

## Könnyű álmok: a protokollok (11. rész)

Sorozatunk ezen részében a napjainkban használt internetes alkalmazások belső működéséről rántjuk le a leplet.



**A** fő célkitűzésünk, hogy ismertessük a legfontosabb protokollok működését, valamint hogy miként lehet őket csomagszűrővel elérhetővé tenni.

Mint arról már a korábbi cikkekben szó esett [1.], a Linux csomagszűrője (a 2.2-es és a 2.4-es rendszermagé is) három alapvető szűrőláncot tartalmaz. Az input láncra érkeznek be a hálózatról és a helyi csatolóról (lo) érkező csomagok, az átmenő forgalom a forward láncan halad keresztül, és a kimenő csomagok az output láncan keresztül hagyják el a hálózati alrendszert.

Ne felejtjük el, hogy míg a 2.2-es rendszermag csomagszűrőjénél egy átmenő csomag valóban minden láncan keresztülhaladt, addig az új 2.4-es rendszermag szűrőjénél (Netfilter, lásd Linuxvilág 2001. október, 27. oldal) az átmenő forgalom csak a forward láncan halad keresztül.

A cikk 2.2-es rendszermagra vonatkozó példáinál az egyszerűség kedvéért az átvivendő csomagok engedélyezését kizárólag az input láncan mutatjuk be. Ennek helyes működéséhez vagy a másik két láncan is engedélyezni kell e csomagok továbbítását, vagy biztosítanunk kell, hogy a másik két lánc ne akadályozza a forgalom zavartalanságát.

Írásunkban szereplő példákban feltételezzük, hogy tűzfalunknak csupán két lába van. Az eth0 csatoló a belső (védett), míg az eth1 csatoló a külső hálózat (Internet) felé mutat.

### Ping

A ping a legalapvetőbb programok egyike. Gyakran használják a távoli gépek működésének ellenőrzésére vagy a hálózat késleltetésének mérésére. Először nézzük meg a ping által létrehozott hálózati forgalmat!

Mint az 1. ábrán szereplő tcpdump kimenetből is látható, a *cheetah* nevű gép egy icmp echo request csomagot küld (az icmp csomagtípusa echo request, a kódértékek ehhez a típushoz nem tartoznak). A megcímezett gép (a példán a *dolphin* nevű) erre egy icmp echo reply csomaggal (az icmp csomagtípusa echo reply, kód értékek ehhez a típushoz sem tartoznak) válaszol. A ping program az echo request elküldése és az echo reply beérkezése közti időt méri, és a kettő különbségét írja a time mezőbe.

A ping parancs az icmp csomag adatrészeiben olyan adatokat helyez el, hogy a visszakapott választ azonosítani tudja (például kiszűrhesse a többszörös válaszokat, valamint felismerhesse a csomagvesztést). Amennyiben a csomagot az üzenetszört (broadcast) címre irányítjuk, a hálózaton elhelyezkedő gépek közül több is válaszolhat – erre láthatunk példát a 2. listán. Az üzenetszört címre érkező echo request csomagok elfogadása néha hasznos lehet, ugyanakkor veszélyeket is rejt magában. Előszórával használják DoS (Denial of Service)-támadásoknál. Gondoljunk csak bele, hogy a támadó beküld egy olyan csomagot a hálózatunkba, amelynek forráscímét egy ártatlan gép címére hamisította. Ebben az esetben megcímezett gépeink válaszolnak, és ahány gépünk csak van, mind echo reply csomagot fog küldeni

