



SZÉCHENYI ISTVÁN EGYETEM
MŰSZAKI TUDOMÁNYI KAR
Távközlési Tanszék



TMDK Dolgozat

A DNS64 és NAT64 IPv6 áttérési technikák egyes implementációinak teljesítőkéesség- és stabilitás-vizsgálata

Répás Sándor

**Villamosmérnöki MSc szak
Távközlési rendszerek és szolgáltatások
2. évfolyam**

Témavezető: Dr. Lencse Gábor egyetemi docens

Győr, 2012

1. Tartalom

1.	Tartalom.....	1
2.	Bevezetés	4
3.	IPv6 bevezetési technikák.....	6
3.1.	Fontosabb áttérési technikák	6
3.1.1.	DNS64/NAT64	6
3.1.2.	Dual Stack.....	8
3.1.3.	Dual Stack Lite.....	8
3.1.4.	6to4.....	9
3.1.5.	6rd (IPv6 rapid deployment).....	12
3.1.6.	Teredo	12
3.1.7.	ISATAP (Intra-Site Automatic Tunnel Addressing Protocol).....	12
3.1.8.	SIIT (Stateless IP/ICMP Translation).....	14
3.1.9.	NAT-PT/NAPT-PT	14
3.1.10.	Application Level Gateway/Proxy	14
3.1.11.	PortProxy.....	14
4.	DNS64 és NAT64 implementációk	15
4.1.	Korábbi vizsgálatok.....	15
4.2.	Implementációk kiválasztása.....	15
5.	DNS64 mérés.....	18
5.1.	A mérési összeállítás	18
5.1.1.	Hálózat felépítése.....	18
5.1.2.	Számítógép konfigurációk	18
5.2.	Mérési módszerek.....	20
5.2.1.	IPv4 DNS szerver.....	20
5.2.2.	DNS64 szerver	21
5.3.	Mérési eredmények	25

5.3.1.	BIND rezolverként Linuxon futtatva	25
5.3.2.	BIND rezolverként OpenBSD-n futtatva	27
5.3.3.	BIND rezolverként FreeBSD-n futtatva	28
5.3.4.	BIND forwarderként Linuxon futtatva	29
5.3.5.	BIND forwarderként OpenBSD-n futtatva	29
5.3.6.	BIND forwarderként FreeBSD-n futtatva	30
5.3.7.	TOTD forwarderként Linuxon futtatva	30
5.3.8.	TOTD forwarderként OpenBSD-n futtatva	31
5.3.9.	TOTD forwarderként FreeBSD-n futtatva	31
5.4.	Értékelés	32
6.	NAT64 mérés	33
6.1.	A mérési összeállítás	33
6.2.	Mérési módszerek	33
6.2.1.	Linux alapú TAYGA	34
6.2.2.	OpenBSD Packet Filter	35
6.2.3.	Responder számítógép	36
6.2.4.	Kliens számítógépek	36
6.2.5.	NAT64 mérés	36
6.3.	Mérési eredmények	37
6.3.1.	Linux alapú TAYGA	37
6.3.2.	OpenBSD Packet Filter	38
6.4.	Értékelés	39
7.	Összefoglalás	41
8.	Irodalomjegyzék	42
9.	Függelék	46
9.1.	IPv6 hálózat kialakítása Linux NAT44 és 6to4 router segítségével	46
9.2.	IPv6 hálózat kialakítása Windows Server 2008 R2 NAT és 6to4 router segítségével	48
9.3.	6to4 hálózat kialakítása Linux 6to4 relay alkalmazásával	50

9.4.	6to4 hálózat kialakítása Linux 6to4 Windows 7 klienssel	54
9.5.	ISATAP hálózat létrehozása Windows Server 2008 R2 ISATAP routerrel	55
9.6.	ISATAP hálózat létrehozása Linux ISATAP routerrel	60
9.7.	Linux ISATAP kliens beállítása.....	61

