerikus nézetben

A routing tábla alapján a routing a következőképpen történik. Soronként (a maszk által meghatározott hálózat azonosító bitek száma szerinti sorrendben, a legnagyobbal kezdve) összehasonlítja a cél IP címből és az aktuális sorban lévő mask–ból bitenkénti éssel kapott értéket az aktuális célcímmel. Ha ezek azonosak, akkor továbbítja a csomagot az adott interfészen keresztül. Ha a Gateway mezőben nem 0.0.0.0 található, akkor az adott gépnek küldi tovább, egyébként pedig az adott hálózatban lévő hoszthoz közvetlenül továbbítja. A tábla utolsó sorában általában megtalálható a default gateway. Mivel itt a mask és a célcím is csupa nullákból áll, ezért itt minden (eddig nem illeszkedő) cím illeszkedni fog, azaz ezek a csomagok ide továbbítódnak, ennek a gépnek a routing táblája alapján pedig tovább. Ha nincs default gateway és egyik sor sem illeszkedik, akkor a csomagot nem tudja továbbítani és ICMP hibaüzenetet kapunk.

5. Ethernet interfész konfiguráció, router megadás

Az Ethernet interfész konfigurációjára UNIX rendszerekben az **ifconfig** paranccsal van lehetőség, kizárólag a root felhasználó számára.

A loopback interfész a következő paranccsal konfigurálható:
ifconfig lo 127.0.0.1

Az Ethernet kártya konfigurációjának általános alakja:
ifconfig *interfész IP_ cím* broadcast *broadcast_ cím* netmask *netmask*
Pl.: ifconfig eth0 192.168.1.45 broadcast 192.168.1.255 netmask 255.255.255.0

A broadcast cím és a netmask megadása nem kötelező, ha nem adjuk meg az alapértelmezetteket állítja be.

Ekkor már tudunk kommunikálni a saját hálózatunkban lévő gépekkel. Hogy más hálózatokban lévő eszközökkel is tudjuk kommunikációt folytatni, meg kell adnunk egy default gateway–t a route paranccsal, melynek általános, legegyszerűbb alakja:
route add default gw *gateway_ címe*

Lehetőség van hálózatok bejegyzésére is:
route add -net *hálózat_ címe/netmask_ egyesek_ száma* gw *gateway_ címe*

Ezek eltávolítása úgy lehetséges, hogy az add helyett del–t használunk.

Gyakorlati feladat: jegyezzük fel gépünk beállításait, majd a trinux CD segítségével próbáljuk ki az interfész beállítást és a router megadást a feljegyzett adatokkal!

Gyakorlati feladat: hozzunk létre a teremben 3 alhálózatot, majd ezeket tegyük egymásból közvetlenül elérhetővé!
(a hálózatok címe legyen: 192.168.1.0, 192.168.2.0, 192.168.3.0; a netmask: 255.255.255.0, a gateway pedig a saját gép)

A routing tesztelésére a **traceroute** paranccsal van lehetőség: követi a csomagok útját, kiírja milyen kiszolgálókon és routereken halad keresztül.

Gyakorlati feladat: traceroute kipróbálása: /usr/sbin/traceroute *IP_ cím*

IV. fejezet
Szállítási réteg

Ebben a rétegben két szállítási protokollt definiálunk:

- **TCP**: megbízható, összeköttetés alapú protokoll, azaz a kommunikáció kezdetekor ki kell építeni egy kapcsolatot, a végén pedig le kell bontani. A TCP a beérkező adatfolyamot feldarabolja és továbbítja a hálózati rétegnek. A célállomás TCP folyamata összegyűjti a beérkezett üzeneteket, sorbarendezi, és egyetlen kimeneti adatfolyamként továbbítja. Szükség lehet forgalomszabályozásra is, ami azt jelenti, hogy a gyors forrásállomás csak annyi üzenetet küldjön a lassabb célállomásnak, amennyit az fogadni képes.

  Hogy egy csomópont egyszerre több TCP kapcsolatot is fenn tudjon tartani, szükség van egy az alkalmazási réteg számára nyújtott azonosítóra, amit **port**–nak nevezünk. A port egy 16 bites szám.
- **UDP** (User Datagram Protocol): nem megbízható, összeköttetés mentes protokoll, lényegében az IP szolgáltatásait nyújtja az alkalmazási réteg felé. Akkor célszerű alkalmazni, ha kevés csomagot akarunk küldeni, és nem szükséges sem az üzenetek sorbarendezése, sem a forgalomszabályozás, hanem sokkal fontosabb a sebesség.