az „ártatlan” gépnek. Ezzel a támadó eléri, hogy egyetlen csomag költségével hálózatunkat erősítőként használva gépeink számával megegyező számú csomagot küldjön áldozatának. A támadás neve az angol szakirodalomban *smurf attack*. Az üzenetszört címre érkező csomagokra a Linux-rendszermag alapesetben válaszol, de ez le is tiltható. A leltáshoz az alábbi parancsot használhatjuk:

```
echo 1 >
↳ /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

Természetesen ezt minden indításkor meg kell tenni.

A feladatot bizzuk a `sysctl` vagy `systune` parancsokra, vagy a fenti sort tegyük be egy olyan állományba, amely minden indításkor végrehajtódik.

Visszatérve az ábrához: látható, hogy a *cheetah* gép echo request csomagjára mind a *dolphin*, mind a *hawk* gép válaszol. Érdeemes megfigyelni, hogy a *cheetah* gép magának is válaszol. Az ábra megmutatja, hogy először a gép saját válasza érkezik meg, és csak a másik kettőt minősíti másodpéldánynak.

A ping parancsral kapcsolatban fontos megjegyezni, hogy az echo request csomagot a távoli gép rendszermaga dolgozza fel, az echo reply csomagot is ő küldi. Amennyiben a távoli gép az echo request kérésre válaszol, akkor a távoli rendszer TCP-, illetve IP-alrendszerének ezen része működőképes. Ettől azonban a gép még teljesen használhatatlan is lehet. Ugyanígy ha a ping-re elmarad a válasz, ez még nem feltétlenül jelenti a távoli rendszer teljes leállítását. Oka lehet egy hálózati hiba vagy paranoiás tűzfal is.

Amennyiben a tűzfalon engedélyezni szeretnénk, hogy a belső gépek a külső gépeket ping-gel ellenőrizhessék, kifelé engedélyeznünk kell az echo request, befelé pedig az echo reply csomagokat. Ha Linux-rendszerünk a belső hálózaton saját címosztályt használ, használt rendszermagunk `CONFIG_IP_MASQUERADE_ICMP` beállítási lehetőségének engedélyezettnek kell lennie. A ping-et az alábbi parancsokkal engedélyezhetjük abban az esetben, ha nem használunk szűrést sem az átmenő, sem a kimenő forgalomnál:

- 2.2.x-es rendszermagsorozat esetén:
 

```
ipchains -A input -i eth0 -p icmp
↳ --icmp-type echo-request -j ACCEPT
ipchains -A input -i eth1 -p icmp
↳ --icmp-type echo-reply -j ACCEPT
```
- 2.4.x-es rendszermagsorozat esetén (Netfilter használata mellett):
 

```
iptables -A FORWARD -i eth0 -p icmp
↳ --icmp-type echo-request -j ACCEPT
iptables -A FORWARD -i eth1 -p icmp
↳ --icmp-type echo-reply -j ACCEPT
```

## Traceroute

A traceroute program rendkívül hasznos segédeszköz a forgalomirányítási hibák felderítésére. A program működésének megértéséhez célszerű felelevenítenünk a korábbi számokban leírtakat az IP-protokoll működéséről [2].

A program a hálózati út meghatározásához trükkösen összeállított UDP-csomagokat használ. A trükk a csomagban az, hogy az élettartam-számlálóját (time to live) 1 értékre állítja. Így a legelső forgalomirányító berendezés, amely veszi a csomagot, rögtön megállapítja, hogy a csomag élettartama már lejárt, amit a feladóval egy icmp time exceeded típusú üzenetben azonnal közöl is. A time exceeded üzenet tartalmazza a hibát kiváltó UDP-csomag elejét, ezáltal a küldő azonosítani tudja. A csomag feladója a legelső forgalomirányító berendezés, így annak címét egyszerűen visszakaptuk. A time exceeded csomag vétele után a program az előző lépéseket ismétli, csak folyamatosan növelt ttl, valamint a célkapuértékkel, amíg csak a célszámítógépet el nem éri. Ezt szemlélteti a 3. lista (26. CD Magazin/Konnyu).

Mint láttuk, a közbenső forgalomirányító berendezések általában a time exceeded üzenetet küldik vissza. A célgépig eljutó csomagnál más a helyzet. A traceroute alkalmazást úgy írták meg, hogy általában kihasználatlan UDP-kapukat

címezzen, hiszen ilyenkor icmp port unreachable üzenet jön vissza. Amennyiben a célrendszer nem érhető el, úgy icmp destination unreachable üzenet is visszaérkezhet. A művelet sikerét nagymértékben befolyásolhatják az útközben elhelyezett csomagszűrő berendezések. A traceroute engedélyezése a tűzfalon kissé összetettebb, azonban megoldható. Amennyiben engedélyezni szeretnénk, hogy a tűzfalunkig használhassák a traceroute-ot, a 33434..33523 udp tartományra érkező csomagokat utasítsuk el egy icmp port unreachable üzenet segítségével (ezt teszi az IP Chains REJECT művelete is). Ha azt szeretnénk, hogy belső gépeink traceroute segítségével vizsgálhassák a külső gépeket, tűzfalunkon a fenti tartományra engedélyezni kell a kimenő UDP-csomagokat, valamint a bejövő icmp time exceeded és a icmp dest unreachable csomagokat is. Érdemes megfigyelni, hogy kimenő csomagjaink forráskapuja 32 768 feletti. Ennek oka, hogy a unixos traceroute a pid-jének (process id – folyamatazonosító) legelső 15 bitjét adja a 32 768-hoz, és így alakul ki a tényleges forráskapu. Az ehhez szükséges tűzfalbeállítások az alábbiak:

- 2.2.x-es rendszermagsorozatnál:  