2. Bevezetés

Az Internet terjedésének egyre gyorsuló tempója miatt a manapság általánosan használt IPv4 címzésnél rendelkezésre álló címtartomány nem teszi lehetővé, hogy minden az Internetre csatlakozó eszköz saját egyedi, publikus IP címmel rendelkezzen. Az IPv4 esetében a 4 bájtból álló cím elméletileg 2^{32} egyedi címet tenne lehetővé, ez azonban jelentősen csökken a speciális funkciók részére fenntartott címekkel (pl. privát és multicast címek), valamint az Internet korai szakaszában kiosztott „hatalmas” méretű címtartományokkal (PL: A osztályú tartományok). Ezen felül a hálózati útválasztók kielégítő sebességű működéséhez szükség van a címek minél inkább összefüggő tartományokban történő kiosztására is, ami szintén jelentősen csökkenti a ténylegesen rendelkezésre álló címek mennyiségét.

A még szabad nyilvános IPv4 címek darabszáma évek óta folyamatosan csökken, annak ellenére, hogy az évek során egyre több módszert dolgoztak ki ennek lassítására (pl: NAT, PA Assignment), és egyre szigorodó feltételekkel lehet csak címekhez jutni.

Az Internet népszerűségét látva már az 1990-es évektől látszott, hogy a kialakított címzési rendszer hosszú távon nem lesz működőképes. Az IPv4-es címzési módszert leváltani hívatott IPv6-os szabványt az RFC 2460 már 1998-ban rögzítette, de bevezetése az utóbbi évekig nem került előtérbe. [1] Az új publikus IPv4 címek kiosztásánál alkalmazott legnagyobb gondosság ellenére mára elérkeztünk oda, hogy várhatóan már 2012-ben sem tudnak új címeket kiosztani a címgazdálkodást végző szervezetek (pl: APNIC, RIPE). Az IPv6 esetében egy számítógépet vagy egyéb az Internetre csatlakozó eszközt már egy 128 bites cím azonosít. Azaz leegyszerűsítve 2^{128} -on eszköz lehet egy időben az Internetre csatlakoztatva. Így az IPv6-ra történő átállással a címek szűkösségéből adódó probléma várhatóan hosszú időre megoldódik. Az új protokoll kifejlesztésekor figyelembe vették az addig az IPv4 használata során felhalmozott tapasztalatokat is, és a címtartomány bővítésén felül más funkciók is a szabványba kerültek (pl: IPSec). Az új protokoll alkalmazására történő átállás azonban rendkívül költséges, mind eszköz oldalról, mind pedig a szükséges mérnökökárak mennyiségét tekintve. Így a legtöbb vállalkozás az átállást/bevezetést minél későbbi időpontra próbálja kitolni. Az IPv4-ről rendelkezésre álló információk szerint a bevezetés azonban nem halasztható tovább [2]. A bevezetés legfontosabb résztvevői az Internet szolgáltatók. Sajnos még most sem érhető el minden szolgáltatónál IPv6-os szolgáltatás. Ehhez a bevezetéshez keresek dolgozatomban olyan megoldásokat, melyek az áttérés jelenlegi szakaszában jól használhatóak, bevezetésük viszonylag egyszerű, és alacsony költségvonzattal járnak az Internetszolgáltatók számára.

A munka során végigtekintem az összes jelentősebb áttérési technikát (transition techniques), tanulmányozom a kapcsolódó szakirodalmakat, virtualizált környezetben kipróbálom azokat. Végül pedig a kiválasztott DNS64/NAT64 technika megbízhatóságát és teljesítményét laborkörülmények között is megvizsgálom. A vizsgálatok eredményei reményeim szerint sokat segítenek a hálózati szakemberek számára a megfelelő megoldás kiválasztásában és bevezetésében. Így munkám is hozzájárul a következő generációs hálózati protokoll, az IPv6 elterjedésének felgyorsításához.

3. IPv6 bevezetési technikák

Az IPv6 bevezetése során a két protokoll (IPv4 és IPv6) hosszú távú egymás mellett élésére lesz szükség. Ennek az együttélésnek a jelenleg is tartó, első fázisában a viszonylag alacsony számú szerver részére még jut elegendő publikus IPv4-es cím, azonban a folyamatosan gyorsuló ütemben gyarapodó kliensek számára már nem biztosítható a megfelelő számú egyedi IPv4-es cím. Az IPv6 terjedésével a szerverek egyre nagyobb hányada fog rendelkezni mindkét cím típussal, azonban néhány szervert annak üzemeltetője soha nem fog ellátni IPv6-os címmel, és néhány éven belül olyan új szerverek beüzemelése is tömegessé válik, melyek már nem rendelkeznek IPv4-es címmel.