  Az UDP is rendelkezik portokkal, amik teljesen függetlenek a TCP portjaitól.

V. fejezet
Alkalmazási réteg

Az alkalmazási réteghez a felhasználói programok által igényelt protokollok tartoznak, pl. az elektronikus levelezéshez, állománytovábbításhoz szükségesek.

A **kliens/szerver modell** két alkalmazás közötti viszonyt definiál. A kliens valamilyen szolgáltatást igényel, a szerver pedig valamilyen szolgáltatást nyújt.

1. A TCP szolgálati modell

A TCP szolgálat úgy valósul meg, hogy mind a küldő, mind a fogadó létrehoz egy csatlakozónak (socket) nevezett végpontot. A csatlakozó cím a hoszt IP címéből és egy portszámból áll.

A 0 és 1023 közötti portszámok foglaltak (csak a rendszergazda foglalhatja le őket), ezeken találhatóak a jól ismert szolgáltatások (well known services). Az 1023 fölötti portszámok szabadon használhatóak.

2. A /etc/services állomány

A jól ismert szolgáltatások jellemzői UNIX rendszerekben a /etc/services állományban találhatók meg. Az egyes szolgáltatásokhoz 1–1 sor tartozik, amely a következő oszlopokból áll:

- szolgáltatás név
- port/protokoll (tcp vagy udp)
- alias nevek
- # után megjegyzés

A következő pár sor egy részlet egy /etc/services állományból:

```
ftp     21/tcp
fsp     21/udp   fspd
ssh     22/tcp           # SSH Remote Login Protocol
ssh     22/udp           # SSH Remote Login Protocol
telnet  23/tcp
smtp    25/tcp   mail
```

Ezeket a jól ismert szolgáltatásokat a szerver nem biztos hogy nyújtja, és ha nyújtja akkor az is elképzelhető, hogy nem ilyen paraméterekkel.

Gyakorlati feladat: a /etc/services állomány megtekintése.

3. A telnet program

A **telnet** egy általános TCP kliens, segítségével kapcsolódhatunk egy hoszt valamely TCP portjához.

Használata: telnet *hoszt port*

Pl.: telnet irh.inf.unideb.hu 13

Gyakorlati feladat: telnet kipróbálása.

Az **SMTP** (Simple Mail Transfer Protocol – egyszerű levéltovábítási protokoll) az elektronikus levelek küldésének szabályait definiálja. Az SMTP protokoll ismeretében egy egyszerű email–t el tudunk küldeni akár telnettel is, levelezőprogram nélkül.

Gyakorlati feladat: küldjünk email–t egy saját címünkre telnet segítségével!

Az elektronikus levelek nem a saját, hanem más kiszolgálón keresztül való elküldését **relay**–nek nevezik. Ezt a jól beállított SMTP szerverek nem engedélyezik, vagy csak annyit engednek meg, hogy az adott szerverre (vagy bizonyos szabályok alapján más szerverekre) küldhessen bárki ilyen módon levelet. Azokat a szervereket, amelyeken a relay korlátlanul engedélyezve van open relay szervereknek nevezik, általában ezeket használják a kéretlen levelek küldői.

A HTTP szerverek a 80–as porton várják a HTTP protokoll szerinti kéréseket. Ha telnettel kapcsolódunk egy (ilyen szolgáltatást nyújtó) gép 80–as portjához, akkor a GET *állomány_elérési_útja* paranccsal kérhetünk le állományokat. Pl.: GET /index.html

Gyakorlati feladat: kérjük le telnettel a http://www.unideb.hu/index.html weboldalt!

4. Az inetd és a tcpd programok

Az **inetd** (internet daemon) arra szolgál, hogy az adott szerveren igény esetén elindítson bizonyos szolgáltatásokat. Ezt a /etc/inetd.conf konfigurációs állománya alapján teszi, ami soronként 1–1 szolgáltatást jellemez. Az egyes sorok a következőket tartalmazzák:

1. a szolgáltatás neve (a /etc/services állományban benne kell lennie)
2. socket típusa
3. protokoll (a /etc/protocols állományból, pl. tcp vagy udp)
4. wait vagy nowait: azt adja meg, hogy az inetd–nek várakozni kell–e arra, hogy az elindított program befejeződjön vagy folytathatja a kérések fogadását és új kiszolgálókat indíthat, azaz az elindított kiszolgáló program képes–e ezután kezelni a socket–et vagy pedig felszabadítja azt.

