A

A Samba konfigurálása SSL protokollal

Ebben a függelékben arról olvashatunk, miként telepíthetjük úgy a Sambát, hogy biztonságosak legyenek a kapcsolatok a Samba kiszolgáló és az ügyfelei között. A biztonságos kapcsolatról a Netscape által kifejlesztett SSL (Secure Sockets Layer) protokoll gondoskodik. A példánkban egy Samba kiszolgáló és egy Windows NT munkaállomás között hozzuk létre a biztonságos kapcsolatot.

A téma tárgyalása során feltétezzük, hogy az Olvasó tisztában van a nyilvános kulcsú titkosítás és az X.509 bizonyítványok alapjaival. Ha nem így lenne, ajánljuk Bruce Schneier *Applied Cryptography, 2nd Edition* (Wiley) könyvének tanulmányozását, amelyben az alapismeretek mellett a kriptográfia számos más érdekes vonatkozásait is megtalálja. (*Hasonló céllal az Olvasó figyelmébe ajánlható Othmar Kyas Számítógépes hálózatok biztonságtechnikája című munkája, amely 2000 májusában jelent meg a Kossuth Kiadónál - a szerkesztő.)*

A Samba és az SSL kapcsolatáról a Samba disztribúció */docs/textdocs* könyvtárában lévő *SSLeay.txt* dokumentum is számos tudnivalót tartalmaz – ez a függelék is ennek alapján készült.

Röviden a hitelesítésekről

Az alábbiakban néhány kérdést és választ idézünk a Samba dokumentáció *SSLeay.txt* fájljából az SSL és a hitelesítés előnyeiről. A szöveg szerzője Christian Starkjohann.

Mi a bizonyítvány?

A bizonyítványt egy erre illetékes szervezet, általában egy hitelesítő hatóság (Certification Authority, CA) bocsátja ki azzal a céllal, hogy igazoljon valamit. Az igazolás tárgyát a kibocsátó hatóság határozza meg. Így például az a hatóság, amely az elektronikus áruházakat üzemeltető webkiszolgálók biztonságát igazolja, általában csak azt garantálja, hogy egy adott nyilvános kulcs egy adott tartománynévhez tartozik. Egy szervezet belső hitelesítő hatósága azt igazolja, hogy a megfelelő bizonyítvánnyal rendelkező személy valóban a szervezet alkalmazottja, hozzáférhet adott kiszolgálóhoz stb.

Az X.509 bizonyítvány műszaki háttere

Műszaki értelemben a bizonyítvány egy sor adatot tartalmaz, amelyek valódiságát a kibocsátója (CA) az aláírásával igazolja. A bizonyítvány a következő adatokat tartalmazza:

- a bizonyítvány kiállítójának egyedi azonosítója (neve);
- a bizonyítvány időbeli érvényessége;
- a bizonyítvány tulajdonosának egyedi azonosítója (neve);
- a bizonyítvány tulajdonosához tartozó nyilvános kulcs;
- a bizonyítvány kibocsátójának aláírása.

A bizonyítvány érvényességének ellenőrzéséhez az ellenőrzést végzőnek rendelkeznie kell a megbízható hitelesítő hatóságok nevét és nyilvános kulcsát tartalmazó táblával. Az egyszerűség kedvéért ezeknek a tábláknak az illető hatóságok által a saját maguk számára kiállított bizonyítványokat is tartalmazniuk kell (saját maguknak aláírt bizonyítványok).

Mi következik az ilyen bizonyítványok szerkezetéből?

- Mivel a bizonyítvány tartalmazza a tulajdonosának nyilvános kulcsát, a bizonyítvány és egy privát kulcs együtt elegendő a titkosításhoz és a titkosított szöveg visszafejtéséhez.
- A bizonyítványok ellenőrzéséhez szükségünk van az általunk megbízhatónak tartott hitelesítő hatóságok bizonyítványaira.
- „Álbizonyítvány" legegyszerűbben úgy állítható ki, hogy maga a tulajdonos írja alá a bizonyítványt.
- Szükség van hitelesítő hatóságra. Az ügyfél maga nem bocsáthat ki helyi bizonyítványokat az általa megbízhatónak tartott kiszolgálókhoz, mert a kiszolgáló határozza meg, milyen bizonyítványt fogad el.

Követelmények

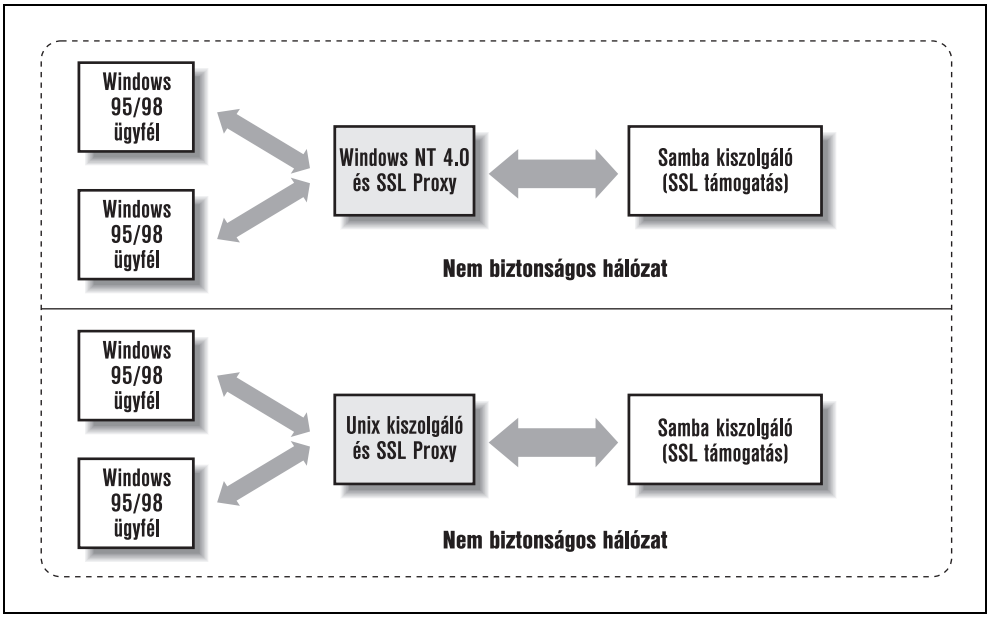
SSL kapcsolatok telepítéséhez a Samba mellett még a következő két programot is le kell tölteni:

SSLeay
: Ez a Unix programkönyvtárak részét alkotó SSL protokoll Eric Young-féle megvalósítása.

SSL Proxy
: Az Objective Development szabadon terjeszthető SSL alkalmazása, amelynek segítségével megbízható kapcsolat hozható létre Unix és NT platformokon.

E két szoftvertermék lehetővé teszi, hogy titkosított SSL kapcsolatot építsünk fel mind a kiszolgálón, mind az ügyfeleinél. Az SSLeay függvénytárak közvetlenül a Unix rendszerben fordíthatók le és telepíthetők. Ezzel szemben az SSL Proxy az ügyfél oldalon tölthető és fordítható le (bináris formátumban is letölthető). Ha az SSL kapcsolat túlsó végéhez Windows NT vagy Samba ügyfél csatlakozik, akkor nincs szükség különleges telepítésre.

Az SSL Proxy program Windows 95/98 gépeken viszont nem működik. Ezért ha egy Samba kiszolgáló és egy Windows 95/98 ügyfél között szeretnénk biztonságos kapcsolatot létrehozni, akkor ugyanabban az alhálózatban, amelyikben a Windows 9*.x* ügyfelek vannak vagy egy Unix kiszolgálót, vagy egy Windows NT gépet is el kell helyezni, és a teljes hálózati forgalmat az SSL Proxy programot futtató gépen keresztül kell lebonyolítani (lásd az A.1. ábrát).



A.1. ábra. Windows 95/98 ügyfelekhez való biztonságos kapcsolódás két lehetősége

A vizsgálatainkhoz példaként egy egyszerű SSL kapcsolatot fogunk létrehozni egy Samba kiszolgáló és egy Windows NT ügyfél között. A bemutatásra kerülő lépések alapján ennél bonyolultabb hálózatok is kiépíthetők a rendszergazda céljától függően.

Az SSLeay telepítése

A Samba a kiszolgáló oldalán az Eric Young által megírt csomagot használja az SSL kapcsolat támogatásához. Az Egyesült Államok exporttörvénye értelmében azonban az SSLeay csomag az Egyesült Államokban készült Samba disztribúciók részeként nem szállítható. Ezért a Samba készítői úgy döntöttek, hogy a csomagot teljes egészében meghagyják különálló szoftverterméknek. Az SSLeay csomag az alábbi FTP-helyekről tölthető le:

- *ftp://ftp.psy.uq.oz.au/pub/Crypto/SSL/*
- *ftp://ftp.uni-mainz.de/pub/internet/security/ssl*
- *ftp://ftp.cert.dfn.de/pub/tools/crypt/sslapps*
- *ftp://ftp.funet.fi/pub/crypt/mirrors/ftp.psy.uq.oz.au*
- *ftp://ftp.sunet.se/ftp/pub/security/tools/crypt/ssleay*

Az amerikai kiadás nyomdába adásakor a 0.9.0b volt a program legfrissebb verziója. A programot ugyanarra a kiszolgálóra töltsük le, ahová a Samba disztribúciót is letöltöttük, csomagoljuk ki és futtassuk le az untar parancsot. Eredményül egy *SSLeay-0.9.0b* könyvtárat kell kapnunk. Lépjünk át ebbe a könyvtárba, és ugyanúgy konfiguráljuk és építsük fel az SSL titkosító csomagot, ahogyan ezt a Sambával is tettük.

Az SSLeay Perl nyelvű konfiguráló szkriptet használ. Ez a szkript módosítja az SSLeay csomag segédprogramjait és függvénytárait létrehozó Make-fájlt. Az alapértelmezés szerinti szkript a Perl értelmezőt a */usr/local/bin/perl* útvonalon keresi. Lehetséges, hogy úgy kell módosítani a konfiguráló szkriptet, hogy arra a helyre mutasson, ahol a Unix rendszerünkben a Perl végrehajtható fájl van. A Perl végrehajtható fájl megtalálásához például az alábbi parancsot írhatjuk be:

```
# which perl
/usr/bin/perl
```

Ezt követően úgy módosítsuk a konfiguráló szkript első sorát, hogy a helyes Perl végrehajtható fájlt használja. Egy Red Hat Linux rendszerben például ez írható be:

```
#!/usr/bin/perl
#
# see PROBLEMS for instructions on what sort of things to do
# when tracking a bug -tjh
...
```

Ezt követően a disztribúció célplatformját megadva futtassuk a konfiguráló szkriptet. A célplatform az alábbiak valamelyik lehet:

```
BC-16              BC-32              FreeBSD            NetBSD-m86
NetBSD-sparc       NetBSD-x86         SINIX-N            VC-MSDOS
VC-NT              VC-W31-16          VC-W31-32          VC-WIN16
VC-WIN32           aix-cc             aix-gcc            alpha-cc
alpha-gcc          alpha400-cc        cc                 cray-t90-cc
debug              debug-irix-cc      debug-linux-elf    dgux-R3-gcc
dgux-R4-gcc        dgux-R4-x86-gcc    dist               gcc
hpux-cc            hpux-gcc           hpux-kr-cc         irix-cc
irix-gcc           linux-aout         linux-elf          ncr-scde
nextstep           purify             sco5-cc            solaris-sparc-cc
solaris-sparc-gcc  solaris-sparc-sc4  solaris-usparc-sc4 solaris-x86-gcc
sunos-cc           sunos-gcc          unixware-2.0       unixware
```

Linux-elf platformhoz az alábbi parancsot kell kiadni:

```
# ./Configure linux-elf
CC            =gcc
CFLAG         =-DL_ENDIAN -DTERMIO -DBN_ASM -O3 -fomit-frame-pointer
EX_LIBS       =
```

```
BN_MULW        =asm/bn86-elf.o
DES_ENC        =asm/dx86-elf.o asm/yx86-elf.o
BF_ENC         =asm/bx86-elf.o
CAST_ENC       =asm/cx86-elf.o
RC4_ENC        =asm/rx86-elf.o
RC5_ENC        =asm/r586-elf.o
MD5_OBJ_ASM    =asm/mx86-elf.o
SHA1_OBJ_ASM   =asm/sx86-elf.o
RMD160_OBJ_ASM=asm/rm86-elf.o
THIRTY_TWO_BIT mode
DES_PTR used
DES_RISC1 used
DES_UNROLL used
BN_LLONG mode
RC4_INDEX mode
```

Miután konfiguráltuk a csomagot, a `make` parancs kiadásával le kell fordítanunk. Ha ez nem sikerülne, akkor olvassuk el a disztribúcióhoz tartozó dokumentációt, vagy a *http://www.cryptsoft.com/ssleay/* webhelyen tájékozódjunk a lehetséges hibákról és tennivalókról. Ha sikerült a csomag lefordítása, akkor a `make install` parancs kiadásával telepítsük a függvénytárakat a rendszerbe. Jegyezzük meg, hogy a make-fájl a csomagot alapértelmezés szerint a */usr/local/ssl* könyvtárba telepíti. Ha a telepítést más könyvtárba akarjuk elvégezni, akkor jegyezzük fel a könyvtár helyét, mert szükségünk lesz rá, amikor a Sambát az SSL használatához konfiguráljuk.