Az Internet szolgáltatók nagy része még rendelkezik kiosztható IPv4-es címtartományokkal, de újabbakat az öt regionális Internet regisztrátor szervezettől (RIPE, AFRINIC, APNIC, ARIN, LACNIC) már nem tudnak igényelni [3]. Így érdemes a még fel nem használt IPv4-es címtartományokkal takarékoskodva biztosítaniuk a jövőben üzembe helyezni kívánt szervereik részére a megfelelő számú címet.

Sajnos sok szolgáltató az IPv6 protokoll bevezetésével kapcsolatosan még a felkészülést sem kezdte meg. Így olyan technikák kifejlesztése és alkalmazása is szükséges, amik a szolgáltató közreműködése nélkül is (csak IPv4-et támogató szolgáltatói hálózat esetén) lehetővé teszik a számítógépek számára mindkét protokoll feletti kommunikációt.

3.1. Fontosabb áttérési technikák

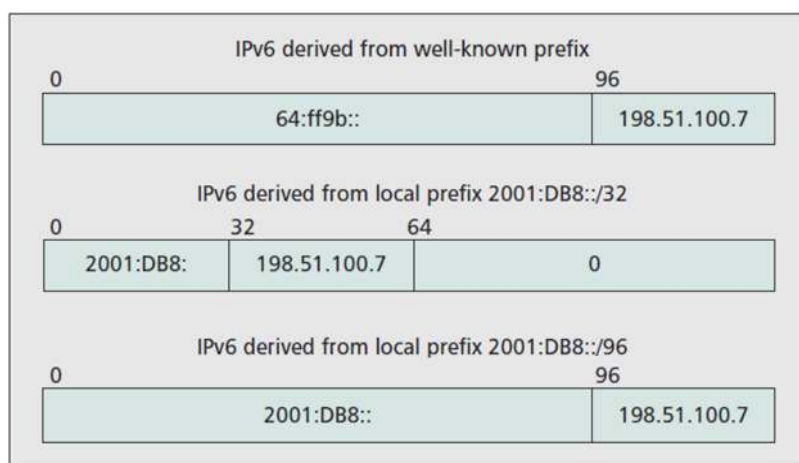
3.1.1. DNS64/NAT64

A DNS64 [4] és a NAT64 [5] segítségével egy csak IPv6-os címmel rendelkező kliens igénybe tudja venni egy csak IPv4-es címmel rendelkező szerver szolgáltatásait. A DNS64 a DNS szolgáltatás kiterjesztése, így várható, hogy rövid időn belül minden jelentősebb DNS szerver támogatni fogja a DNS64-et is.

DNS64

A csak IPv6-ot támogató klienseken DNS szerverként a DNS64 szervert kell beállítani. Ha egy kliens egy szerver szolgáltatását kívánja igénybe venni, először annak nevéhez tartozó IPv6-os címet kell megtudnia, ezért a DNS64 szerverhez fordul névfeloldási kérésével. A DNS64 szerver a normál DNS szerverekkel megegyező módon elvégzi a kért név feloldását, a rekurziót. A névfeloldás eredményétől függően két eset lehetséges:

1. Ha a rekurzió IPv6-os címet tartalmazó AAAA resource rekordot is adott vissza, akkor ezt az AAAA által meghatározott címet adja vissza válaszként a kliens részére. (Ha a válaszban A rekord is szerepelt, úgy azt is változtatás nélkül továbbítja a kliens felé.) Ez esetben normál DNS működés valósul meg DNS64 nélkül.
2. Ha a rekurzió eredménye csak A rekordot tartalmazott, úgy a DNS64 szerver a visszakapott A rekord, valamint a szerver beállításai alapján előállítja a megfelelő AAAA rekordot, melyet a rezolúciót kérő kliensnek az A rekorddal együtt a kliensnek válaszként visszaad. Így ugyanazon hosthoz tartozik egy A és egy speciális AAAA rekord is. Ebben a speciális AAAA rekordban megadott IPv6-os cím egy úgynevezett IPv4-Embedded IPv6 cím. Ennek az IPv6-os címnek az utolsó 32 bitje tartalmazza az eredeti IPv4-es címet, az első 96 bitje pedig vagy a NAT64 számára fenntartott 64:ff9b::/96 (well-known), vagy pedig egy a kliens hálózatában erre a célra kijelölt prefix. [6] Az 1. ábra mindkét esetet szemlélteti.



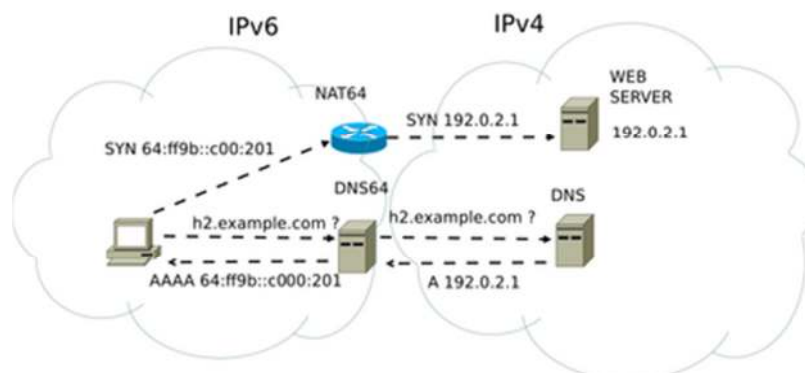
1. ábra A 198.51.100.7 IPv4-es cím IPv6-os megfelelője [7]

A csak IPv6-ot támogató kliens a DNS szervertől válaszként kapott IPv6-os címen keresztül kommunikál a szerverrel.

NAT64

Ha a szerver is rendelkezik IPv6-os címmel, a kommunikáció a kliens számára megegyezik a DNS64/NAT64 nélküli kommunikációval. Azonban, ha a szerver csak IPv4-es címen elérhető, úgy szükség van a NAT64 szolgáltatásaira is. Ilyenkor biztosítani kell, hogy a hálózatban a DNS64/NAT64-hez alkalmazott prefix felé irányuló forgalom a NAT64 gateway felé legyen irányítva. Legegyszerűbb esetben a NAT64 szolgáltatás a default gatewayen fut, de alkalmazható külön eszköz is, ahova akár statikusan, akár valamilyen IGP routing protokollal segítve juttatható el a forgalom. Egyetlen kikötés, hogy ha a kliens csak IPv4-et támogató szervert szeretne elérni (Az IPv4-Embedded IPv6 cím segítségével.), a forgalomnak át kell haladnia a NAT64 gatewayen. A NAT64 gateway rendelkezik egy globális IPv6 és egy publikus IPv4-es címmel is.

A csak IPv6-ot támogató kliens a csak IPv4-es szerver felé felépíti a kapcsolatot az IPv4-Embedded IPv6 cím felé. A kliens elküldi az IPv6 csomagot a kapott címre. A NAT64 gateway a kapott csomag alapján létrehoz egy IPv4-es csomagot az IPv6 célcím utolsó 32 bitjét felhasználva célcímként (Hiszen ez a 32 bit tartalmazza a célhost IPv4 címét.). A csomag összeállításához a gateway felhasználja a kapott csomag tartalmát (payload) és néhány a fejlécben szereplő mezőt. Forráscímként a saját IPv4 címét adja meg, így biztosítva, hogy a válasz csomagok megérkezzenek hozzá. Ezután az összeállított csomagot a NAT64 gateway elküldi a célállomásnak (szervernek). A csak IPv4-et támogató szerver ugyanúgy kiszolgálja a kérést, mint bármely más esetben, kliensként a NAT64 gateway forrás címét érzékelve. Így a szerver által küldött válasz csomag célcíme a NAT64 gateway IPv4-es címe lesz. A NAT64 gateway az IPv4-es válaszcsoport megérkezését követően annak tartalmát átcsoportolja IPv6-os csomagba, felhasználva a kapott IPv4-es fejléc néhány mezőjét, majd a csomagot továbbküldi a kérést kiküldő csak IPv6-ot támogató kliensnek. A NAT64 gatewayen szükség van egy nyilvántartás vezetésére, amiből kikereshető, hogy az egyes válaszcsoportok, mely kérésekhez érkeztek, így a válasz IPv6 csomag összeállítható. A NAT64 gatewayen lezajló folyamat nagyon hasonlít az IPv4 stateful nat-ra. [7]