Ha nowait van megadva, akkor a kiszolgáló indítása után ha új kérés érkezik az inetd új kiszolgáló programot indít, wait esetén viszont nem, csak ha az előzőleg indított már befejezte működését. A wait–et általában csak datagram socket–eknél használják.

5. a felhasználó akinek a jogaival a kiszolgáló programot futtatni kell
6. a kiszolgáló program elérési útja
7. a kiszolgáló program argumentumai (a program nevét is beleértve)

Gyakorlati feladat: a /etc/inetd.conf állomány megtekintése.

Az inetd–vel az a probléma, hogy itt nem tudjuk szabályozni, hogy az egyes szolgáltatásokat kinek nyújtsuk és kinek ne. Erre szolgálnak az úgynevezett tcp wrapper–ek. UNIX rendszerekben egy ilyen program a **tcpd**. Ha egy kérés érkezik a szerverhez, az inetd a tcpd programot indítja el a szolgáltatás helyett, ami pedig a konfigurációs állományai (/etc/hosts.allow és /etc/hosts.deny) alapján eldönti, hogy futtassa-e a kiszolgáló programot, azaz jogosult-e a kliens a szolgáltatás elérésére.

Gyakorlati feladat: a /etc/hosts.allow és /etc/hosts.deny állományok megtekintése.

VI. fejezet
Név – IP cím megfeleltetési mechanizmusok

Az IP címeket nehéz megjegyezni, ezért érdemes az egyes címekhez könnyen megjegyezhető neveket rendelni. Mivel a hálózati rétegben az IP címekre van szükség a kommunikációhoz, ezért szükség van valamilyen mechanizmusra a nevek hálózati címekké konvertálására.

Kezdetben egy **hosts.txt** állományban tárolták a nevekhez tartozó címeket. Ezt a NIC–nél tartották karban, innen volt letölthető. A hálózat növekedésével ez túl nagy és kezelhetetlen lett.

1. A DNS

Ennek a problémának a megoldására tervezték meg a **DNS (Domain Name System)** névfeloldó rendszert. A DNS célja nevekhez információk rendelése, pl.: IP cím, levelezési információ, stb.

A DNS a neveket körzetek (tartományok, domain–ek) hierarchiájába szervezi. Az Internet néhány száz elsődleges körzetre van osztva (top level domain–ek). Ezek lehetnek országra vonatkozóak (pl.: hu, de, ...) vagy általánosak (pl.: com, edu, org, net, ...). Az ezek alá tartozó alkörzetek pedig már azonosíthatnak valamilyen intézményt, céget, és ezek mindegyikéhez is tartozhatnak további alkörzetek. Így a tartománynevek tere egy fával ábrázolható, melynek gyökerében a . (pont – root domain) áll, innen ágaznak le a legfelső szintű domain–ek, és így tovább egészen a hosztnevekig.

A hosztnevektől kezdve összeolvasott név adja a teljes(en minősített) domain nevet (FQDN – Fully Qualified Domain Name). Ezekben az abszolút tartománynevekben az egyes domain–eket ponttal választjuk el, és a végén a fa gyökerét jelentő pontot is kiírjuk. Például a *www.lib.unideb.hu.* a Debreceni Egyetem könyvtárának www nevű gépét azonosítja.

A domain–ek kezelését domain adminisztrátorok végzik. A DNS névtér egymást nem fedő **zónákra** van osztva. Minden zóna tartalmazza a fa egy részét, és tartalmaz általában legalább két **névszervert**, melyek az adott zóna hiteles információit tárolják. Ezek közül egyen történik az információk adminisztrálása (elsődleges névszerver),

a többi pedig innen tölti le őket meghatározott időközönként (másodlagos névszerverek).

A névszerverek egy **adatbázist** tartanak fenn a hozzájuk tartozó körzetekről, azaz az adott zónáról. Az adatbázis egy állomány, ebben találhatóak meg az úgynevezett **erőforrás rekordok**, melyek a következő mezőkből állnak:

- domain név – a tulajdonos domain neve
- élettartam – mennyi ideig tárolják gyorsítótárban (másodpercben)
- osztály – IN: Internethez tartozó információ
- típus
- érték

A fontosabb típusokat és jellemzőiket a 1.táblázat foglalja össze.