Az SSLeay konfigurálása a rendszerünkhöz

Első lépésként úgy kell beállítanunk a PATH környezeti változót a rendszerünkben, hogy tartalmazza az SSL disztribúció */bin* könyvtárát. Ezt az alábbi módon tehetjük meg:

```
PATH=$PATH:/usr/local/ssl/bin
```

Ez a feladat egyszerűbb része. Következő lépésként elő kell állítanunk egy véletlen karakterekből álló sorozatot, amelyet az SSLeay véletlenszám-generátora bemenetként használ. A véletlenszám-generátor hozza létre mind az ügyfelek, mind a kiszolgáló számára a kulcspárokat. A véletlen karakterekből álló sorozatot úgy készíthetjük el, hogy egy szövegszerkesztőben tetszés szerinti összevisszaságban leütjük a billentyűket. Azt is megtehetjük, hogy kiadjuk az alábbi parancsot, és tetszés szerinti betűket írunk be a standard bemenetről:

```
cat >/tmp/private.txt
```

A Samba dokumentációja azt ajánlja, hogy legalább 1 percig írjunk be folyamatosan karaktereket, mielőtt a Control-D kombináció lenyomásával megszakítanánk a bevitelt. Arra is ügyeljünk, hogy ne csak az „ujjra eső” billentyűket nyomjuk le, hanem számokat és különböző szimbólumokat is vigyünk be. Miután elkészült a véletlen karaktereket tartalmazó fájl, az alábbi paranccsal indíthatjuk a véletlenszám-generátort:

```
# ssleay genrsa -rand /tmp/private.txt >/dev/null
2451 semi-random bytes loaded
Generating RSA private key, 512 bit long modulus
..+++++
.................................+++++
e is 65537 (0x10001)
```

A parancs kimenetét nyugodtan figyelmen kívül hagyhatjuk. Miután lefutott a program, töröljük a kulcs létrehozásához használt karaktersorozatot, mert ez alapján újból létre lehetne hozni a véletlenszám-generátorral privát kulcsokat:

```
rm -f /tmp/private.txt
```

Az előző parancs eredményeként létrejön a home könyvtárunkban egy *.rnd* kiterjesztésű rejtett fájl. Az SSLeay ezt a fájlt fogja használni, amikor a jövőben elkészíti a kulcspárokat.

A Samba konfigurálása az SSL használatához

Elérkeztünk ahhoz a ponthoz, amikor a Sambát a SSL használatához konfigurálhatjuk. Emlékezzünk vissza, hogy *„A Samba telepítése Unix rendszerre”* című 2. fejezetben azt mondtuk, hogy először a konfiguráló szkriptet kell futtatni, ami inicializálja a make-fájlt, mielőtt lefordíthatnánk a Sambát. Ahhoz, hogy a Sambát az SSL protokollal használhassuk, újra kell futtatnunk a konfiguráló scriptet:

```
/configure --with-ssl
```

Ezt követően az alábbi parancsokkal fordíthatjuk le a Sambát:

```
# make clean
# make all
```

Ha olyan hibaüzenetet kapnánk, amely közli, hogy az *smbd* végrehajtható fájl nem találja az *ssl.h* fájlt, akkor valószínűleg nem az alapértelmezés szerinti könyvtárba telepítettük az SSLeay csomagot. Vegyük fel a parancshoz a `--with-sslinc` kapcsolót úgy, hogy az SSL disztribúció alapkönyvtárára mutasson – ami ebben az esetben az *include/ssl.h* fájlt tartalmazó könyvtár.

Ha viszont rendben megtörtént a fordítás, akkor rátérhetünk a következő lépésre, a bizonyítványok elkészítésére.

Hogyan válhatunk hitelesítő hatósággá?

Az SSL protokoll a kapcsolatfelvétel egyeztetése során igényli az X.509-es bizonyítványokat, hogy biztos legyen abban, a kommunikációban részt vevő felek legalább egyike valóban az, akinek mondja magát. A valós életben az ilyen bizonyítványok – amelyeket a nyilvános webhelyek is használnak az SSL kapcsolataikban – ára évi 300 USD körül van.

Ennek az az oka, hogy a bizonyítványnak egy hitelesítő hatóság digitális aláírását is tartalmaznia kell. A hitelesítő hatóság olyan szervezet, amely a saját privát kulcsát használva garantálja egy digitális bizonyítvány hitelességét. Így ha valaki meg akarja vizsgálni egy bizonyítvány hitelességét, akkor ezt a hitelesítő hatóság nyilvános kulcsával minden további nélkül megteheti.

Az SSLeay segítségével bármely nyilvános hitelesítő hatóság szolgáltatását igénybe vehetjük, de ezt el is kerülhetjük. Az SSLeay azt is lehetővé teszi, hogy magunkat nevezzük ki megbízható hitelesítő hatóságnak, és eldöntsük, mely ügyfeleinket tekintjük megbízható, és melyeket megbízhatatlan feleknek. Ehhez különböző feladatokat kell elvégeznünk az SSLeay disztribúcióval.

Elsőként meg kell adnunk azt a biztonságos helyet, ahol a kiszolgáló és az ügyfelek bizonyítványait tároljuk. A példánkban alapbeállításként az *etc/certificates* könyvtárat választjuk.

```
# cd /etc
# mkdir certificates
# chmod 700 certificates
```

Figyeljük meg, hogy a gyökéren kívül senki másnak sem engedélyezzük a könyvtár elérését. Ez nagyon fontos.

Következő lépésként úgy készítsük el az SSLeay szkripteket és a konfigurációs fájlokat, hogy az ebben a könyvtárban tárolt bizonyítványokat használják. Ehhez először úgy módosítsuk a */usr/local/ssl/bin/CA.sh* elérési úton lévő *CA.sh* szkriptet, hogy a most létrehozott könyvtárat specifikálja. Keressük meg a fájlban az alábbi bejegyzést tartalmazó sort:

```
CATOP=./demoCA
```

Írjuk át a sort az alábbi módon:

```
CATOP=/etc/certificates
```

Ezt követően a */usr/local/ssl/lib/ssleay.cnf* fájlt is úgy kell módosítanunk, hogy ugyanerre a könyvtárra mutasson. Keressük meg a következő bejegyzést:

```
[ CA_default ]
dir     = ./demoCA             # Where everything is kept
```

Írjuk át az alábbiak szerint:

```
[ CA_default ]
dir     =  /etc/certificates   # Where everything is kept
```

Most futtassuk a hitelesítő hatóságot telepítő *CA.sh* szkriptet, hogy elkészüljenek a bizonyítványok. Biztosítsuk, hogy ezt a műveletet ugyanazon személyként végezzük el, mint aki a véletlenszám-generátor bemenetét is készítette a korábbiakban:

```
/usr/local/ssl/bin/CA.sh -newca
mkdir: cannot make directory '/etc/certificates': File exists
CA certificate filename (or enter to create)
```

Az Enter billentyű lenyomásával hozzuk létre a hitelesítő hatóság bizonyítványát. Ekkor a következőket látjuk:

```
Making CA certificate ...
Using configuration from /usr/local/ssl/lib/ssleay.cnf
Generating a 1024 bit RSA private key
..........................+++++
.....................+++++
writing new private key to /etc/certificates/private/cakey.pem
Enter PEM pass phrase:
```

Adjunk meg új jelszót a bizonyítványhoz. A jelszót kétszer kell beírnunk, mielőtt elfogadná az SSLeay:

```
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
```

Jól jegyezzük meg ezt a jelszót. Szükségünk lesz rá, amikor majd az ügyfelek bizonyítványát kell aláírnunk. Miután az SSLeay elfogadta a jelszót, újabb kérdéseket tesz fel, és az ezekre adott válaszokkal kitölti az X.509 bizonyítvány megfelelő rovatait:

```
You are about to be asked to enter information that will be
incorporated into your certificate request.
What you are about to enter is what is called a Distinguished
Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
```

Töltsük ki a mezőket a szervezetünkre vonatkozó adatokkal. A könyv szerzői például az alábbi adatokkal készítették el a bizonyítványukat:

```
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:California
Locality Name (eg, city) []:Sebastopol
Organization Name (eg, company) []:O'Reilly
Organizational Unit Name (eg, section) []:Books
Common Name (eg, YOUR name) []:John Doe
Email Address []:doe@ora.com
```

Ezt követően az SSLeay hitelesítő hatóságként konfigurálja magát, és alkalmas lesz arra, hogy bizonyítványokat írhassunk alá a Samba kiszolgálóhoz csatlakozó ügyfelek számára.

Bizonyítványok készítése az ügyfelek számára

Nagyon egyszerűen elkészíthetjük egy ügyfélgép bizonyítványát. Először generálnunk kell a számára egy nyilvános és privát kulcsból álló kulcspárt, majd el kell készítenünk a bizonyítványkérő fájlt, végül az SSLeay segítségével hitelesítő hatóságként alá kell írnunk a fájlt.

A példabeli `phoenix` ügyfélgép bizonyítványának elkészítéséhez mindössze három SSLeay parancsot kell kiadnunk. Az első parancs generálja a kulcspárját, és elhelyezi a *phoenix.key* fájlban. A privát kulcsot titkosítja, esetünkben háromszoros DES eljárással. Ezt követően meg kell adnunk egy jelszót, amire majd a következő lépésben lesz szükség:

```
# ssleay genrsa -des3 1024 phoenix.key
1112 semi-random bytes loaded
Generating RSA private key, 1024 bit long modulus
.....................................+++++
.............+++++
e is 65537 (0x10001)
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
```

Miután lefutott ez a program, adjuk ki az alábbi parancsot:

```
# ssleay req -new -key phoenix.key -out phoenix-csr
Enter PEM pass phrase:
```

Adjunk meg egy jelszót az ügyfél előbb elkészült bizonyítványához (ne a hitelesítő hatóság jelszavát). Most ismét válaszolnunk kell az előzőekből már ismert kérdésekre, ezúttal azonban az ügyfél nevében. Egy ún. challenge jelszót és egy szervezet nevét is meg kell adnunk, de ezeknek itt nincs jelentőségük. Miután lefutott a program, készen áll a *phoenix-csr* nevű fájlban a bizonyítványkérés.