```
ipchains -A input -i eth0 -p udp
↳-s 0.0.0.0/0 32768:
↳-d 0.0.0.0/0 33434:33523 -j ACCEPT
ipchains -A input -i eth1 -p udp -d
↳0.0.0.0/0 33434:33523 -j REJECT
ipchains -A input -i eth1 -p icmp
↳--icmp-type destination-unreachable
↳-j ACCEPT
ipchains -A input -i eth1 -p icmp
↳--icmp-type time-exceeded -j ACCEPT
```
- 2.4.x-es rendszermagsorozatnál (Netfilter használata mellett):  

```
iptables -A FORWARD -i eth0 -p udp -s
↳0.0.0.0/0 --sport 32768: -d 0.0.0.0/0
↳--dport 33434:33523 -j ACCEPT
iptables -A FORWARD -i eth1 -p udp -d
↳0.0.0.0/0 --dport 33434:33523 -j REJECT
iptables -A FORWARD -i eth1 -p icmp
↳--icmp-type destination-unreachable
↳-j ACCEPT
iptables -A FORWARD -i eth1 -p icmp
↳--icmp-type time-exceeded -j ACCEPT
```

### 1.a lista Ping parancs kimenete

```
[bozo@cheetah bozo]$ ping -c 3 dolphin
Warning: no SO_TIMESTAMP support, falling back to SIOCGSTAMP
PING dolphin.home (10.1.2.251) from 10.1.2.232 : 56(84) bytes of data.
64 bytes from dolphin.home (10.1.2.251): icmp_seq=0 ttl=255 time=1.083 msec
64 bytes from dolphin.home (10.1.2.251): icmp_seq=1 ttl=255 time=930 usec
64 bytes from dolphin.home (10.1.2.251): icmp_seq=2 ttl=255 time=916 usec

--- dolphin.home ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.916/0.976/1.083/0.079 ms
```

### 1.b lista Ping a vonalon

```
16:21:57.663751 < cheetah > dolphin: icmp: echo request
16:21:57.663910 > dolphin > cheetah: icmp: echo reply
16:21:58.655136 < cheetah > dolphin: icmp: echo request
16:21:58.655305 > dolphin > cheetah: icmp: echo reply
16:21:59.655105 < cheetah > dolphin: icmp: echo request
16:21:59.655272 > dolphin > cheetah: icmp: echo reply
```

## Domain

A domain protokoll az Internet egyik legalapvetőbb szolgáltatása, nélküle nagyon nehéz lenne a kapcsolattartás. A protokoll célja, hogy lehetővé tegye az „emberi fogyasztásra alkalmas” tartománynevek leképezését IP-címekre, valamint az IP-címek visszaalakítását tartománynevekké. Hathatós támogatást nyújt az elektronikus levelezéshez is. Maga a protokoll meglehetősen összetett, ezért csak a fontosabb részeit ismertetjük a teljesség igénye nélkül.

A DNS (Domain Name System) információinak alapegysége az úgynevezett RR (Resource Record). Egy RR tartalma lehet például egy név-IP-cím összerendelés. Az RR túl kis egység ahhoz, hogy hatékonyan kezelni lehessen, ezért az RR-eket zónákba szervezték. A zóna az RR-ek hierarchikus

## 2.a lista Üzenetszórt ping, duplikátumok jelölése