2. ábra DNS64/NAT64 működése [8]

3.1.2. Dual Stack

A kliens globális IPv6 [9] és publikus IPv4 [10] címet is kap. Ez lenne a tökéletes megoldás, azonban az IPv4-es címek szükségessége nem nyújt megoldást.

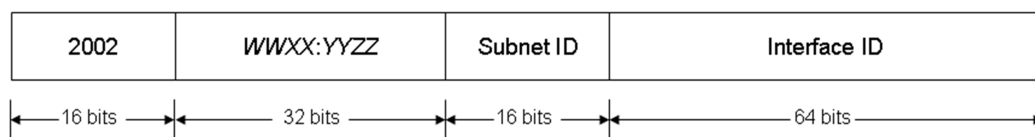
3.1.3. Dual Stack Lite

Az RFC 6333 draft [11] definiálja.

A kliensnél elhelyezett végponti eszköz (CPE) minden a kienstől érkező IPv6-os csomagot továbbít a szolgáltató felé, az pedig azt változtatás nélkül az Internet felé. Ha viszont a kienstől IPv4-es csomagot kap a CPE, akkor azt becsomagolja egy IPv6-os csomagba és a szolgáltatóhoz továbbítja, ahol lekerül róla az IPv6 csomagolás, majd egy carrier grade NAT eszköz publikus címmé alakítja. Segítségével kikerülhető a többszörös NAT-ból eredő probléma.

3.1.4. 6to4

Az RFC 3056 [12] definiálja részletesen. Segítségével egy csak IPv4-es címmel rendelkező gép magának oszthat ki egy egyedi global unicast IPv6-os címet a 2002:WWXX:YYZZ::/48 tartományból, ahol a wwxyyzz a host IPv4-es címe hexadecimális alakban (3. ábra). Ráadásul a host még szabadon gazdálkodhat 16 bitnyi alhálózati maszkkal is, ami 65536 darab 64 bites alhálózatot jelent. A host a többi 6to4-es számítógéppel (ezek szintén a 2002::/16-os címtartományban szerepelnek) szükséges kommunikáció esetén automatikus tunnelt épít ki, így ezekkel közvetlenül tud kommunikálni csak IPv4-et támogató, NAT-ot nem használó hálózatok esetén. Ha a 6to4-es host egy nem 6to4-es (például natív) IPv6-os állomást akar elérni, azt 6to4 relayek segítségével tudja csak megtenni. Ilyen esetekben ezekhez a relayekhez építi ki automatikusan a tunnelt.



3. ábra 6to4 IP cím felépítése [13]

A 6to4 relay kiszolgálók anycast protokollal érhetőek el, így nagy valószínűséggel a forgalom két irányba eltérő relay kiszolgálón fog áthaladni, aszimmetrikus működés várható.

A 6to4 kommunikációban résztvevő hostok (4. ábra) betöltött szerepeik szerint négyfelé oszthatók:

- 6to4 host

Minden olyan host, ami rendelkezik 6to4 IPv6-os címmel.