típus	jelentés	érték
SOA	jogosultság kezdete	az ehhez a zónához tartozó paraméterek
A	egy host IP címe	32 bites egész
MX	levél csere	prioritás, a levelet váró körzet
NS	névszerver	egy a körzethez tartozó szerver neve
CNAME	kanonikus név	körzet név
PTR	mutató	hoszt a teljes domain nevével
HINFO	hoszt leírás	CPU és operációs rendszer
TXT	szöveg	tetszőleges ASCII szöveg

1. táblázat. A fontosabb erőforrás rekord típusok

A következő pár sorban egy névszerver adatbázisából származó néhány erőforrás rekord látható:

```
$ORIGIN klte.hu.
math      IN  MX     10 neumann.math.klte.hu.
$ORIGIN math.klte.hu.
pc315m11  IN  A      193.6.135.195
pc315m12  IN  A      193.6.135.196
pc315m13  IN  A      193.6.135.197
pc315m14  IN  A      193.6.135.198
pc315m15  IN  A      193.6.135.199
w3        IN  CNAME  sunsite.math.klte.hu.
www       IN  A      193.6.135.56
```

1.1. Névfeloldás DNS–sel

A DNS segítségével történő névfeloldás általában a következő lépésekből áll. (Egy konkrét implementáció ettől eltérő lehet.)

- Egy alkalmazás egy név feloldását kéri a saját névszerverétől.

- A névszerver megnézi, hogy a kért információ megtalálható–e a saját gyorsítótárában (cache).
  - Ha igen válaszol.
  - Ha nem, akkor megnézi, hogy a kérdés a saját zónájára vonatkozik–e?
    - Ha a szerver által adminisztrált zónára vonatkozik, akkor az adatbázis alapján válaszol és frissíti a cache–t. Ha nem talál választ a kérdésre, akkor nem létezik ilyen információ.
    - Ha nem, akkor
      - ha ismer olyan névszervereket, melyek tudhatják a kért információt (pl. gyorsítótárból, vagy egy a hierarchiában alatta lévő zónára vonatkozik a kérés), akkor az adott névszerverek címeivel válaszol.
      - egyébként a root domain–hez tartozó névszerverek címeivel válaszol.

Ezt a fajta lekérdezést nevezzük **iteratív lekérdezésnek**. Ekkor az alkalmazás az adott névszervertől, ha az nem tudja a keresett információt, névszerverek címeit kapja meg, és neki kell valamelyikhez fordulnia.

Rekurzív lekérdezés esetén a névszerverek maguk teszik fel a kérdéseket más szervereknek, és a kérdést feltevő alkalmazás a kért információt kapja meg, ha létezik ilyen információ. Általában a rekurzív típust használják.

Előfordulhat olyan eset, hogy ismerjük egy hoszt IP címét, és szeretnénk megtudni a hozzá tartozó teljes domain nevet. Ezt **fordított lekérdezésnek** nevezzük. Ilyen lekérdezésekhez az in-addr.arpa. speciális domain használatára van szükség. Ez tartalmazza a hosztok IP címeit fordított pontozott jelöléssel, azaz a cím 4 bájtja fordított sorrendben kerül be a fába. Például az 1.2.3.4 IP című gép 4.3.2.1.in-addr.arpa. néven lesz bejegyezve. Itt használják a PTR típusú erőforrás rekordokat, lekérdezésük pedig az előbbihez hasonló módon történik.

2. Az nslookup parancs

Az nslookup egy olyan program, ami névszerverektől való lekérdezésre használható interaktív és nem interaktív módban.

A nem interaktív mód használatát a következő parancsok szemléltetik:

nslookup *név*
nslookup *név névszerver*
nslookup *IP_cím*
nslookup *IP_cím névszerver*
nslookup *-type=rekord_típus név*

Például:

nslookup www.unideb.hu

nslookup -type=mx inf.unideb.hu
nslookup 193.6.128.5

Az interaktív mód az nslookup paranccsal indítható úgy, hogy a gépnév helyett egy - (kötőjel)–et adunk meg. Ekkor egy promptot kapunk, ahová beírhatunk neveket, IP címeket (így történik a lekérdezés), valamint különböző parancsokat. Ezt a módot akkor érdemes használni, ha több lekérdezést is ki akarunk adni egymás után.