Ezt követően megbízott hitelesítő hatóságként alá kell írnunk a bizonyítványkérést. Írjuk be a következő parancsot:

```
# ssleay ca -days 1000 -inflies phoenix-csr phoenix.pem
```

Ez a parancs bekéri a *hitelesítő hatóság* PEM jelszavát. Nagyon vigyázzunk, nehogy a most létrehozott ügyfélbizonyítványhoz tartozó PEM jelszót írjuk be! Miután megadtuk a helyes jelszót, a következőket kell látnunk:

```
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows:
...
```

A fenti sorok után az ügyfél előbb elkészített bizonyítványához megadott információk jelennek meg. A program értesítést küld, ha valamilyen hiba történt. Ha minden rendben

ment, akkor az SSLeay megerősíti, hogy aláírja a bizonyítványt, és elhelyezi az adatbázisban. Ezzel létrehoz egy bejegyzést a */etc/certificates/newcerts* könyvtárban.

Az eddigi műveletek végeredményeként az aktuális könyvtárban létrejött a *phoenix.key* és a *phoenix.pem* fájl. Az ügyfél ezeket a fájlokat kapja meg, amikor az SSL-hez konfigurált Samba kiszolgálóhoz kapcsolódik, és az SSL Proxy is ezeket a fájlokat használja.

A Samba kiszolgáló konfigurálása

Következő lépésként az alábbi beállítások felvételével módosítanunk kell a Samba konfigurációs fájlját. A beállítások elkészítésénél abból indultunk ki, hogy a hitelesítő hatóság számára az */etc/certificates* könyvtárban hoztuk létre a bizonyítványok fájljait:

```
[global]
    ssl = yes
    ssl server cert = /etc/certificates/cacert.pem
    ssl server key = /etc/certificates/private/cakey.pem
    ssl CA certDir = /etc/certificates
```

Most le kell állítanunk, majd kézzel újra kell indítanunk a Samba démonjait:

```
# nmbd -D
# smbd -D
Enter PEM pass phrase:
```

A Samba démonjainak elindításához be kell írnunk a hitelesítő hatóság jelszavát. Megjegyezzük, hogy ez problémát okozhat, ha a programot a szokásos módokon kell elindítani, de az olyan fejlett szkript nyelvek, mint az Expect vagy a Python segítségével megkerülhető ez az akadály.

Tesztelés az smbclient programmal

A Samba helyes működését jól tesztelhetjük az *smbclient* programmal. A Samba kiszolgálón adjuk ki az alábbi parancsot (a megosztás és a felhasználó helyett a tényleges megosztás és a tényleges felhasználó nevét írjuk be):

```
# smbclient //hydra/megosztás -U felhasználó
```

Különböző hibavizsgáló sorok jelennek meg, majd egy olyan sor, amely jelzi a megállapodás szerinti titkosítást, mint például:

```
SSL: negotiated cipher: DES-CBC3-SHA
```

Ezt követően írjuk be a jelszavunkat, és a szokásos módon kapcsolódjunk a megosztáshoz. Ha ez sikerül, akkor biztosak lehetünk abban, hogy a Samba helyesen támogatja az SSL kapcsolatokat. Most még az ügyfélnél is el kell végezni a beállításokat.

Az SSL Proxy telepítése

Az SSL Proxy önálló programként bináris és forráskód alakban is hozzáférhető. A program a *http://obdev.at/Products/sslproxy.html* webhelyről tölthető le.

Miután letöltöttük, ugyanúgy konfigurálhatjuk és fordíthatjuk le, mint a Sambát. A példánkban Windows NT rendszeren mutatjuk be a konfigurálását, de majdnem pontosan ugyanezeket a lépéseket kell elvégezni Unix rendszerben is. Ügyeljünk arra, hogy a most következő lépéseket adminisztrátorként (rendszergazdaként) hajtsuk végre.

Ha a bináris fájlokat töltöttük le egy Windows NT rendszerre, akkor az alábbi fájloknak kell lenniük a választott könyvtárban:

- *cygwinb19.dll*
- *README.TXT*
- *sslproxy.exe*
- *dummyCert.pem*

A fentiek közül csak az SSLProxy végrehajtható fájl az érdekes most. Másoljuk át a korábban az ügyfél részére létrehozott *phoenix.pem* és *phoenix.key* fájlokat ugyanabba a könyvtárba, ahol az SSLProxy végrehajtható fájl van. Ügyeljünk arra, hogy a könyvtárt elrejtsük más felhasználók kíváncsi szeme elől.

Következő lépésként biztosítanunk kell, hogy a Windows NT gép képes legyen a Samba kiszolgáló NetBIOS nevének feloldására. Ez azt jelenti, hogy vagy lennie kell a hálózatban egy WINS kiszolgálónak (ezt a feladatot a Samba kiszolgáló is elláthatja, ha felvettük a `wins support = yes` beállítást), vagy a névnek szerepelnie kell a rendszer megfelelő *hosts* fájljában (a WINS kiszolgálóról a 7. fejezetben volt szó).*

Végül indítsuk el az SSPL Proxy programot az alábbi paranccsal. Itt is feltételezzük, hogy `hydra` a Samba kiszolgáló neve.

```
# C:\SSLProxysslproxy -l 139 -R hydra -r 139 -n -c phoenix.pem -k
phoenix.key
```

A parancs paraméterei azt közlik a programmal, hogy figyeljen a 139-es portra érkező kapcsolatkérésekre, és irányítsa át ezeket a `hydra` nevű gép 139-es portjára. Emellett a programnak a *phoenix.pem* és *phoenix.key* fájlokat kell használnia az SSL kapcsolat kezdeményezéséhez szükséges bizonyítvány és kulcsok generálásához. A parancsra az SSL Proxy a következő választ küldi:

```
Enter PEM pass phrase:
```

Írjuk be az ügyfél kulcspárjához tartozó PEM jelszót. Erre a következő választ kapjuk:

```
SSL: No verify locations, trying default
proxy ready, listening for connections
```

* Ha az SSL Proxy programot Unix kiszolgálón futtatjuk, akkor biztosítsuk, hogy feloldható legyen a Samba kiszolgáló DNS neve.

Ezzel a kiszolgáló kész a kérelmek fogadására. A parancsot akár a Unixon, akár a Windows NT-n elhelyezhetjük egy indító szekvenciába, ha minden bejelentkezéshez használni akarjuk. Gondoskodjunk arról, hogy az NT kiszolgálóhoz tartozó összes ügyfél (beleértve magát az NT kiszolgálót is) erre a kiszolgálóra, és ne a Samba kiszolgálóra mutasson.

Miután végeztünk a fenti telepítéssel, próbáljunk kapcsolatot teremteni az NT proxy kiszolgálón keresztül. Tapasztalni fogjuk, hogy szinte észrevehetetlen a közbeiktatott kiszolgáló jelenléte.

Az SSL konfigurációs beállításai

Az A.1. táblázatban összefoglaltuk az SSL konfigurációs beállításokat. Megjegyezzük, hogy ezek mindegyike globális hatókörű, vagyis ezeket a beállításokat a konfigurációs fájl `[global]` szakaszába kell felvenni.

A.1. táblázat. Az SSL beállítási lehetőségei

Beállítás	Paraméterek	Funkció	Alapértelmezett érték	Hatókör
`ssl`	Boolean érték	Jelzi, hogy engedélyezett-e a Sambában az SSL üzemmód.	`no`	Globális
`ssl hosts`	Karakterlánc (címlista)	Megadja azon gazdagépek listáját, amelyeknek mindig az SSL használatával kell kapcsolódniuk.	Nincs	Globális
`ssl hosts resign`	Karakterlánc (címlista)	Megadja azon gazdagépek listáját, amelyek sohasem használják az SSL-t a kapcsolódáshoz.	Nincs	Globális
`ssl CA certDir`	Karakterlánc (teljes elérési út)	Megadja a bizonyítványokat tároló könyvtárat.	Nincs	Globális
`ssl CA certFile`	Karakterlánc (teljes elérési út)	Megadja a Samba összes bizonyítványát tároló fájlt.	Nincs	Globális
`ssl server cert`	Karakterlánc (teljes elérési út)	Megadja a kiszolgáló bizonyítványának helyét.	Nincs	Globális
`ssl server key`	Karakterlánc (teljes elérési út)	Megadja a kiszolgáló privát kulcsának tárolási helyét.	Nincs	Globális
`ssl client cert`	Karakterlánc (teljes elérési út)	Megadja az ügyfél bizonyítványának tárolási helyét.	Nincs	Globális
`ssl client key`	Karakterlánc (teljes elérési út)	Megadja az ügyfél privát kulcsának tárolási helyét.	Nincs	Globális

A.1. táblázat folytatása

Beállítás	Paraméterek	Funkció	Alapértelmezett érték	Hatókör
`ssl require client-cert`	Boolean érték	Azt jelzi, hogy megköveteli-e a Samba, hogy minden ügyfelének legyen bizonyítványa.	`no`	Globális
`ssl require server-cert`	Boolean érték	Azt jelzi, hogy magának a kiszolgálónak kell-e rendelkeznie bizonyítvánnyal.	`no`	Globális
`ssl ciphers`	Karakterlánc	Megadja a protokoll egyeztetése során használandó titkosító eljárást.	Nincs	Globális
`ssl version`	`ssl2` vagy `3`, `ssl3` vagy `tls1`	Megadja az SSL használandó verzióját.	`ssl2` vagy `3`	Globális
`ssl compatibility`	Boolean érték	Jelzi, hogy szükség van-e az SSL egyéb megvalósításaival való kompatibilitásra.	`no`	Globális

ssl

Ezzel a beállítással konfigurálható úgy a Samba, hogy az SSL protokollt használja a maga és az ügyfelei közötti kapcsolatokban. A beállításhoz alapértelmezés szerint a `no` érték tartozik, amit az alábbi módon változtathatunk meg:

```
[global]
    ssl = yes
```

Ne feledjük, hogy a beállítás használatához lennie kell egy proxy kiszolgálónak, amelyen keresztül a Windows 95/98 ügyfelek kapcsolódhatnak a Samba kiszolgálóhoz.

ssl hosts

Ebben a beállításban azokat a gazdagépeket sorolhatjuk fel, amelyeknek kötelező az SSL használata. A beállítás szintaxisa ugyanaz, mint a `hosts allow` és a `hosts deny` konfigurációs beállításoké. Például:

```
[global]
    ssl = yes
    ssl hosts = 192.168.220.
```

A fenti beállítással azt írjuk elő, hogy mindazoknak a gazdagépeknek, amelyek a 192.168.220 alhálózatba tartoznak, SSL kapcsolatokat kell használniuk az ügyfelekkel való kommunikáció során. Ez az eljárás akkor hasznos, ha tudjuk, hogy különböző kapcsolatok jönnek létre az alhálózat és más, nem megbízható hálózatok, mint például az internet között.