- 6to4 router

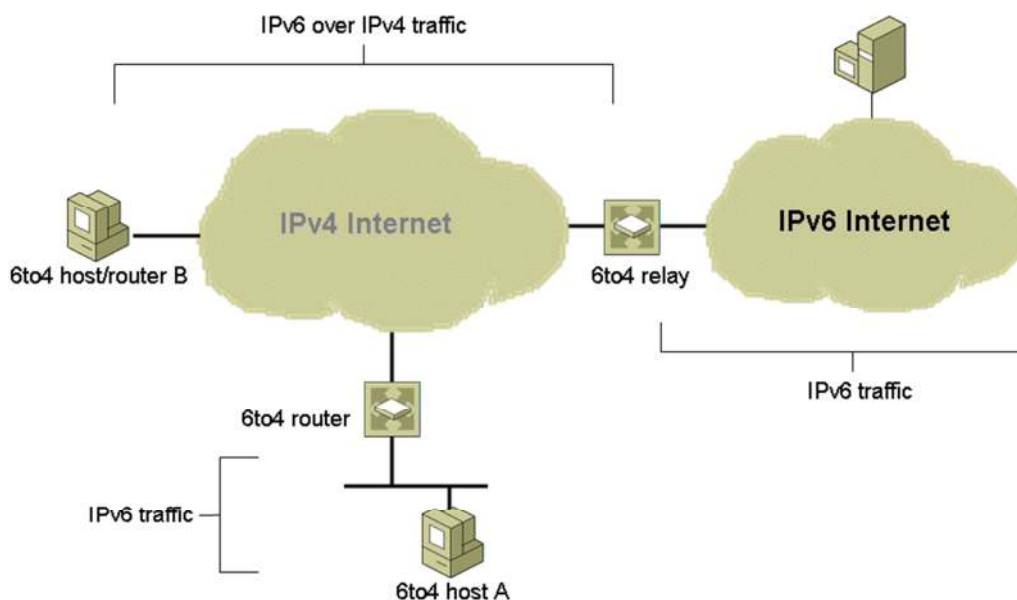
Azok a hostok, amik a 6to4 interfészen keresztül adatokat továbbítanak a 6to4 hostok, routerek, vagy relayek között.

- 6to4 host/router

Azok a hostok, amik 6to4-en keresztül kommunikálnak más 6to4 hostokkal, routerekkel, relayekkel, de más hostok csomagjait nem továbbítják.

- 6to4 relay

Olyan host, ami a 6to4 és a natív IPv6 közötti átjárhatóságot biztosítja mindkét irányban. Rendelkeznie kell IPv4, IPv6, 6to4-es címekkel és viszonylag gyors internet csatlakozással.



4. ábra 6to4 hálózat felépítése [13]

A kommunikáció lehetséges esetei:

- Azonos hálózaton lévő 6to4-es hostok.

Ez a legegyszerűbb eset. A két host egymással közvetlenül az IPv4-es címet használva képes kommunikálni. Semmilyen útválasztó, vagy relay alkalmazására nincs szükség. A küldő közvetlenül a címzetthez küldi a csomagot.

- Más hálózaton lévő 6to4 hostok.

Ez esetben már a két host között megjelenik egy, vagy több csak IPv4-et támogató útválasztó eszköz is. Problémát ez nem okoz, hiszen a két host felépít egy IPv4 csomagokban IPv6-os csomagokat továbbító tunnelt. Általánosan támogatott, jól működő NAT támogatása nincs a protokollnak, így az IPv4-es hálózatban nem alkalmazhatunk NAT-ot, és a 6to4 hostoknak is publikus IPv4 címmel kell rendelkezniük.

- 6to4 host natív IPv6 felé.

Ebben az esetben a 6to4 host egy IPv4 fölötti IPv6-os tunnelt épít ki a 6to4 relay felé, a relay a 6to4 interfészen kapott csomagot natív IPv6-os interfészen továbbítja. A 6to4 relay megkeresésére több megoldás is készült. Bizonyos kliensekben beállítható az IPv4-es címe, mások előre beégetett címen keresik a relayt. Az RFC-ben a 192.88.99.0/24-es anycast tartomány, valamint a 192.168.99.1-es cím került rögzítésre erre a célra [14], de például a Windows 7 a 6to4.ipv6.microsoft.com címet próbálja meg használni alapesetben. [13] Így fontos megoldani valamilyen interior routing protokoll (statikus, RIP, OSPF, IBGP) alkalmazásával, hogy az általunk választott anycast címet a hálózathoz el lehessen érni. Nem kell feltétlenül üzemeltetnünk 6to4 relayt a működéshez, mert az Interneten van több nyilvánosan elérhető, azonban ahogy a képen is látszik, ezek általában csak hosszú