```
[bozo@cheetah bozo]$ sudo ping -b -c 3 10.1.2.255
WARNING: pinging broadcast address
Warning: no SO_TIMESTAMP support, falling back to SIOCGSTAMP
PING 10.1.2.255 (10.1.2.255) from 10.1.2.232 : 56(84) bytes of data.
64 bytes from 10.1.2.232: icmp_seq=0 ttl=255 time=272 usec
64 bytes from 10.1.2.251: icmp_seq=0 ttl=255 time=1.016 msec (DUP!)
64 bytes from 10.1.2.250: icmp_seq=0 ttl=255 time=1.373 msec (DUP!)
64 bytes from 10.1.2.232: icmp_seq=1 ttl=255 time=200 usec
64 bytes from 10.1.2.251: icmp_seq=1 ttl=255 time=952 usec (DUP!)
64 bytes from 10.1.2.250: icmp_seq=1 ttl=255 time=1.068 msec (DUP!)
64 bytes from 10.1.2.232: icmp_seq=2 ttl=255 time=179 usec

--- 10.1.2.255 ping statistics ---
3 packets transmitted, 3 packets received, +4 duplicates, 0% packet loss
round-trip min/avg/max/mdev = 0.179/0.722/1.373/0.456 ms
```

## 2.b lista Üzenetszórt ping a vonalon