Az itt használható legfontosabb (de nem minden nslookup verzióban implementált) parancsok a következők:

help – kilistázza a használható parancsokat
server *szerver* – kicseréli az alapértelmezett névszervert
lserver *szerver* – kicseréli az alapértelmezett névszervert, de az induláskor használt szerverrel keresi meg az új címét
set type=*típus* – például a következő típusok adhatók meg: A, ANY, CNAME, MX, PTR, SOA, ...
set [no]recurse – rekurzív és nem rekurzív módok váltása
finger *felhasználó* – az utolsó cím lekérdezés által beállított aktuális hoszt finger szerverétől lekérdezi az adott felhasználó adatait
ls [*opciók*] domain [> *fájlnév*] – az adott domain összes rekordját kilistázza (opciók pl.: -t *típus* : az adott típusú rekordokat, -d : minden rekordot listáz)
exit vagy Ctrl+D – kilépés az nslookup–ból

Gyakorlati feladat: Az nslookup parancs kipróbálása rekurzív és iteratív módban (általában /usr/bin/nslookup vagy /usr/sbin/nslookup).

3. Kliens oldali DNS konfiguráció

A következőkben azt tekintjük át, hogy UNIX típusú rendszerekben hogyan konfigurálhatjuk a névfeloldást a kliens oldalon, azaz hogyan tehetjük azt elérhetővé az egyes alkalmazásaink számára.

3.1. A /etc/resolv.conf állomány

Ebben a konfigurációs állományban az található meg, hogy mi a gépünket magába foglaló domain neve, mely domain–ekben hajtsuk még végre a keresést (a nevek megadhatók ún. relatív formában is, ekkor egy részfa gyökerét rögzítjük és addig írjuk le a nevet – itt pedig azt adjuk meg, hogy ezeket mivel próbálja kiegészíteni a rendszer). Továbbá itt adjuk meg az elérhető névszerverek listáját.

A következő pár sorban egy resolv.conf állomány tartalma látható:

```
domain unideb.hu
search inf.unideb.hu
nameserver 193.6.128.5
nameserver 193.6.138.66
```

Gyakorlati feladat: A /etc/resolv.conf állomány megtekintése.

3.2. A /etc/host.conf és /etc/hosts állományok

A /etc/host.conf konfigurációs fájlban elsősorban azon források sorrendjét adjuk meg, amelyekből a számítógép kikeresi a domain–név információkat. Általában a következő bejegyzés található meg benne: *order hosts,bind*.

Ez azt jelenti, hogy először a /etc/hosts állományban fogja keresni az adott információt, majd a bind program felhasználásával a DNS–től próbálja megtudni.

Gyakorlati feladat: A /etc/host.conf és /etc/hosts állományok megtekintése.

3.3. A /etc/nsswitch.conf állomány

A név szolgáltatás kapcsoló állomány hasonlóan a /etc/host.conf állományhoz megadja, hogy milyen sorrendben történjenek a kérés feloldások (a glibc2 pl. ezt az állományt használja).

Például a következő bejegyzést találhatjuk meg benne: *hosts: files nis dns*. Ez azt jelenti, hogy a gépnév feloldó függvény először a /etc/hosts állományt nézze, majd a NIS–t, végül pedig a DNS–t kérdezze.

A NIS (Network Information Service – korábban Yellow Pages-nek nevezték) egy helyi hálózati információszolgáltató rendszer. A NIS szerver az információkat (pl. felhasználók adatai, helyi hosztnév információk) egy adatbázisban tárolja, és innen kérdezhetjük le őket. Így ezeket elég az adott helyi hálózat egy központi gépen tárolni és karbantartani. Például a jelszó adatbázis lekérdezhető az *ypcat passwd* paranccsal.

Gyakorlati feladat: A /etc/nsswitch.conf állomány megtekintése.

4. Névszerver konfiguráció

A UNIX rendszerekben legleterjedtebb DNS megvalósítás a **bind** (Berkeley Internet Name Domain). A következőkben a bind 9–es változatának konfigurációját tekintjük át.

A bind szerver elsődleges konfigurációs állománya a **/etc/named.conf**. (A named.conf állomány elhelyezése implementációfüggő, leggyakrabban a /etc/named vagy a /etc/bind könyvtárban található meg.) Ebben az options utasítással a szerverre vonatkozó általános beállításokat tehetjük meg, például:

```
options {
       directory "/var/cache/bind";
       forwarders {
             193.6.128.5;
       };
       forward first;
       allow-transfer {
```

```
                193.6.7.8;
                193.135.6.7;
        };
};
```

Ezzel megadtuk a szerver munkakönyvtárát, a *forwarders* beállítással annak a szervernek az IP címét, ahová a kéréseket továbbítani kell (*forward first* esetén ha a megadott névszerverek nem tudják a kért információt, akkor megpróbálja a saját adatbázisa alapján megválaszolni, *forward only* esetén csak továbbít).