Ha sem ezt, sem az `ssl hosts resign` beállítást nem adjuk meg, és az `ssl` beállításhoz a `yes` érték tartozik, akkor a Samba az összes ügyfele számára csak SSL kapcsolatokat engedélyez.

ssl hosts resign

Ebben a beállításban azokat a gazdagépeket sorolhatjuk fel, amelyek nincsenek kényszerítve SSL kapcsolatok használatára. A beállítás szintaxisa ugyanaz, mint a `hosts allow` és a `hosts deny` konfigurációs beállításoké. Például:

```
[global]
    ssl = yes
    ssl hosts resign = 160.2.310. 160.2.320.
```

A fenti példában azt írjuk elő, hogy azoknak a gazdagépeknek, amelyek a 160.2.310 vagy a 160.2.320 alhálózatokba tartoznak, nem kell SSL kapcsolatokat használniuk az ügyfelekkel való kommunikáció során. Ha sem ezt, sem az `ssl hosts` beállítást nem adjuk meg, és az `ssl` beállításhoz a `yes` érték tartozik, akkor a Samba az összes ügyfele számára csak SSL kapcsolatokat engedélyez.

ssl CA certDir

Ebben a beállításban azt a könyvtárat adhatjuk meg, amely a hitelesítő hatóság (CA) által kiállított bizonyítványokat tartalmazza. A Samba ezeket a bizonyítványokat használja az ügyfelei hitelesítéséhez A könyvtárban minden egyes hitelesítő hatósághoz külön fájlnak kell tartoznia. A könyvtárban lévő egyéb fájlok figyelmen kívül maradnak. Például:

```
[global]
    ssl = yes
    ssl hosts = 192.168.220.
    ssl CA certDir = /usr/local/samba/cert
```

A beállításhoz alapértelmezés szerint nem tartozik érték. E beállítás helyett az `ssl CA certFile` beállítást is használhatjuk, ha a hitelesítő hatóságokkal kapcsolatos összes adatot ugyanabban a fájlban akarjuk tárolni.

ssl CA certFile

Ebben a beállításban azt a fájlt adhatjuk meg, amely a hitelesítő hatóság (CA) által kiállított bizonyítványokat tartalmazza. A Samba ezeket a bizonyítványokat használja az ügyfelei hitelesítéséhez. Ez a beállítás annyiben különbözik az `ssl CA certDir` beállítástól, hogy csak egyetlen fájlt hoz létre, amely az összes hitelesítő hatóság adatait tartalmazza. Példa a beállítás használatára:

```
[global]
    ssl = yes
    ssl hosts = 192.168.220.
    ssl CA certFile = /usr/local/samba/cert/certFile
```

A beállításhoz alapértelmezés szerint nem tartozik érték. E beállítás helyett az `ssl CA certDir` beállítást is használhatjuk, ha a hitelesítő hatóságokkal kapcsolatos adatokat külön fájlokban akarjuk tárolni.

ssl server cert

Ez a beállítás a kiszolgáló bizonyítványának a helyét adja meg. A beállítás használata kötelező, mert a kiszolgálóhoz kell tartoznia bizonyítványnak az SSL használatához. Például:

```
[global]
    ssl = yes
    ssl hosts = 192.168.220.
    ssl CA certFile = /usr/local/samba/cert/certFile
    ssl server cert = /usr/local/samba/private/server.pem
```

A beállításhoz alapértelmezés szerint nem tartozik érték. Figyeljük meg, hogy a bizonyítvány a kiszolgáló privát kulcsát is tartalmazhatja.

ssl server key

Ez a beállítás a kiszolgáló privát kulcsának a helyét határozza meg. Gondoskodnunk kell arról, hogy a fájlt tároló helyhez a rootfelhasználón kívül senki más ne férhessen hozzá. Példa:

```
[global]
    ssl = yes
    ssl hosts = 192.168.220.
    ssl CA certFile = /usr/local/samba/cert/certFile
    ssl server key = /usr/local/samba/private/samba.pem
```

A beállításhoz alapértelmezés szerint nem tartozik érték. Figyeljük meg, hogy a bizonyítvány a kiszolgáló privát kulcsát is tartalmazhatja.

ssl client cert

Ez a beállítás az ügyfél bizonyítványának a helyét határozza meg. A bizonyítványt a Samba kiszolgáló az `ssl require clientcert` beállítással kérheti, de az *smbclient* is használhatja azt. Például:

```
[global]
    ssl = yes
    ssl hosts = 192.168.220.
    ssl CA certFile = /usr/local/samba/cert/certFile
    ssl server cert = /usr/local/ssl/private/server.pem
    ssl client cert= /usr/local/ssl/private/clientcert.pem
```

A beállításhoz alapértelmezés szerint nem tartozik érték.

ssl client key

Ez a beállítás az ügyfél privát kulcsának a helyét határozza meg. Gondoskodnunk kell arról, hogy a fájlt tároló helyhez a rootfelhasználón kívül senki más ne férhessen hozzá. Példa:

```
[global]
    ssl = yes
    ssl hosts = 192.168.220.
    ssl CA certDir = /usr/local/samba/cert/
    ssl server key = /usr/local/ssl/private/samba.pem
    ssl client key = /usr/local/ssl/private/clients.pem
```

A beállításhoz alapértelmezés szerint nem tartozik érték. A beállítást csak akkor kell felvennünk, ha van az ügyfélnek bizonyítványa.

ssl require clientcert

Ezzel a beállítással azt jelezhetjük, hogy kell-e rendelkeznie egy ügyfélnek bizonyítvánnyal. Ha előírjuk a bizonyítvány kötelező meglétét, akkor a kiszolgáló végignézi az `ssl CA certDir` vagy az `ssl CA certFile` beállításokban megadott helyeket, hogy meggyőződjön arról, van-e érvényes bizonyítványa az ügyfélnek, majd ez alapján hitelesíti, és engedélyezi a kapcsolódását. Például:

```
[global]
    ssl = yes
    ssl hosts = 192.168.220.
    ssl CA certFile = /usr/local/samba/cert/certFile
    ssl require clientcert = yes
```

Javasolható, hogy mindazoktól az ügyfelektől megköveteljük a bizonyítvány meglétét, akik kapcsolódhatnak a Samba kiszolgálóhoz. A beállításhoz alapértelmezés szerint a `no` érték tartozik.

ssl require servercert

Ezzel a beállítással azt jelezhetjük, hogy kell-e rendelkeznie a kiszolgálónak bizonyítvánnyal. Ezt a bizonyítványt az *smblient* program is használja. A beállításhoz egy egyszerű Boolean érték rendelhető, például:

```
[global]
    ssl = yes
    ssl hosts = 192.168.220.
    ssl CA certFile = /usr/local/samba/cert/certFile
    ssl require clientcert = yes
    ssl require servercert = yes
```

Bár azt javasoltuk, hogy mindazoktól az ügyfelektől megköveteljük a bizonyítvány meglétét, akik kapcsolódhatnak a Samba kiszolgálóhoz, a kiszolgálónál ez nem kötelező, de ugyancsak ajánlott. A beállításhoz alapértelmezés szerint a `no` érték tartozik.

ssl ciphers

Ebben a beállításban azokat a titkosító eljárásokat adhatjuk meg, amelyeket a Samba az SSL kapcsolat létrehozásának egyeztetési fázisában kiválaszthat. A Samba az alábbi eljárások közül választhat:

```
DEFAULT
DES-CFB-M1
NULL-MD5
RC4-MD5
EXP-RC4-MD5
RC2-CBC-MD5
EXP-RC2-CBC-MD5
IDEA-CBC-MD5
DES-CBC-MD5
DES-CBC-SHA
DES-CBC3-MD5
DES-CBC3-SHA
RC4-64-MD5
NULL
```

Ha nem ismerjük kellőképpen az SSL protokollt, és nem akarjuk, hogy a Samba kötelezően egy adott titkosító eljárást használjon, akkor inkább ne vegyük fel ezt a beállítást.

ssl version

Ebben a beállításban az SSL azon verzióját írhatjuk elő, amelyet a Sambának a titkosított kapcsolatok kezeléséhez használnia kell. A beállításhoz alapértelmezés szerint az `ssl2or3` karakterlánc tartozik, ami az SSL protokollnak akár a 2-es, akár a 3-as verzióját jelenti attól függően, hogy a kiszolgáló és az ügyfél a kapcsolatfelvétel egyeztetése során melyikben állapodik meg. Ugyanakkor a protokoll valamely meghatározott verzióját is előírhatjuk az alábbi módon:

```
[global]
    ssl version = ssl3
```

Ezzel kapcsolatban is az mondható, hogy ha nem ismerjük kellőképpen az SSL protokollt, és nem akarjuk, hogy a Samba kötelezően egy adott verzióját használja, akkor inkább ne vegyük fel ezt a beállítást.

ssl compatibility

Ezzel a beállítással azt írhatjuk elő, hogy úgy legyen-e konfigurálva a Samba, hogy az SSL más verzióit is tudja használni. Mivel a könyv írásának idején az SSL-nek nem léteztek más verziói, a beállításnak nincs jelentősége, ezért legjobb, ha meghagyjuk az alapértelmezett `no` értékét.

B

A Samba teljesítményének finomhangolása

Ebben a függelékben olyan eljárásokkal ismerkedünk meg, amelyek segítségével elvégezhetjük a Samba teljesítményének behangolását és a rendszer igényeink szerinti méretezését. A teljesítmény hangolásán itt azt értjük, hogy megkeressük a rendszerben meglévő szűk keresztmetszeteket, és különböző lépésekkel megszüntetjük ezeket. A rendszer méretezése viszont olyan eljárásokat jelent, amelyek révén elejét vehetjük a szűk kapacitások kialakulásának. Normál körülmények között az ilyen problémákkal nem kell törődnünk. A Samba egy abszolút hangolatlan kiszolgálón is jól elboldogul a felhasználók kisebb csoportjával. Ha viszont helyesen van behangolva a kiszolgáló, akkor a Samba legkevesebb kétszer annyi ügyfél kiszolgálására is képes. A fejezetben az ilyen teljesítménynövelő és helyes méretezési eljárásokról olvashatunk, amelyek segítségével a felső határáig növelhetjük a Samba képességeit.

Egy egyszerű teszt

Honnan állapíthatjuk meg, hogy elfogadható a rendszerünk teljesítménye? Egyszerű összehasonlítást végezhetünk a Samba és az FTP között. A B.1. táblázat kilobájt/s-ban mutatja két kiszolgálón az átbocsátási sebességeket: egy közepes méretű Sun SPARC Ultra és egy kis méretű Linux Pentium kiszolgálón.

B.1. táblázat. Egyszerű összehasonlítás

Parancs	**FTP**	**Hangolatlan Samba**	**Behangolt Samba**
Sparc get	1014.5	645.3	866.7
Sparc put	379.8	386.1	329.5
Pentium get	973.27	N/A	725
Pentium put	1014.5	N/A	1100

Ha ugyanezeket a teszteket a saját kiszolgálónkon is lefuttatjuk, akkor valószínűleg más számokat kapunk. Ugyanakkor az is valószínű, hogy a Samba és az FTP átbocsátása közötti arány ezekhez hasonló lesz. Nem célszerű, ha a Samba teljesítményét csak az FTP-vel hasonlítjuk össze. Aranyszabályként a következőt alkalmazhatjuk: ha a Samba lényegesen lassabb, mint az FTP, akkor ideje, hogy behangoljuk.

Azt gondolhatnánk, hogy ugyanezt a tesztet a Samba és az NFS összehasonlításához is használhatjuk. A valóságban azonban az ilyen összehasonlításoknak nincs sok értelme. Attól függően, hogy az NFS melyik verzióját használjuk, és mennyire jól van behangolva, a Samba akár gyorsabb, akár lassabb is lehet az NFS-nél. A szerzők azt tapasztalták, hogy általában a Samba a gyorsabb, de ez félrevezető lehet; az NFS más algoritmust használ mint a Samba, így azok a hangolások, amelyek optimálisak az NFS számára, hátrányosak lehetnek a Sambára nézve. Ugyanakkor viszont ha a Sambát egy jól behangolt NFS kiszolgálón futtatjuk, meglehetősen rossz eredményt kaphatunk.

Ennél népszerűbb teszt Ziff-Davis *NetBench* programja, amely számos olyan felhasználót szimulál az ügyfélgépeken, akik szövegszerkesztőt futtatnak, és Samba kiszolgálón tárolt adatokhoz férnek hozzá. Bár ez sem tökéletes eljárás, arra azért jó, hogy azonos kiszolgálókat össze lehessen hasonlítani. Jeremy Allison 1998 novemberében elvégzett tesztjeiben a Samba 2.0 egy SGI multiprocesszoros rendszeren nagyobb teljesítményt ért el, mind egy NT Server 4.0 (2-es javítócsomaggal) egy hasonlóan csúcsminőségű Compaq gépen. Ugyanezt az eredményt erősítette meg egy azonos hardvereken futó NT és Linux Sm@rt Reseller tesztje 1999 februárjában.