útvonalon, lassabban érhetőek el (5. ábra). Ez viszont azt eredményezheti, hogy egy 6to4-es hosthoz közeli natív IPv6-os host csak lassan, jelen esetben, Németországon keresztül érhető el.

```
root@ns1:~# traceroute 192.88.99.1
traceroute to 192.88.99.1 (192.88.99.1), 30 hops max, 60 byte packets
 1 ahol-if.router0.ahol.hu (80.64.65.65)  1.308 ms  1.498 ms  1.655 ms
 2 border0.ahol.hu (80.64.64.17)  0.390 ms  0.383 ms  0.517 ms
 3 xe-10-2-0-210.bud-001-score-2-re0.interoute.net (84.233.170.53)  0.830 ms  0.825 ms  0.816 ms
 4 ae0-0.bud-001-score-1-re0.interoute.net (84.233.147.109)  17.204 ms  17.196 ms  17.187 ms
 5 ae2-0.prg-001-score-2-re0.interoute.net (84.233.138.213)  16.354 ms  16.347 ms  16.342 ms
 6 ae2-0.ber-alb-score-2-re0.interoute.net (84.233.138.234)  17.119 ms  16.782 ms  16.770 ms
 7 Gil-0-0.ber-007-access-3.interoute.net (84.233.146.158)  16.689 ms ae0-0.ber-alb-score-1-re0.interoute.net (84.233.146.129)  16.331 ms Gil-0-0.ber-007-access-3.interoute.net (84.233.146.158)  16.333 ms
 8 Gi3-0-0.ber-007-access-1.interoute.net (84.233.146.154)  16.319 ms PO5-0.ber-007-access-1.interoute.net (84.233.146.146)  16.595 ms  16.296 ms
 9 84.233.129.186 (84.233.129.186)  24.982 ms  25.670 ms  25.645 ms
10 te-1-1.bbr-muc-01.de.infra.cablesurf.de (85.233.35.214)  35.997 ms  35.404 ms  35.389 ms
11 85.233.35.133 (85.233.35.133)  37.793 ms * *
```

5. ábra Nyilvános 6to4 relay elérhetősége

- Natív IPv6 host 6to4 host felé.

Egy natív IPv6-os host felől a kommunikáció szintén egy 6to4 relay segítségével, anycast alkalmazásával valósul meg, itt azonban a 2002::/16-os címtartomány kerül hirdetésre. Itt csak akkor tudjuk befolyásolni, hogy melyik 6to4 relay kerüljön alkalmazásra, ha a natív host oldali hálózatot mi felügyeljük, ekkor a 2002::/16-os címtartományra érkező csomagokat nem szabad a default route felé irányítani (vagy ezt a prefixet az eBGP hirdetésekből elfogadni), hanem egy saját célra kialakított relay felé kell route-olni. Ez a relay fogja ezután a natív IPv6-os interfészen érkezett csomagot 6to4 tunnelben IPv4-es hálózaton továbbítani a célállomásra.

Ha azonban a felügyeletet nem mi látjuk el, akkor egy az Interneten hirdetett, közeli relay kerül alkalmazásra a natív host részéről (6. ábra). Így ebben az esetben is előfordulhat, hogy a csomag több országot megjárva jut el a szomszéd épületig.

```
*> 2002::/16          2001:1478:17:7::1
                        10          0 8928 39912 15598 i
```

6. ábra 2002::/16 a BGP routing táblában

A fentiekből jól látható, hogy az IP address spoofing könnyen problémát okozhat, hiszen egyszerűen küldhet bárki hamis IPv6-os címeiről csomagokat célállomások felé. A 6to4 támogatott a Windows XP/Vista/2003/2008/7 esetében és a Linux rendszerekben is.