Az *allow-tarnsfer* beállítással azoknak a hosztoknak az IP címeit adhatjuk meg, amelyek letölthetik az adatbázisunkat.

Ezután megadjuk, hogy melyik állományban találhatók meg a gyökér domain–hez tartozó névszerverek címei. A bind szerver induláskor ezek valamelyikétől kéri le a gyökér domain–hez tartozó aktuális szerverek címeit.

```
zone "." {
        type hint;
        file "/etc/bind/named.root";
};
```

A named.root állomány tartalma megtekinthető a Függelékben. Ezután az egyes zónák megadása a következőképpen történik:

```
zone "localhost" {
        type master;
        file "/etc/bind/db.local";
};
zone "127.in-addr.arpa" {
        type master;
        file "/etc/bind/db.127";
};
zone "0.in-addr.arpa" {
        type master;
        file "/etc/bind/db.0";
};
zone "255.in-addr.arpa" {
        type master;
        file "/etc/bind/db.255";
};

zone "m213" {
        type master;
        file "/etc/bind/db.m213";
};
zone "168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/db.168.192";
```

```
};
```

Ezek közül az első 4 bejegyzés a visszacsatolási és a broadcast zónára vonatkozik. Ezeken kívül az m213 zónát definiáltuk, amit természetesen csak a helyi hálózatunkban használhatunk. A *master* típus azt jelenti, hogy ennek a zónának mi tároljuk a hiteles információit.

Gyakorlati feladat: A /etc/bind könyvtárban található állományok megtekintése.

Másodlagos névszerver esetén pedig például a következő zóna megadást használhatjuk:

```
zone "m213" {
        type slave;
        masters { 193.4.5.6;};
        file "/etc/bind/db.m213";
};
```

Az így konfigurált névszerver a 193.4.5.6 IP című géptől próbálja letölteni annak az ehhez a zónához tartozó adatbázisát meghatározott időközönként.

A db.m213 állományban a következőképpen történik az erőforrás rekordok megadása:

```
$TTL   604800
@      IN     SOA    pc1.m213. root.pc1.m213. (
           2003100501         ; Serial
               604800         ; Refresh
                86400         ; Retry
               2419200         ; Expire
                604800 )       ; Negative Cache TTL
;
@      IN     NS     pc1.m213.
pc1    IN     A      192.168.1.1
@      IN     MX     10  pc1.m213.
www    IN     CNAME  pc1
ftp    IN     CNAME  pc1

pc2    IN     A      192.168.1.2
pc3    IN     A      192.168.1.3
```

A SOA rekord érték mezőjében először a zónához tartozó elsődleges névszervert és a zónáért felelős személy email címét kell megadni (@ helyett .–ot használva). Ezt követi a sorozatszám, az hogy mennyi időnként kell a másodlagos szervereknek az információkat frissíteni (itt másodpercben), hogy sikertelen frissítés esetén mennyi idő múlva próbálják azt újra, az információk lejárati ideje (sikertelen frissítés esetén), valamint az alapértelmezett Time To Live érték.

Az inverz lekérdezésekhez tartozó db.168.192 adatbázis állomány pedig a következőket tartalmazza:

```
$TTL   604800
@      IN      SOA     localhost. root.localhost. (
           2003100501          ; Serial
               604800          ; Refresh
                86400          ; Retry
              2419200          ; Expire
               604800 )        ; Negative Cache TTL
;
@      IN     NS     pc1.m213.
$ORIGIN 1.168.192.in-addr.arpa.
1      IN     PTR    pc1.m213.
2      IN     PTR    pc2.m213.
3      IN     PTR    pc3.m213.
```

Gyakorlati feladat: A névszerver konfigurációja az előbbi példák alapján. Ellenőrizzük ezt különböző lekérdezésekkel!
Próbáljuk ki a *forwarders* beállítás megadását!

VII. fejezet
Postfix konfiguráció

A postfix egy gyors, nagyon könnyen konfigurálható és a sendmail–nél biztonságosabb levéltovábbító (MTA – Mail Transport Agent).