1999 áprilisában a Mindcraft tesztlaboratórium olyan tesztről adott ki jelentést, amely szerint a Samba egy négyprocesszoros Linux gépen lényegesen lassabb volt, mint egy ugyanilyen gépen futó Windows NT natív fájlkiszolgálója. Az Open Source közösség ugyan nagyon élesen bírálta az eredeti jelentést, mondván, hogy a teszt a Microsoft megbízásából készült, és a rendszereket a Windows NT-re hegyezték ki, egy későbbi, ennél kiegyensúlyozottabb teszt valóban azt bizonyította, hogy a Linux egyes részein javítani kell a teljesítményt, különösen többprocesszoros környezetben. Kevés szó esett magáról a Sambáról. A Sambáról tudható, hogy jól méretezhető többprocesszoros rendszerekben, és az SGI 0200 négyprocesszoros gépén meghaladja a teljesítménye a 440 MB/s sebességet, jóval felülmúlva a Mindcraft által kimutatott 310 MB/s sebességet.

Az NT és a PC hardverek gyorsabbá válásával természetesen változhatnak a teljesítményviszonyok, de ezzel együtt a Samba is javul. Így például a Samba 1.9.18 csak akkor bizonyult gyorsabbnak, ha az ügyfelei száma meghaladta a 35-öt. Ezzel szemben a Samba 2.0-nál már nem számít az ügyfelek száma. Röviden összefoglalva a Samba az iparban található legjobb hálózati szoftverekkel összehasonlítva is nagyon versenyképes, és egyre jobb lesz.

A könyv nyomdába adásának idején Andrew Tridgell bejelentette egy Sambához és SMB hálózatokhoz használható tesztprogram alfateszt verzióját. A programról az *ftp://samba.org/pub/tridge/dbench/README* címen olvashatók további tudnivalók. A Samba fejlesztőcsoportjától is várhatók további teljesítményvizsgáló tesztek.

A Samba finomhangolása

Ennyi bevezető után lássuk, miként tehetünk egy gyors hálózati programcsomagot még fürgébbé.

Nyúzópróba

A következőkben néhány egyszerűbb vizsgálatnak vetjük alá a Sambát. Mivel a Samba kiszolgálók elsődleges feladata fájlok átvitele, csak az átviteli sebességet fogjuk vizsgálni, a

különböző eseményekre való válaszadási időkkel nem foglalkozunk. A fájlátviteli sebességeket eléggé egyszerűen mérhetjük, és a Samba válaszadási képességei nem olyan roszszak, hogy emiatt kifinomultabb vizsgálatokat kelljen végeznünk.

A vizsgálatok elvégzéséhez a következő stratégiát választjuk:
- Keresünk egy ésszerű méretű fájlt, amit lemásolunk, és egy olyan programot, amely kijelzi a másolási sebességet (ilyen lehet például az *smbclient* program).
- Keresünk egy nyugalmas (tipikus) időt a teszt elvégzéséhez.
- Néhányszor előzetesen lefuttatjuk az egyes teszteket, hogy feltöltsük a puffereket.
- Különböző időkben futtatjuk a teszteket, és figyeljük, nem fordulnak-e elő szokatlan események.
- Részletesen rögzítjük a futások eredményét.
- Összehasonlítjuk az eredmények átlagát a várt értékekkel.

Miután a fenti lépéseket elvégezve megkaptuk a kiinduló adatokat, egyenként módosítunk különböző paramétereket, és újra elvégezzük a fenti vizsgálatokat. A fejezet végén található üres táblázatok arra szolgálnak, hogy beírjuk a kapott eredményeket.

Kipróbálható beállítások

Szó szerint ezerszámra hozhatunk létre a Sambában olyan beállításkombinációkat, amelyek szerint kereshetjük a tökéletesen működő rendszert. Ha viszont nem kifejezetten rendszergazdai szempontból közelítjük meg a kérdést, akkor a vizsgálatokat az alábbiakban felsorolt beállításokra szűkíthetjük. Ezek azok a beállítások, amelyek a legnagyobb valószínűséggel befolyásolják a rendszer általános teljesítményét. A felsorolási sorrendjük nagyjából tükrözi a fontossági sorrendjüket.

Naplózási szint

Ennek a beállításnak nyilvánvaló a teljesítményt befolyásoló szerepe. A naplózási szint (`log level` vagy `debug level` beállítás) növelése jó módszer egy hiba megkereséséhez, de csak abban az esetben, ha nem éppen a teljesítménnyel kapcsolatos valamilyen hibának akarunk utánajárni. Amint a 4. fejezetben említettük, a Samba a 3-as vagy az ennél nagyobb naplózási szintek mellett üzenetek ezreit generálja, és ezek lemezre írása lassú művelet. Az *smbclient* és ftp-s tesztjeinkben azzal, hogy 0-ról 3-ra növeltük a szintet, 645,3 KB/s-ról 622,2 KB/s-ra csökkent az átviteli sebesség, ami durván öt százalékos romlásnak felel meg. A nagyobb szintek még nagyobb teljesítménycsökkenést okoznának.

Socket beállítások

Következő feladatként a `socket options` konfigurációs beállításokat kell vizsgálnunk. Bár ezek igazából a gazdarendszer hangolásával függnek össze, de kapcsolatonként állíthatók be, és a Samba módosíthatja ezeket azokon a socketokon, amelyeket az *smb.conf* fájl `[global]` szakaszába felvett `socket options = socketlista` beállítás specifikál. Nem minden szoftvergyártó támogatja ezeket a beállításokat – a szoftver dokumentációjában tájékozódjunk a *setsockopt* vagy a *socket* részleteiről.

A fontosabb beállítási lehetőségek:

`TCP_NODELAY`
: Hagyjuk, hogy a kiszolgáló annyi csomagot küldjön, amennyi szükséges, hogy alacsony szinten tartsuk a késéseket. Ezt a telnet kapcsolatokban használják jó válaszidők elérése céljából, de akkor is jó eredménnyel jár, ha rövidebbek a kérések, vagy ha késleltetve vannak a visszaigazolások (úgy tűnik, hogy ez a helyzet a Microsoft TCP/IP-nél). Ezzel önmagában 30–50 százalékkal növelhető a teljesítmény. A Samba 2.0.4 verzióban a `socket options` beállításhoz alapértelmezés szerint a `TCP_NODELAY` érték tartozik.

`IPTOS_LOWDELAY`
: Ezzel a beállítással ugyancsak csökkenthető a késlekedési idő, de nem a kiszolgálón, hanem a forgalomirányítón és más rendszerekben. Az IPTOS beállítások újdonságok, amelyeket nem támogat minden operációs rendszer és forgalomirányító. Ha a rendszerünk támogatja ezt, akkor minden olyan helyen használjuk az `IPTOS_LOWDELAY` beállítást, ahol a `TCP_NODELAY` beállítást is használjuk.

`SO_SNDBUF és SO_RCVBUF`
: A küldő és fogadó pufferek méretét gyakran nagyobbra választhatjuk annál, mint amit az operációs rendszer állított be. Ezzel némileg növelhető a sebesség (amíg el nem érjük azt a pontot, ahol eltűnnek a válaszok).

`SO_KEEPALIVE`
: Ez a beállítás időszakos (négy óránkénti) vizsgálatot indít abból a célból, hogy lássa, nem tűnt-e el az ügyfél. A lejárt kapcsolatok némileg jobban kezelhetők a Samba `keepalive` és `dead time` beállításaival. Mindhárom beállításnak az a feladata, hogy lezárja a megszakadt kapcsolatokat, és a használaton kívüli memóriaterületeket, valamint a processztábla-bejegyzéseket ismét az operációs rendszer rendelkezésére bocsássa.

További socketbeállításokat is megvizsgálhatunk (például `SO_SNDLOWAT`), de ezek gyártóról gyártóra változhatnak. Aki a Samba hangolásával kapcsolatban többet szeretne tudni ezekről a beállításokról, a *TCP/IP Illustrated* folyóiratban tájékozódhat a részletekről.

read raw és write raw

Ezek a beállítások jelentős mértékben befolyásolják a teljesítményt. Lehetővé teszik ugyanis, hogy a Samba egyetlen SMB kérelem során akár 64 kilobájtos írási vagy olvasási műveletet hajtson végre a hálózaton. Ehhez az SMB `SMBreadraw` és `SMBwriteraw` csomagszerkezetét is a legnagyobb méretre kell állítani. Jegyezzük meg, hogy ez nem azonos a Unix *raw read* műveletével. A Unixban ez a fogalom a lemezek fájlrendszer használata nélküli olvasását jelenti, ami meglehetősen eltér attól, amiről itt a Samba kapcsán szólunk.

A múltban egyes ügyfélprogramok lefagytak, ha megpróbáltuk volna használni a `read raw` beállítást. A mai ismeretek szerint már nincsenek ilyen problémák. Alapértelmezés szerint mind a `read raw`, mind a `write raw` beállításhoz a `yes` érték tartozik, és célszerű meghagyni ezt, hacsak nincs egy hibás ügyfélprogramunk.

Opportunista zárolás

Az opportunista zárolások (röviden: *oplocks*) lehetővé teszik, hogy az ügyfelek helyileg gyorsítsák a fájlokat, ami 30 százalékos teljesítményjavulást is eredményezhet. A beállításhoz alapértelmezés szerint a `yes` érték tartozik. A csak olvasható fájlok vonatkozásában a `fake oplocks` beállításnak ugyanez a funkciója anélkül, hogy ténylegesen sor kerülne a gyorsításra. Ha olyan fájljaink lennének, amelyek nem gyorsíthatók, kikapcsolhatjuk az ilyen zárolásokat.

Az adatbázis-fájlokat sohase gyorsítsuk, és az olyan fájlokat sem, amelyek mind a kiszolgálón, mind az ügyfélnél frissülhetnek, és amelyek módosulásait azonnal látni kell. A `veto oplock files` beállítás teszi lehetővé, hogy akár a fájlok felsorolásával, akár helyettesítő karaktereket is tartalmazó minta megadásával elkerüljük az ilyen fájlok gyorsítását. Az opportunista zárolásokat megosztásokra ki is kapcsolhatjuk, ha sok olyan fájlunk van, amelyeknek nem akarjuk engedélyezni a gyorsítását. Az ilyen zárolásokról az 5. fejezetben volt részletesebben szó.

IP csomagméret (MTU)

A hálózatokon általában korlátozva van az egyidejűleg átvihető csomagok mérete. Az ilyen méret angol neve Maximum Segment Size, illetve ha a méretbe a csomag fejlécét is beleértjük, akkor Maximum Transport Unit (MTU) a neve. A Samba nem adja meg ezt az MTU méretet, viszont szüksége van olyan `max xmit` beállításra, amelyhez az MTU-nál nagyobb érték tartozik, mert különben csökkenne a teljesítménye. Erről további részletek az alábbi megjegyzésben olvashatók. Az MTU egy Ethernet kártyán általában 1500 bájtra, míg egy FDDI hálózaton 4098 bájtra van beállítva. Általánosságban az mondható, hogy a túlságosan alacsony érték lassítja a forgalmat, míg a túlságosan nagy érték hirtelen teljesítményromlást idézhet elő a csomagok széttöredezése és újbóli elküldése miatt.

Ha a hálózati forgalmunk forgalomirányítón keresztül bonyolódik le, akkor egyes rendszerek soros kapcsolatnak (például T1-nek) tekinthetik ezt, és az MTU-t nagyjából 536 bájtra állítják be. A Windows 95 is elköveti ezt a hibát, aminek következtében a közeli ügyfelekkel való kommunikáció során ugyan megfelelő a teljesítmény, de a forgalomirányító túloldalán lévő ügyfeleknél már észrevehetően kisebb a sebesség. Ha az ügyfél az ellenkező hibát követi el, és nagy MTU értéket állít be az olyan kapcsolatokra, amelyek kisebb értéket igényelnek, akkor a csomagok szegmensekre szakadnak szét. Ezzel csökken az átviteli sebesség, és a hálózaton bekövetkező bármilyen hiba miatt újból el kell küldeni a szegmenseket, ami jelentősen rontja a Samba teljesítményét. Szerencsére módosíthatjuk a Windows MTU értékét, és mindkét hibát elkerülhetjük.