```
16:23:56.581375 B cheetah > 10.1.2.255: icmp: echo request (DF)
16:23:56.581551 > dolphin > cheetah: icmp: echo reply
16:23:56.582350 P hawk > cheetah: icmp: echo reply
16:23:57.573525 B cheetah > 10.1.2.255: icmp: echo request (DF)
16:23:57.573694 > dolphin > cheetah: icmp: echo reply
16:23:57.574256 P hawk > cheetah: icmp: echo reply
16:23:58.573493 B cheetah > 10.1.2.255: icmp: echo request (DF)
16:23:58.573661 > dolphin > cheetah: icmp: echo reply
16:23:58.574277 P hawk > cheetah: icmp: echo reply
```

halmaza egészen a fa leveleig vagy egy másik zóna határáig. A zónák az Interneten elosztva és redundánsan kerülnek tárolásra. Minden zónához létezik egy kitüntetett kiszolgáló, ahol a zóna mesterpéldánya található. Ez a kiszolgáló a zóna elsődleges névkiszolgálója. A redundancia miatt fontos, hogy a zónára tartalék kiszolgálók is jussanak, hiszen a zónára vonatkozó adatok akkor is szükségesek, ha az elsődleges névkiszolgáló valami miatt nem érhető el (vonalhiba, túlterhelés vagy működésképtelen névkiszolgáló következtében). Minden zónához legalább egy tartalék névkiszolgálót kötelező bejegyezni. Ezeket hívjuk az adott zóna másodlagos névkiszolgálójának. Természetesen egy névkiszolgáló több zónának is az elsődleges forrása lehet, ugyanakkor más zónák másodlagos névkiszolgálójaként is működhet. A másodlagos kiszolgáló a zónában meghatározott időnként ellenőrzi az elsődleges kiszolgálót, és ha a zóna megváltozott, azt letölti onnan. A teljes zóna letöltését zónatranszfernek nevezik.

Az adott zóna elsődleges kiszolgálója és másodlagos kiszolgálói (amennyiben a nála levő másolat elég friss) teljes bizonyossággal válaszolhatnak a zónát érintő kérdésre. Ezt úgy nevezzük, hogy az adott zóna *authoritatív*. Mit is jelent ez a fogalom és miért is kell ilyenekkel foglalkoznunk? Minden DNS-kiszolgáló (az Internet hatékony működése érdekében) a lekért adatokat bizonyos ideig tárolja, így a kérdezőknek tovább is tudja küldeni.

Most tekintsük át, hogy miként is működik mindez a gyakorlatban! Valaki például meg szeretné nézni a [www.kiskapu.hu](http://www.kiskapu.hu) weblapot, de nem tudja a címét. Minden helyesen beállított számítógépnek rendelkezésére áll egy vagy több DNS-kiszolgáló címe, amit valamilyen módon megadtak neki. Ezek a

névkiszolgálók általában a felhasználó hálózatán helyezkednek el vagy az internetszolgáltatójának névkiszolgálóiról van szó. A lényeg, hogy a felhasználó számítógépéhez közel helyezkednek el. Amikor a felhasználó számítógépe nevet vagy IP-címet szeretne feloldani, a listán szereplő legelső kiszolgálóhoz fordul. Amennyiben elérhetetlen, a soron következővel próbálkozik (természetesen a folyamat kissé bonyolultabb, de erre most nem térünk ki). Az egyszerűség kedvéért tekintsük úgy, hogy csak egy ilyen kiszolgáló létezik. A felhasználó számítógépe kérést küld a beállított névkiszolgálónak, melyben a [www.kiskapu.hu](http://www.kiskapu.hu) névhez tartozó IP-címe(ke)t szeretné megtudni. Amennyiben

a névkiszolgáló ismeri a kérdésre a választ (például az adat rendelkezésre áll a gyorsárában (cache)), visszaküldi az ügyfélnek, és a folyamat lezárul. Amennyiben a névkiszolgáló gyorsára teljesen üres, nem tehetünk mást, minthogy „elindulunk a kályhától”. Ebben az esetben a kiszolgáló visszaküldi az üzenetet, miszerint a válasz nem található, és megadja az Internet root DNS-kiszolgálóinak a címét. Ezekután az ügyfél tőlük is megkérdezi, amire kíváncsi. Nem valószínű, hogy a root DNS-kiszolgálók személyesen ismernék a [www.kiskapu.hu](http://www.kiskapu.hu) címét, így szintén a „nincs találat” választ küldik vissza – a [.hu](http://www.kiskapu.hu) névkiszolgálóinak listájával kiegészítve. A válasz vétele után a felhasználó számítógépe a [.hu](http://www.kiskapu.hu) névkiszolgálóihoz fordul, ahol szintén elutasítják, azonban a válaszban már a [kiskapu.hu](http://www.kiskapu.hu) névkiszolgálóinak címét kapja meg. Az utolsó körben a felhasználó számítógépe a [kiskapu.hu](http://www.kiskapu.hu) névkiszolgálóihoz fordul, ahol megkapja a hőn áhított adatot. Amennyiben a kívánt név nem létezik (például elgépeztük, és [www.kiskapu.hu](http://www.kiskapu.hu)-t írtunk), a [kiskapu.hu](http://www.kiskapu.hu) zóna névkiszolgálója szintén elutasít minket, de azt is közli, hogy ez az adat autoritatív (azaz a gép tényleg nem létezik). A fenti folyamat kissé vadregényesnek tűnik, és a hatékonyság csökkenni szikrája sem lelhető fel benne. Ezen a gondon segít, hogy a névkiszolgálók a kapott adatokat bizonyos ideig megőrzik és felhasználják. Könnyen lehet, hogy szolgáltatónk DNS-kiszolgálója tisztában van a [www.kiskapu.hu](http://www.kiskapu.hu) címmel vagy legalább a [kiskapu.hu](http://www.kiskapu.hu) zónával, ezért a folyamat jóval rövidebb. A protokoll tervezői azonban egyszerűbb megvalósításokra is gondoltak, ezért az úgynevezett rekurzív lekérdezést is megvalósították. Ebben az esetben névkiszolgálónk a fenti folyamatot maga végzi el, a felhasználó számítógépe már csak a végső választ kapja meg. (Ez jelen esetben nem

„42” – az indokért lásd *Douglas Adams* „Galaxis útikalauz stopposoknak” c. regénysorozatát). Az IP-cím és tartomány-név összerendelését egy ügyes trükk segítségével oldották meg: az IP-címeket is beforgatták a rendes namespace-be. Az IP-cím bájttjait fordított sorrendben írjuk le, majd az `in-addr.arpa` tartományt tesszük mögé. Például a `192.168.1.2` cím keresésekor a `2.1.168.192.in-addr.arpa` névre keresünk.

A domain protokoll UDP- és TCP-kaput is használhat egyszerre. Az UDP a leggyakrabban használt protokoll, hiszen ebben az esetben a kapcsolat felépítése és lebontása elmarad.

Az UDP-válaszcsomagban azonban a nagyobb válaszok nem férnek el, ezért ebben az esetben TCP-t kell használnunk.

A zónatranszfer kötelező módon TCP-t használ. Nagyon fontos, hogy mindkét elérést lehetővé tegyünk, ellenkező esetben majd megfoghatatlannak tűnő hibáknak lehetünk a tanúi.

A vonali forgalom elemzésekor megfigyelhető, hogy az egyszerű felhasználói gépek a DNS-kiszolgálóktól eltérően működnek. Míg az UDP-kérések az egyszerű gépek esetén véletlenszerű (1023 feletti) forráskapuról érkeznek, addig a DNS-kiszolgálók általában rögzített forráskaput használnak.

Ez pedig jelentősen leegyszerűsíti a csomagszűrő szabályok létrehozását. A TCP protokoll használata esetén ez a különbség nem létezik: itt mind az egyszerű ügyfél, mind a DNS-kiszolgáló a kérdést 1023 feletti (úgynevezett non privileged) kapuról kezdeményezi. Egy DNS-címkerést és egy zónatranszfert láthatunk a 4. listán (26. CD Magazin/Konnyu).

Amennyiben valamely gépünkön DNS-kiszolgálót futtatunk, ne feledkezzünk meg az UDP-kapú rögzítéséről! Amennyiben a domain-kérések rögzített UDP-kapuról mennek ki, csomagszűrőnkkel sokkal „szorosabbra húzhatjuk”.

## NTP

A kiszolgálók és az ügyfélgépek óráinak összehangolása nagyon hasznos dolog. Enélkül a naplóállományok ellenőrzése sokkal bonyolultabb, különösen, ha több rendszerről van szó. Bár utóbbi eset pusztán kellemetlenséggel jár (elvileg ismerhetjük a gépünk órája és a valós idő közötti különbséget, amit helyesbíthetünk is), számos esetben a gépek óráinak összehangolása elkerülhetetlen. Ilyen lehet valamely osztott állományrendszer használata vagy az időfüggő azonosítás (például *Kerberos* vagy *SecurID*). Most nézzük át, miként is valósítható meg ez a gyakorlatban!

Az egyik legelterjedtebb módszer, hogy a rendszergazda időnként ránéz a karórájára, majd a gép rendszeridejét átállítja.

A megoldás számos hátránnyal jár: egyrészt a beállítás nem lesz túl pontos, másrészt az egész művelet kissé esetleges.