A postfix konfigurációs állományai alapértelmezés szerint a /etc/postfix könyvtárban találhatók meg. A *main.cf* fájlban beállítható legfontosabb paraméterek a következők:

myhostname – a postfix–et futtató gép teljes domain neve
myorigin – ez a paraméter adja meg a levél feladójának címében megjelenő domain nevet, ha az nem volt megadva
mydestination – itt adjuk meg azokat a domain neveket, melyekre küldött leveleket a saját gépen lévő felhasználónak kell továbbítani
mynetworks – itt adhatók meg a megbízható hálózatok, ezek számára engedélyezett a relay
relayhost – azok a hosztok, melyeknek a kimenő leveleket továbbítani kell
mail_owner – a levél küldési sor és a legtöbb postfix processzek tulajdonosa
queue_directory – levél küldési sort tartalmazó könyvtár
command_directory – az adminisztrációs parancsokat tartalmazó könyvtár
daemon_directory – a daemon programokat tartalmazó könyvtár
config_directory – a konfigurációs állományok és az adminisztrációs shell scriptek helye
proxy_interfaces – azoknak a (proxy vagy NAT) hosztoknak a címe, melyeken keresztül a postfix leveleket fogad
masquerade_domains – az a domain név adható meg, amire a levélben a feladó címében szereplőt le kell cserélni

Gyakorlati feladat: A /etc/postfix/main.cf állomány megtekintése.

A. Függelék
A named.root állomány

```
;   This file holds the information on root name servers
;   needed to initialize cache of Internet domain name servers
;   (e.g. reference this file in the "cache  .  <file>"
;   configuration file of BIND domain name servers).
;
;   This file is made available by InterNIC
;   under anonymous FTP as
;       file                /domain/named.root
;       on server           FTP.INTERNIC.NET
;
;   last update:    Nov 5, 2002
;   related version of root zone:   2002110501
;
;
; formerly NS.INTERNIC.NET
;
.                       3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.     3600000      A     198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                       3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.     3600000      A     128.9.0.107
;
; formerly C.PSI.NET
;
.                       3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.     3600000      A     192.33.4.12
;
; formerly TERP.UMD.EDU
;
.                       3600000      NS    D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.     3600000      A     128.8.10.90
;
; formerly NS.NASA.GOV
;
.                       3600000      NS    E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.     3600000      A     192.203.230.10
;
; formerly NS.ISC.ORG
```

```
;
.                        3600000      NS    F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.      3600000      A     192.5.5.241
;
; formerly NS.NIC.DDN.MIL
;
.                        3600000      NS    G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.      3600000      A     192.112.36.4
;
; formerly AOS.ARL.ARMY.MIL
;
.                        3600000      NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.      3600000      A     128.63.2.53
;
; formerly NIC.NORDU.NET
;
.                        3600000      NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.      3600000      A     192.36.148.17
;
; operated by VeriSign, Inc.
;
.                        3600000      NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.      3600000      A     192.58.128.30
;
; housed in LINX, operated by RIPE NCC
;
.                        3600000      NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.      3600000      A     193.0.14.129
;
; operated by IANA
;
.                        3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.      3600000      A     198.32.64.12
;
; housed in Japan, operated by WIDE
;
.                        3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.      3600000      A     202.12.27.33
; End of File
```

B. Függelék
Alhálózatokra bontás

Legyen a hálózat címe 193.4.5.0, a hozzá tartozó netmask pedig 255.255.255.0. Ebben szeretnénk 6 alhálózatot kialakítani.

1. 6 alhálózat esetén 3 bitet kell használni az alhálózatok azonosítására (és mivel $2^3 = 8$, így valójában 8 alhálózatot kapunk).
2. Ezt hozzávesszük az eredeti netmask–hoz, ami így 255.255.255.224 lesz (tehát 5 bit marad a hosztok azonosítására).
3. Az eredeti hálózatcím alapján fel tudjuk írni a lehetséges alhálózat címeket, melyekben már nem 24, hanem 27 bit azonosítja a hálózatot:

   193.4.5.0
   193.4.5.32
   193.4.5.64
   193.4.5.96
   193.4.5.128
   193.4.5.160
   193.4.5.192
   193.4.5.224

4. Mivel a csupa 0 és csupa 1 alhálózat azonosítókat általában nem használják, ezért a keresett 6 alhálózat azonosítója, az egyes alhálózati hosztcím–tartományok valamint a broadcast címek a következők:

1.	193.4.5.32	193.4.5.33 – 193.4.5.62	193.4.5.63
2.	193.4.5.64	193.4.5.65 – 193.4.5.94	193.4.5.95
3.	193.4.5.96	193.4.5.97 – 193.4.5.126	193.4.5.127
4.	193.4.5.128	193.4.5.129 – 193.4.5.158	193.4.5.159
5.	193.4.5.160	193.4.5.161 – 193.4.5.190	193.4.5.191
6.	193.4.5.192	193.4.5.193 – 193.4.5.222	193.4.5.223