A TCP fogadási ablaka

A TCP/IP úgy működik, hogy kisebb csomagokra bontja a küldendő adatokat, és az egyes csomagokat küldi el az egyik gépről a másikra. Miután mindegyik csomagot feladta, az utolsó csomaggal együtt egy ellenőrzőösszeget is kiküld, amely alapján a fogadó fél ellenőrizni tudja, hogy nem volt hiba az átvitelben. Elméletileg a fogadó félnek minden egyes

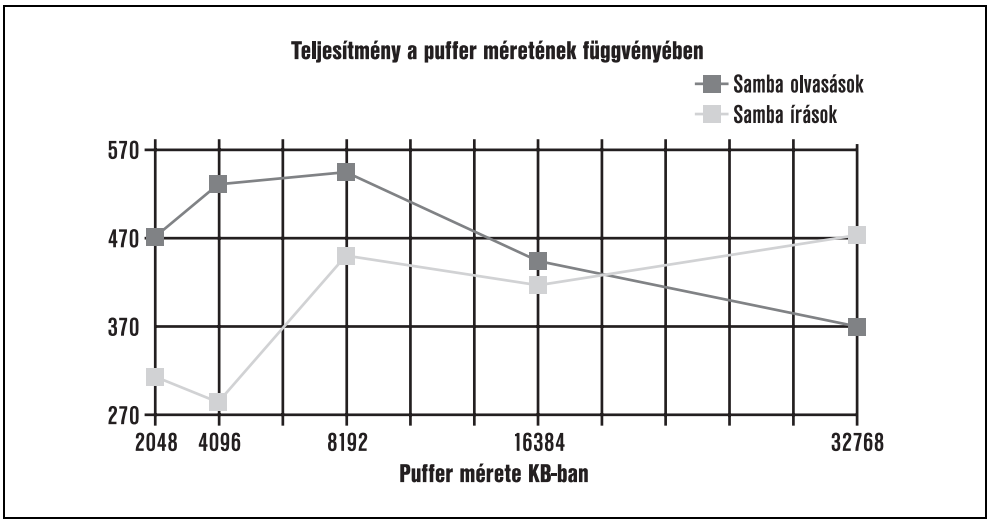
csomag beérkezését és ellenőrzését követően vissza kellene küldenie egy másik csomagot, amelyben közölné a feladóval, hogy „minden rendben megérkezett, jöhet a többi".

A forgalom gyorsítása érdekében azonban a TCP elfogad egy több csomagot is magában foglaló keretet, amivel lehetővé teszi, hogy a küldő fél megszakítás nélkül egyszerre több csomagot is elküldhessen, és ne kelljen minden egyes csomag elküldése után a visszaigazolásra várnia. (Így a visszaigazoló csomagok egy nagyobb csomagba foghatók össze, és egyszerre elküldhetők.) Ez a keret, amit *fogadási ablaknak* is neveznek, azoknak a bájtoknak a számát adja meg, amelyeket a küldő fél elküldhet, mielőtt megállna, és a visszaigazoló csomagok beérkezésére várna. Az MTU-hoz hasonlóan ez az ablak is automatikusan beállítódik a kapcsolat típusától függően. Ha túl kicsi az ablak, akkor fölöslegesen kell várakozni a visszaigazoló üzenetekre. A különböző operációs rendszerek közepes pufferméreteket állítanak be az egyes socketokhoz, hogy egyetlen program ne sajátíthassa ki magának az összes memóriát.

A puffer méretét bájtokban kell megadni a `socket options` beállításban, mint például `SO_SNDBUF=8192`. Egy példa a beállítás használatára:

```
socket options = SO_SNDBUF=8192
```

A különböző operációs rendszerek az alapértelmezett értéknél nagyobbakkal próbálkoznak: a SunOS 4.1.3 és az SVR4 a 4098-as értéket, míg az AIX, a Solaris és a BSD a 8192-16384 közötti értékeket használja. A 16384-es értéket megfelelő kiindulópontnak tekinthetjük: Steven a könyvében megemlít egy nem a Sambával végzett tesztet, amelynek során 40 százalékos javulást ért el. Kísérletezzünk különböző értékekkel, mert az is előfordulhat, hogy egy bizonyos határ fölött újból csökken a teljesítmény. Ez látható a B.1. ábrán, mely egy Linux rendszeren végzett teszt eredményét mutatja be.



B.1. ábra. Az SO_SNDBUF puffer mérete és a teljesítmény

A `socket options` beállításban az `O_SNDBUF` és az `SO_RCVBUF` pufferekhez ne rendeljünk az alapértelmezett értéknél kisebb értéket. A nagyobb értékek javíthatják a teljesítményt, de csak egy, a hálózattól függő pontig. Ha viszont ezt a határt túllépjük, akkor újból hirtelen csökkenhet a teljesítmény.

max xmit

A Sambában ez a beállítás közvetlen kapcsolatban áll az MTU és a fogadási ablak méretével. A beállításban annak a legnagyobb adatblokknak a méretét adhatjuk meg, amelyet a Samba egyetlen művelettel írni tud. Esetenként ezt a beállítást úgy is emlegetik, mint *write size* (írásméret), bár a Sambában nem ez a neve.

Mivel az egyes blokkokkal járó átlagos veszteség százalékaránya annál kisebb, minél nagyobbak a blokkok, a max xmit beállításhoz hagyományosan a lehető legnagyobb értéket rendelik. Ez a protokoll által megengedett legnagyobb érték, vagyis 64 kilobájt. A legkisebb érték, ami még nem okoz jelentős lassulást, a 2048 bájt. Ha megfelelően kis értéket állítunk be, akkor ez lesz a Samba által küldhető legnagyobb csomagméret felső határa. Ezzel szimulálhatunk kisebb MTU értékeket, ha egy megbízhatatlan hálózaton kell teszteket végeznünk.

read size

Ha a `max xmit` beállításnál azt mondtuk, hogy ezt írásméretnek is nevezik, akkor azt gondolhatnánk, hogy a `read size` (olvasásméret) beállítással annak a legnagyobb adatblokknak a méretét adhatjuk meg, amelynek az olvasását a Samba a hálózaton keresztül az ügyfeleinek engedélyezi. Ez azonban nem így van: ez a beállítás az *előreírás* (write ahead) műveletét váltja ki. A Samba ugyanis nem várja meg, hogy befejeződjön egy lemezről való olvasás vagy egy hálózatra történő írás művelete, hanem már azt megelőzően új műveletet indít. Ez a beállítás adja meg azt, hogy meddig várakozzon a Samba, mielőtt egy új, az előző (írási vagy olvasási) műveletet időben átlapoló írási vagy olvasási műveletet elindít.

A `read size` beállítás nem gyakorol különösebb befolyást a Unix rendszerre, hacsak nem adunk meg hozzá túlságosan kis értéket. Ebben az esetben észrevehető a rendszer lassulása. A beállításhoz alapértelmezés szerint a 2048-as érték tartozik, és 1024-nél nem lehet kisebb.

read prediction

Ez a beállítás mára elavult. Azt tette lehetővé, hogy a Samba előre olvashassa azokat a fájlokat, amelyeket csak olvasásra nyitottak meg az ügyfelek. A Samba 2.0 (és az 1.9 újabb kiadásai) már nem engedélyezik ezt a beállítást, mert ütközést okozhat az opportunista zárolásokkal.

Egyéb beállítások

Az alábbiakban felsorolunk néhány olyan beállítást, amelyek hibás megadása – a naplózási szintekhez hasonlóan – jelentősen befolyásolhatja a Samba teljesítményét.

`hide files`

A `hide files` beállításhoz mintát rendelhetünk azzal a céllal, hogy a Windows ügyfél rejtettként azonosítsa mindazokat a fájlokat, amelyek egyrészt megfelelnek az

adott mintának, másrészt be van kapcsolva a DOS rejtett attribútuma. A Sambának ekkor a könyvtárak kilistázásakor minden egyes fájlt össze kell vetnie a mintával, ami jelentősen lassítja a kiszolgáló működését.

`lpq cache time`
: Ha az `lpq` (a nyomtatás várakozási sorának tartalma) parancs végrehajtása sokáig tartana, akkor az `lpq cache time` beállításhoz nagyobb értéket rendeljünk, mint amennyi időt az `lpq` parancs a lefutásához igényel, hogy ne kelljen a Sambának új lekérdezést kezdeményeznie, amikor már fut egy másik. Az alapértelmezés szerinti 10 másodperc elfogadható érték.

`strict locking`
: A `strict locking` beállítással azt írhatjuk elő, hogy a Samba minden egyes hozzáféréskor vizsgálja a zárolásokat, és ne csak akkor, amikor egy ügyfél ezt kéri. A beállítás eredeti rendeltetése egy programhiba kiküszöbölése volt, amivel meg lehetett akadályozni, hogy a hibásan megírt DOS és Windows alkalmazások tönkretehessenek megosztott fájlokat. Ugyanakkor ez a beállítás lassítja a működést, és normál körülmények között kerülendő az alkalmazása.

`strict sync`
: A `strict sync` beállítás azt írja elő, hogy amikor a Samba felír egy csomagot a lemezre, várja meg a művelet befejeződését, ha az ügyfél bekapcsolta a csomagban a szinkronizáló bitet. A Windows 98 Intézője ezt a bitet minden egyes átküldött csomagban bekapcsolja, ezért ha a Sambában is bekapcsoljuk, akkor a Windows 98 ügyfelek azt hihetik, hogy iszonyúan lassúak a Samba kiszolgálók.

`sync always`
: Ez a beállítás arra utasítja a Sambát, hogy minden írást azonnal rögzítsen a lemezen. A beállításnak jó hasznát vehetjük, ha gyakran összeomlik a rendszerünk, de a teljesítményt illetően nagy árat kell fizetnünk ezért. Az SMB kiszolgálók általában az opportunista zárolásokat és az automatikus újrakapcsolódást használják az ilyen összeomlások káros mellékhatásainak elkerülése érdekében, ezért normál esetben nincs szükség erre a beállításra.

`wide links`
: A `wide links` beállítás kikapcsolásával elejét vehetjük annak, hogy a Samba az egyik lemezmegosztásban lévő szimbolikus hivatkozásokat olyan fájlokig kövesse, amelyek nincsenek az adott megosztásban. A beállítás alapértelmezés szerint be van kapcsolva, mert a hivatkozások követése a Unixban nem jelent biztonsági kockázatot. A kikapcsolása többletműveleteket jelent minden egyes nyitott fájlon. Ha kikapcsoljuk a `wide links` beállítást, akkor ne feledkezzünk meg a `getwd cache` beállítás bekapcsolásáról, hogy gyorsítsuk a szükséges adatok egy részét.
  Van még egy `follow symlinks` beállítás is, amelynek kikapcsolásával az összes szimbolikus hivatkozás követését megakadályozhatjuk. Ennek azonban a teljesítményt illetően nincs jelentősége.

`getwd cache`
: Ez a beállítás gyorsítja az aktuális könyvtár elérését, mert elkerüli a fájlszerkezetben való keresgélést. Jó lehetőség egy nyomtatókiszolgáló teljesítményének növelésére, vagy arra az esetre, ha kikapcsoltuk a `wide links` beállítást.

A szerzők ajánlásai

Az alábbiakban egy olyan *smb.conf* fájl következik, amely az eddig felsorolt teljesítménynövelő beállításokat tartalmazza. A jobb oldali oszlopban a megjegyzések olvashatók.

```
[global]
    log level = 1                       # 0 az alapbeállítás
    socket options = TCP_NODELAY IPTOS_LOWDELAY
    read raw = yes                      # alapbeállítás
    write raw = yes                     # alapbeállítás
    oplocks = yes                       # alapbeállítás
    max xmit = 65535                    # alapbeállítás
    dead time = 15                      # 0 az alapbeállítás
    getwd cache = yes
    lpq cache = 30
[okplace]
    veto oplock files = this/that/theotherfile
[badplace]
    oplocks = no
```

Samba kiszolgálók méretezése

A méretezés itt azt az eljárást jelenti, amelynek segítségével még azt megelőzően elkerülhetjük a szűk keresztmetszeteket, mielőtt előfordulnának. Kiindulásként azt kell tudnunk, hogy másodpercenként hány kérés érkezhet, és hány kilobájt átvitelére van szüksége az ügyfeleknek. Ezek ismeretében kell biztosítanunk azt, hogy a kiszolgáló mindegyik összetevője legalább ekkora teljesítményre legyen képes.