Az ilyen műveletet a rendszergazdák csak szóróványosan szokták elvégezni, emiatt előfordulhat, hogy két állítás között valamikor egy hét, de megeshet, hogy több hónap telik el. Szerencsésebb lehet, ha kapcsoltvonalas szolgáltatás esetén kitarcsázás után, állandó kapcsolat esetén meghatározott időnként egy parancsot indítunk el, amely egy megadott viszonyítási alaphoz hangolja gépünk óráját. Ilyen feladatot lát el például az `rdate` parancs. Az eddigiek folyamán még nem szoltunk a módszerek másik hátrányos tulajdonságáról, hogy az idő elkezd „furcsán viselkedni”. Gondoljunk csak bele, mi történne, ha hirtelen visszaugranánk a múltba! Márpedig ha a rendszerünk órája siet, akkor pontosan ez fog történni, ami a működésében számos hibát okozhat. (Gondoljunk át a `make` parancs működését: a programok fordítását úgy vezérli, hogy a lefordított állomány módosítási idejét összeveti a forrásállomány módosítási idejével. Amennyiben hátraugrunk az

időben, a frissen módosított forrás régebbi lehet, mint az előzőleg keletkezett bináris változat!) Ezeket javítandó fejlesztették ki az NTP-t, azaz a Network Time Protocollt.

Az összehangolás két részben valósul meg. Az első rész (választhatóan) a rendszer elindulásakor történik meg, amikor is a rendszer órája rááll a viszonyítási időre (itt ugyanolyan ugrásról van szó, mint a fenti esetben). Ennek célja, hogy a rendszeridő megközelítőleg azonos legyen a viszonyítási idővel. Ez legegyszerűbben az `nptdate` parancs használatával valósítható meg. Ezekután következik a tényleges összehangolási folyamat, valamint az összhangban tartás. Ezt egy egyedi démon valósítja meg, ami az `ntpd` névre hallgat (a régebbi változatát `xntpd`-nek nevezték). Ez a démon folyamatosan méri a rendszeridőnek a viszonyítási időtől való távolságát, valamint a két rendszer közötti késleltetést (általában hálózati viszonyítási forrásokat használunk). Ezek figyelembevételével a gép rendszeróráját sietteti vagy késlelteti. Az idő múlási sebességének állításával elérhető, hogy a két idő közelítsen egymáshoz, ugyanakkor az idő monoton növekedése megmarad, ami a naplózó alrendszer szempontjából nagyon fontos.

Az NTP-protokoll UDP-t használ. Az NTP-kiszolgáló a számítógép 123-as UDP-kapuján vár csomagokra. Az ügyfelek esetén előfordulhat, hogy tetszőleges kapuról (nem csak 1024 alattiakról) indítják a kéréseiket. Amikor két NTP-kiszolgáló kapcsolatba lép egymással, mind a forrás-, mind a célkapu a 123-as. Az NTP-kiszolgálók a hatékonyabb kapcsolattartás érdekében képesek csoportcímezést alkalmazni, ilyenkor a csomag cílcíme `224.0.1.1` lesz. Az NTP vonali működését az 5. lista (26. CD Magazin/Konnyu) szemlélteti.

Fontos azonban, hogy tisztában legyünk az önműködő óra-összehangolás veszélyeivel is. Egy támadó átveheti gépünk órája felett a hatalmat, vagy kihasználhatja az óra-összehangolással kapcsolatos programok valamelyikében található biztonsági réseket. Mindkét eset súlyos következményekkel járhat. Ezeket mindig mérlegeljük egy új szolgáltatás használata során. Megoldás lehet, hogy saját időalapot állítunk üzembe, vagy az időt a hálózatról vesszük ugyan, de eltérő megvalósításokat használunk (például az időt közvetlenül egy forgalomirányító berendezés veszi, és mi azzal hangoljuk össze a kiszolgálónkat).

## Hivatkozásjegyzék

- [1.] Linuxvilág: 2001. február-március 54. oldal – Könnyű álmok (4. rész)
- [2.] Linuxvilág: IP-alapok 2001. január 44. oldal – Könnyű álmok 3. rész



*Borbély Zoltán* (bozo@andrews.hu), okleveles mérnök-informatikus. Főként Linuxon futó számítógépes biztonsági rendszerek tervezésével és fejlesztésével foglalkozik. A 1.0.9-es rendszermag ideje óta linuxozik. Szabadidejét barátaival tölti.



*Mátó Péter* (atya@andrews.hu), informatikus mérnök és tanár. Biztonsági rendszerek ellenőrzésével és telepítésével, valamint oktatással foglalkozik. 1995-ben találkozott először linuxos rendszerrel. Ha teheti, kirándul vagy olvas.