Gyakorlati feladatok:

1. bontsuk fel a 195.46.57.0/24 című hálózatot 2 részre
2. bontsuk fel a 193.172.0.0/16 című hálózatot 5 részre
3. bontsuk fel a 172.140.0.0/14 című hálózatot 9 részre
4. bontsuk fel a 172.17.136.0/21 című hálózatot 6 részre
5. bontsuk fel a 192.168.1.128/25 című hálózatot 2 részre

C. Függelék
CIDR címkiosztás

A CIDR (Classless Inter-Domain Routing – osztálynélküli körzetek közötti forgalomirányítás) egy olyan címzési mód, ami lehetővé teszi az IP címek az osztály alapúnál sokkal hatékonyabb kiosztását. Erre azért volt szükség, mert egyrészt nagyon gyorsan fogytak a kioszthatóó IP címek, másrészt pedig egyes router–eknél már óriási méretű routing táblákra volt szükség. A CIDR lényege, hogy a router–ekben egy bejegyzés hálózat–csoportokat is azonosíthat.

A különböző célokat (hálózatok, hosztok) egy IP címmel és egy maszkkal (prefix hosszal) írjuk le. Az Internet–szolgáltatók egy–egy adott tartomány kiosztásáról rendelkeznek, amiből kisebb tartományokat úgy osztanak ki, hogy az adott méretű tartományokat igénylőknek a megfelelő, a sajátjánál hosszabb prefix–szel adnak hálózatcímeket. Tekintsük erre a következő példát:

A szolgáltató a 150.60.0.0 – 150.63.255.255 tartomány kiosztásáról rendelkezik, a szolgáltatót (kívülről) specifikáló információ: <150.60.0.0, 255.252.0.0>. Először a következő három igény érkezik hozzá (nagy időkülönbségekkel):

1. 4000,
2. 6000 és
3. 1000 csomópont címzésére elegendő címtartomány igények.

Mivel

$2^{11} < 4000 \leq 2^{12}$,

$2^{12} < 6000 \leq 2^{13}$ és

$2^{9} < 1000 \leq 2^{10}$,

ezért 12, 13 és 10 bit szükséges az egyes hálózatokban a hosztok azonosítására, azaz a prefix hossz 20, 19 és 22 lesz.

1. igénylő

20 prefix hossz esetén a lehetséges hálózat azonosítók:

150.60.0.0
150.60.16.0
150.60.32.0
150.60.48.0
150.60.64.0
150.60.80.0

...

Az 1. igénylő ebből megkapja az első még ki nem osztottat, azaz a 150.60.0.0–át 20 prefix hosszal. A kapott címtartomány tehát: 150.60.0.0 – 150.60.15.255.

2. igénylő

19 prefix hossz esetén a lehetséges hálózat azonosítók:
150.60.0.0
150.60.32.0
150.60.64.0
150.60.96.0
150.60.128.0
150.60.160.0
...

A 2. igénylő ebből szintén megkapja az első még ki nem osztottat, ami a 150.60.32.0. A kapott címtartomány tehát: 150.60.32.0 – 150.60.63.255.

3. igénylő

22 prefix hossz esetén a lehetséges hálózat azonosítók:
150.60.0.0
150.60.4.0
150.60.8.0
150.60.12.0
150.60.16.0
150.60.20.0
...

A 3. igénylő a 150.60.16.0–át kapja 22 prefix hosszal, mivel az elsőnek már ki lett osztva a 150.60.0.0 – 150.60.15.255 tartomány. Itt a kapott címtartomány tehát: 150.60.16.0 – 150.60.19.255.

Gyakorlati feladatok:
A szolgáltatót specifikáló információ: <191.16.0.0, 255.240.0.0>. Osszuk ki a következő méretű tartományokat úgy, hogy először azt feltételezzük, hogy az igények nem egyszerre érkeztek, majd pedig azt, hogy egyszerre, tehát van lehetőség a sorrend módosítására:

1. 2000, 8000, 2000
2. 1000, 2000, 4000
3. 4000, 6000, 8000
4. 16000, 4000, 500
5. 2000, 6000, 1000

D. Függelék
Ajánlott irodalom

Andrew S. Tanenbaum: *Számítógép–hálózatok*, Panem, 2004.

Tony Bautts, Terry Dawson, Gregor N. Purdy: *Linux hálózati adminisztrátorok kézikönyve*, Kossuth, 2005.