A szűk kapacitások

Szűk kapacitásként elsősorban a CPU, a lemez I/O műveletei és a hálózat jöhet számításba. A gépek többségén általában nem a CPU fogja vissza a teljesítményt. Egyetlen Sun SPARC 10 CPU másodpercenként 700–800 I/O műveletet tud indítani (és befejezni), ami 5600–6400 KB/s sebességet jelent átlag 8 KB-os adatcsomagok mellett (ez a puffer szokásos mérete). Egyetlen Intel Pentium 133 MHz-es CPU-nak ennél kisebb ugyan a teljesítménye, de ennek nem a CPU kisebb ereje, hanem a lassúbb gyorsító és buszinterfészek az okai. A célnak megfelelően megtervezett Pentium alapú kiszolgálók, mint például a Compaq kiszolgálók CPU-nként 700 művelet indítására is képesek, és akár négy CPU-val is dolgozhatnak.

Ezzel szemben a kevés memória könnyen szűk kapacitást okozhat; a Samba minden egyes processze 600–800 KB-ot igényel egy Intel Linux gépen, és még ennél is többet RISC processzorokon. Ha szűkös a memória, akkor megnövekszik a virtuális memória területe, és a lapozás csökkenti a teljesítményt. Solaris rendszerben, ahol ez mérésre került, az *smbd* programonként és megosztott könyvtárakként 2,6 MB-ot, továbbá kapcsolódó ügyfelenként plusz 768 KB-ot igényelt. Az *nmdb* 2,1 MB-ot, plusz az (egyetlen) segédprocessze további 496 KB-ot foglalt el.

A merevlemezek mindig is szűk keresztmetszetet jelentenek a másodpercenkénti I/O műveleteket illetően: egy 7200 fordulat/perc fordulatszámú SCSI lemez másodpercenként 70 műveletet tud végrehajtani, ami 560 KB/s teljesítménynek felel meg; egy 4800 fordulat/perc fordulatszámú lemez már csak 50 műveletre képes, ami 360 KB/s teljesítményt jelent. Egy IDE lemez még ennél is lassúbb. Ha a lemezek függetlenek, vagy RAID 0 elrendezés szerint vannak összefűzve, akkor egyenként 400–560 KB/s teljesítményre képesek, és ez a teljesítmény egyenes arányban nő a számuk növelésével. Ez azonban csak a RAID 0 elrendezésre (összefűzés) igaz – a RAID többi szintjén egyéb lemezadminisztráció növeli a terheket.

Az Ethernet (és más hálózatok) szintén szűk kapacitást jelentenek: egy 10 Mb/s (megabit/sec) teljesítményű Ethernet hálózat nagyjából 1100 KB/s (kilobájt/sec) sebességgel képes kezelni 1500 bájtos csomagokat. Egy 100 Mb/s teljesítményű Fast Ethernet hálózatban 6500 KB/s sebesség érhető el ugyanilyen csomagméret mellett. Az FDDI a maga 155 Mb/s teljesítményével megközelítőleg 6250 KB/s sebességet is lehetővé tesz, és jóval nagyobb csomagokat (4 KB) is kiválóan kezel.

Az ATM még ennél is jobb, de a könyv írásának idején még olyan újdonságnak számított, amelynek nem volt előre látható a tudása; várhatóan 7,125 Mb/s teljesítményre lesz képes 9 KB-os csomagokkal számolva.

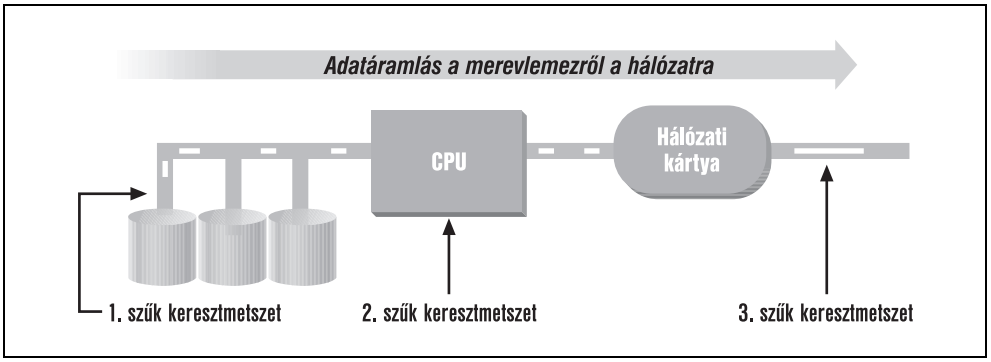
Természetesen egyéb módon is előállhatnak szűk kapacitások: nem jó, ha ugyanarra a lemezvezérlőre egynél több IDE lemez kapcsolódik, mint ahogy az sem jó, ha háromnál több 3600-as SCSI-I lemezmeghajtó van egy lassú/szűk vezérlőre kötve, vagy háromnál több 7200-as SCSI-II lemez kapcsolódik egy gyors/széles vezérlőre. A RAID 5 ugyancsak lassú, és kétszer annyi írást igényel, mind a független lemezek vagy a RAID 1.

Ha már túljutottunk a második Ethernet készleten és a második lemezvezérlőn, akkor foglalkozzunk a buszok sávszélességével, főként ha ISA/EISA buszok vannak a rendszerünkben.

A szűk keresztmetszetek csökkentése

A fenti adatok ismeretében már elkészíthető egy modell, amely alapján megállapíthatjuk egy adott gép maximális képességét. Az adatok többsége Brian Wong *„Configuration and Capacity Planning for Solaris Servers”* című könyvéből származik, ezért a példáinkban használt Sun géphez képest némi eltérések mutatkoznak.

A bemutatásra kerülő modell nem tökéletes. Ne gondoljuk azt, hogy ez a modell az összes szűk keresztmetszetre rámutat, de még azt sem, hogy a becslések akár a 10%-os hibahatáron belül maradnak. Részletesebb és pontosabb adatok csak ennél jóval bonyolultabb modellektől várhatók. (Valós modellekről Raj Jain *„The Art of Computer Systems Performance Analysis”* című könyvében olvashatunk. A fentebb és az itt említett könyv adatait a 9. fejezet végén, a *Szakkönyvek* alcim alatt olvashatják.) Ezek előrebocsátásával lássuk a rendszerünket (B.2. ábra).



B.2. ábra. Adatáramlás egy Samba kiszolgálón a lehetséges szűk keresztmetszetekkel

Olvasási műveletnél az adatok a merevlemezről elindulva áthaladnak a buszon, majd a CPU-n keresztül vagy azt megkerülve a hálózati kártyára jutnak. A kártya csomagokat készít belőlük, és elküldi a csomagokat a hálózatra. A példánkban végigkövetjük az adatok áramlását, és megnézzük, hol ütköznek szűk keresztmetszetekbe. Elég könnyen összeállíthatunk olyan táblázatokat, amelyekbe felvesszük a szokásos merevelemezek, CPU-k és hálózati kártyák maximális teljesítményét. A következőkben ezt tesszük.

Vegyünk egy konkrét példát! A rendszer alapja egy Linux Pentium 133 MHz-es gép, egyetlen 7200 fordulat/perc fordulatszámú adatlemezzel, PCI busszal és egy 10 Mb/s teljesítményű Ethernet kártyával. Egy ilyen paraméterekkel rendelkező gép már kiválóan használható kiszolgálóként. Elsőként a B.2. táblázatot töltöttük ki, amelybe a rendszerünk első számú potenciális szűk keresztmetszetének, a merevlemeznek az adatait vettük fel.

B.2. táblázat. A merevlemez teljesítménye

Merevlemez fordulatszám/perc	**I/O műveletek/s**	**KB/s**
7200	70	560
4800	60	480
3600	40	320

A merevlemez teljesítményét a másodpercenként átvitt adatok kilobájtban mért mennyisége jelenti. Az értékét abból számítottuk ki, hogy hány 8 KB-os I/O műveletet képes a lemez másodpercenként elvégezni, ami viszont erősen függ a lemez fordulatszámától és a bitsűrűségtől. A kérdés lényegében a következő: mennyi adat haladhat át az olvasófejek alatt egy másodperc alatt? A példabeli rendszerünkben az egyetlen, 7200 fordulatszámú merevlemezen másodpercenként 70 I/O műveletet mértünk, ami nagyjából 560 KB/s teljesítménynek felel meg.

A következő lehetséges szűk keresztmetszet a CPU lehet. A mai gépeken az adatoknak már nem kell áthaladniuk a CPU-n, ezért itt némileg közvetett módon számítottuk az adatátbocsátást.

A CPU feladata az I/O kérések kiadása és az erre érkező megszakítások kezelése, majd az adatok továbbítása a buszon a hálózati kártyához. A korábbi tapasztalataink alapján tudjuk, hogy a fájlrendszer kódjából adódóan eltekinthetünk attól, hogy más szoftverek is futnak a gépen. Az adatátbocsátást egyszerűen úgy számítottuk ki, hogy a CPU által másodpercenként elvégezhető fájl I/O műveletek (mért) számát megszoroztuk ugyanazzal a 8 KB-os átlagos adatmennyiséggel. Az eredményt felvettük a B.3. táblázatba.

B.3. táblázat. A CPU teljesítménye

CPU	**I/O műveletek/s**	**KB/s**
Intel Pentium 133	700	5600
Dual Pentium 133	1200	9600
Sun SPARC II	660	5280
Sun SPARC 10	750	6000
Sun Ultra 200	2650	21 200

Ezt követően összevetettük a lemez és a CPU teljesítményét. A Linux rendszerben egyetlen, 7200-as fordulatszámú merevlemez van, aminek 560 KB/s a teljesítménye, ugyanakkor a benne lévő CPU-ra a másodpercenkénti 700 I/O műveletet kaptuk, aminek viszont 5600 KB/s teljesítmény felel meg. Amint várható volt, ebben az összehasonlításban a merevlemez bizonyult a szűk keresztmetszetnek.

Utolsó lehetséges szűk keresztmetszetként a hálózatot vizsgáltuk. Ha ennek a sebessége 100 Mb/s alatt van, akkor ez lesz a fő akadály. Emellett természetesen a hálózati kártya is lassíthatja a forgalmat. A B.4. táblázatban néhány hálózat átlagos sebességét tüntettük fel. A hálózati sebességeket ugyan bit/s mértékegységben szokás megadni, mi itt KB/s egységet használtunk, hogy egyszerűbb legyen az előző táblázatok adataival való összehasonlítás.

B.4. táblázat. Hálózati sebességek

Hálózat típusa	**KB/s**
ISDN	16
T1	197
Ethernet 10m	1113
Token ring	1500
FDDI	6250
Ethernet 100m	6500
ATM 155	7125*

Az előző példánkban a szűk keresztmetszetet a lemez okozta a maga 560 KB/s teljesítményével. A B.4. táblázat azt mutatja, hogy egy normál 10 megabit/s teljesítményű Ethernet hálózat (1113 KB/s) messze gyorsabb, mint a lemez, vagyis továbbra is az utóbbi a fő akadály.

* Ez az érték nagyobb lesz. Az olyan rendszerek, mint a Cray, a Sun Ultra, és a DEC/Compaq Alpha már ennél jobb eredményeket is elérnek.

A bemutatott táblázatok alapján kijelenthető, hogy a kisebb kiszolgálóknak nem lesznek problémáik a CPU-val, és a nagyobb, többprocesszoros rendszerek is inkább a lemezösszefűzést és a többszörös Ethernetet fogják támogatni, mielőtt még a CPU teljesítményével lennének gondjaik.

Gyakorlati példák

Wong a *„Configuration and Capacity Planning for Solaris Servers"* című könyvében egy példában kimutatta, hogy egy kétprocesszoros SPARCstation 20/712 négy Ethernet kártyával és hat 2,1 GB-os merevlemezzel az ideje nagy részét azzal tölti, hogy a lemezektől visszaküldendő adatokra vár. Ha több merevlemez lenne a rendszerben (Brian Wang egyenesen 34 darabot javasol), akkor az Ethernet kártyák szorítanák 1200 KB/s alá a sebességet. Ha szeretnénk teljes mértékben kiaknázni egy ilyen gép teljesítményét, akkor további Ethernet kártyákat kellene felvenni, és 100 Mbps Fast Ethernet vagy 155 Mbps FDDI hálózatot kellene alkalmazni.

Az ilyen következtetések levonásához a B.5. táblázat nyújt segítséget.

B.5. táblázat. Középkategóriás kiszolgáló hangolása

Számítógép	**Lemez teljesítménye**	**CPU teljesítménye**	**Hálózat teljesítménye**	**Eredő teljesítmény**
Dual SPARC 10, 1 lemez	560	6000	1113	560
További 5 lemez	3360	6000	1113	1113
További 3 Ethernet kártya	3360	16 000	4452	3360
Váltás 20 lemezes tömbre	11 200	6000	4452	4452
100 Mbps dual Ethernet	11 200	6000	13 000	11 200

Kezdetben az egyetlen, mindössze 560 KB/s teljesítményű lemez okozta a szűk kapacitást. Ezen úgy javítottunk, hogy öt további lemezt vettünk fel a rendszerbe. Ekkor a lemezek teljesítménye már meghaladta az Ethernet hálózatét, vagyis az utóbbi vált akadállyá. Ezért aztán e két összetevő többféle kombinációját is kipróbáltuk. A lemezek, CPU-k és hálózati kártyák számának növelésekor különböző helyekre tevődött át a szűk kapacitás. A lényeg az, hogy újabb hardverelemek felvételével mindaddig próbálkozzunk a szűk kapacitások megszüntetésével, amíg nem érjük el a kívánt teljesítményt, vagy amíg (sajnos) fizikailag lehetséges, vagy el nem fogy a pénzünk.

E könyv szerzőinek tapasztalatai is megerősítik a fenti számításokat; egyikük egy nagy SPARC 10 fájlkiszolgálót üzemeltetett, amely egy Ethernet plus hálózaton két processzorral megközelítőleg elérte egy FDDI ring hálózat teljesítményének egyharmadát. Majdnem ugyanezt az eredményt kapta egyetlen processzorral is, bár ehhez gyorsabb és erősen túlpörgetett operációs rendszert használt.

Ugyanezek a megfontolások vonatkoznak a célirányosan megtervezett kiszolgálókra is. A szerzők azt tapasztalták, hogy azonos szabályok érvényesülnek a DECstation 2100, a legújabb Alpha vagy Compaq, a régi MIPS 3350 és az új SGI O2 gépekre. Általánosságban az mondható, hogy egy többprocesszoros kiszolgálónak elegendően nagy a busz-sávszélessége és CPU teljesítménye ahhoz, hogy a fájlszolgáltatás során a lemez I/O műveleteire szűkítse le a forgalmi akadályokat.

Hány ügyfél kiszolgálására képes a Samba?

Nos, ez teljes mértékben attól függ, hogy mennyi adatra van szüksége az egyes felhasználóknak. Egy kisebb, három SCSI-I merevlemezzel felszerelt kiszolgáló, amely 960 KB/s sebességgel tudja küldeni az adatokat, 36–80 olyan ügyfelet tud kiszolgálni, akik egy normál irodai környezetben nagyjából azonos méretű táblázatokat vagy szövegszerkesztőbeli dokumentumokat nyitnak meg és mentenek (36 ügyfél 2,3 átvitel/s 12 KB fájl = 1 MB/s).

Ugyanez a kiszolgáló egy fejlesztőintézetben, ahol programozók futtatnak nagy adatigényű szerkesztési és fordítási teszteket, már egyetlen felhasználó is igényelheti az 1 MB/s teljesítményt, ezért az ügyfelek száma esetleg a 25-öt sem érheti el. Még egy lépéssel továbbmenve: egy nyomdai levilágító rendszer, amelynek egyik ügyfele sem éri be 10 MB/s-nál kisebb teljesítménnyel, a kiszolgáló méretétől függetlenül gyengén fog működni, ha az ügyfelek egy 10 Mbps-os Ethernet hálózathoz kapcsolódnak.

Ha nem tudjuk, hogy egy átlagos felhasználónak mennyi adatra van szüksége, akkor a létező NFS, Netware vagy LAN Manager kiszolgálók mintájára méretezzük a Samba kiszolgálót. Ügyeljünk arra, hogy az új kiszolgálóban annyi merevlemez és lemezvezérlő legyen, mint amennyi a mintául szolgáló kiszolgálóban is van.

Ha tudjuk, hogy egy létező kiszolgáló hány ügyfelet tud kiszolgálni, akkor kedvezőbb a helyzetünk. Ekkor ugyanis elemezhetjük a kiszolgáló működését, megvizsgálhatjuk, hogy mekkora a maximális kapacitása, és megbecsülhetjük, mennyi adatot kell szolgáltatnia. Ha például egy két IDE lemezes PC kiszolgáló lassúnak bizonyul, ha 30 PC-s ügyfél home könyvtárát kell szolgáltatnia, de 25 ügyfél esetén már elfogadható a teljesítménye, akkor abból indulhatunk ki, hogy a szűk kapacitást az Ethernet I/O műveletei (megközelítőleg 375 KB/s), és nem a lemez I/O műveletei (640 KB/s) jelentik. Amennyiben így van, akkor arra következtethetünk, hogy az ügyfelek átlagos igénye 15 KB/s (375/25 = 15).

Egy 75 ügyfélből álló új hálózat elkészítéséhez 1125 KB/s teljesítményre van szükség, amelyet több (célszerűen három) Ethernet kártyára kell szétosztani, a kiszolgálót pedig legkevesebb három 7200 fordulat/perc fordulatszámú merevlemezzel és ennek megfelelő teljesítményű CPU-val kell felszerelni. Ezeknek a követelményeknek egy Pentium 133 MHz-es vagy ennél gyorsabb processzor már megfelel, ha a busz is támogatja ezt a sebességet (PCI).

Egy egyedileg megtervezett PC kiszolgáló vagy a többprocesszoros üzemmódra is képes olyan munkaállomások, mint a DEC/Compaq Alpha, egy SGI vagy ehhez hasonlók könnyebben méretezhetők, akárcsak egy Fast Ethernet hálózaton működő gép, kiegészítve egy elosztóval (hub), amelyen keresztül egyedi 10 MB/s-os Ethernet kártyákon keresztül lehet meghajtani az ügyfélgépeket.

Tippelési tanácsok

Ha semmilyen elképzelésünk nincs arról, hogy mire van szükségünk, akkor a legjobb, ha mások tapasztalatai alapján tippelünk. Az egyes ügyfélgépek átlagos igénye másodpercenként egy I/O műveletnél kevesebb is lehet (kereskedelemben, könyvelésben használt normál PC vagy Mac gépek), de a négyet is meghaladhatja a számuk (nagy alkalmazásokat futtató gyors munkaállomások). Egy fordítóprogramot futtató gyors munkaállomás átlagosan 3-4 MB/s adatátviteli sebességre tart igényt, de a grafikus alkalmazásoknak még ennél is többre van szükségük.

Mit javasolhatunk? Nézzünk meg valakinél egy hasonló konfigurációt, és próbáljuk meg felmérni a szűk kapacitásait. Javasoljuk továbbá Brian Wong *„Configuration and Capacity Planning for Solaris Servers"* című könyvének tanulmányozását. Bár a könyv példái többnyire Sun Solaris rendszerre vonatkoznak, amelyben a merevlemezek és a hálózati kártyák jelentik a szűk kapacitást, a legtöbb nagyobb szoftvergyártóra is érvényesek a megállapításai. Az FTP kiszolgálókra vonatkozó táblázatok adatai is közel állnak azokhoz az adatokhoz, amelyeket jelen könyv szerzői a Samba kiszolgálókon számítottak, és megfelelő kiindulási pontot jelentenek a becslésekhez.

Mérési űrlapok

A B.6. és a B.7. táblázat űrlapokat tartalmaz, amelybe feljegyezhetjük az adatainkat. A szűk kapacitásoknak az előző példában bemutatott kiszámítását táblázatkezelő programban, vagy kézzel a B.8. táblázat szerint végezhetjük el. Ha a Samba legalább olyan jó vagy jobb, mint az FTP kiszolgáló, és nem futtatunk a szokásostól nagyon eltérő teszteket, akkor megfelelően van konfigurálva a rendszerünk. Ha a loopback nem sokkal gyorsabb, mint bármi más, akkor a TCP/IP szoftverben lehet valamilyen hiba. Ha mind az FTP, mind a Samba lassú, akkor a hálózattal lehet probléma: hibás lehet az Ethernet kártya, de az is előfordulhat, hogy a kártya véletlenül félduplex üzemmódba van állítva, amikor nem ilyen elosztóhoz kapcsolódik. Ne feledjük, hogy míg a CPU-k és a merevlemezek sebességét bájtokban szokás megadni, addig a hálózat sebességét bitekben számítják.

A táblázatokban a bájtok és a bitek számára is van egy-egy oszlop. Az utolsó oszlopban a kapott eredményt a hagyományos Ethernet 10 MB/s sebességével hasonlítjuk össze százalékban kifejezve.

B.6. táblázat. Ethernet interfész ugyanahhoz a gazdához: FTP

Futtatás száma	**Méret bájtban**	**Idő (sec)**	**Bájt/s**	**Bit/s**	**10 Mb/s %-ában**
1					
2					
3					
4					
5					
Átlag:					
Eltérés:					

B.7. táblázat. Ethernet interfész ugyanahhoz a gazdához: FTP

Futtatás száma	Méret bájtban	Idő (sec)	Bájt/s	Bit/s	10 Mb/s %-ában
1					
2					
3					
4					
5					
Átlag:					
Eltérés:					

B.8. táblázat. Szűk kapacitásokat számító táblázat

CPU	CPU teljesítménye	Lemezek száma	Lemez teljesítménye	Hálózatok száma	Hálózat teljesítménye	Valós teljesítmény

A B.8. táblázatban:
- CPU teljesítménye = (KB/s a B.3. táblázatból) x (CPU-k száma);
- lemez teljesítménye = (KB/s a B.2. táblázatból) x (lemezek száma);
- hálózat teljesítménye = (KB/s a B.4.táblázatból) x (hálózatok száma);
- valós teljesítmény = min (lemez, CPU és hálózat teljesítménye).

Egy tipikus tesztre, ebben az esetben az FTP get parancsára a B.9. táblázatban látható értékeket írtuk volna be:

B.9. táblázat. Ethernet interfész ugyanahhoz a gazdához: FTP

Futtatás száma	Méret bájtban	Idő (sec)	Bájt/s	Bit/s	10 Mb/s %-ában
1	1812898	2,3	761580		
2		2,3	767820		
3		2,4	747420		
4		2,3	760020		
5		2,3	772700		
Átlag:		2,32	777310	6218480	62
Eltérés:		0,04			

A korábbi példában használt Sparc gépre a B.10. táblázatban látható értékeket kapnánk:

B.10. táblázat. Sparc 20-as példa, Redux

CPU	CPU teljesítménye	Lemezek száma	Lemez teljesítménye	Hálózatok száma	Hálózat teljesítménye	Valós teljesítmény
2	6000	1	560	1 10base2	1113	560
2	6000	6	3360	1	1113	1113
2	6000	6	3360	4 10base2	4452	3360
2	6000	20	11 200	4	4452	4452
2	6000	20	11 200	2 100base2	13 000	11 200