A függelékben megtalálható az általam virtualizált környezetben megvalósított és kipróbált lehetséges 6to4 alkalmazások pontos leírása:

- IPv6 hálózat kialakítása Linux NAT44 és 6to4 router segítségével
- IPv6 hálózat kialakítása Windows Server 2008 R2 NAT és 6to4 router segítségével
- 6to4 hálózat kialakítása Linux 6to4 relay alkalmazásával
- 6to4 hálózat kialakítása Linux 6to4 Windows 7 klienssel

3.1.5. 6rd (IPv6 rapid deployment)

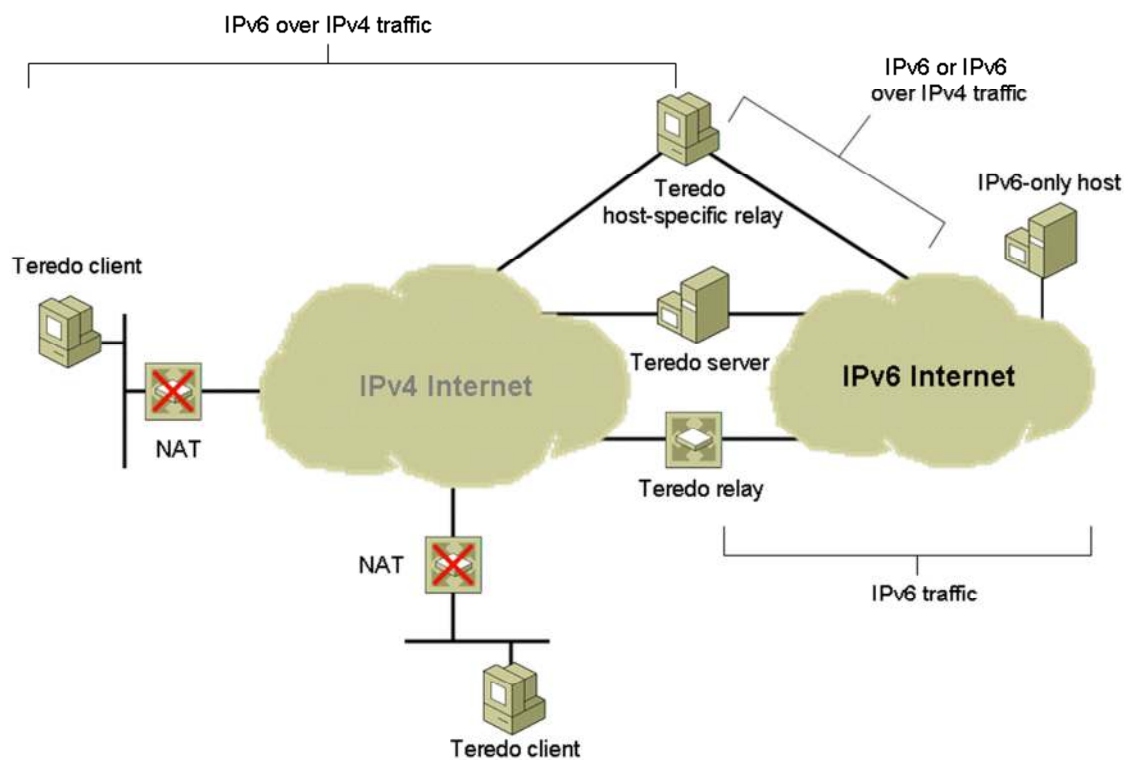
Az RFC 5569 definiálja. [15] A 6to4-ből alakították ki oly módon, hogy a teljes átalakítás az ISP hálózatában történik. A 6to4-től eltérően itt az ISP biztosít saját címtartományt a klienseknek, és így kizárólag csak a saját kliensei által igénybe vehető 6to4 relayeket tud üzemeltetni.

A Linux 2.6.33 verziótól kezdődően támogatja.

3.1.6. Teredo

RFC 4380 [16] és 6081 [17] definiálja pontosan.

Legfontosabb tulajdonsága, hogy (akár többszörös) NAT44 mögött lévő hostok is használhatják IPv6-os kommunikációra. Az IPv6-os csomagokat IPv4 UDP csomagokban továbbítja. A legvégső megoldásnak dolgozták ki, ami azt jelenti, hogyha natív IPv6, vagy ISATAP, vagy a 6to4 működik, akkor a Teredo nem aktiválódik (legalábbis Windowson).



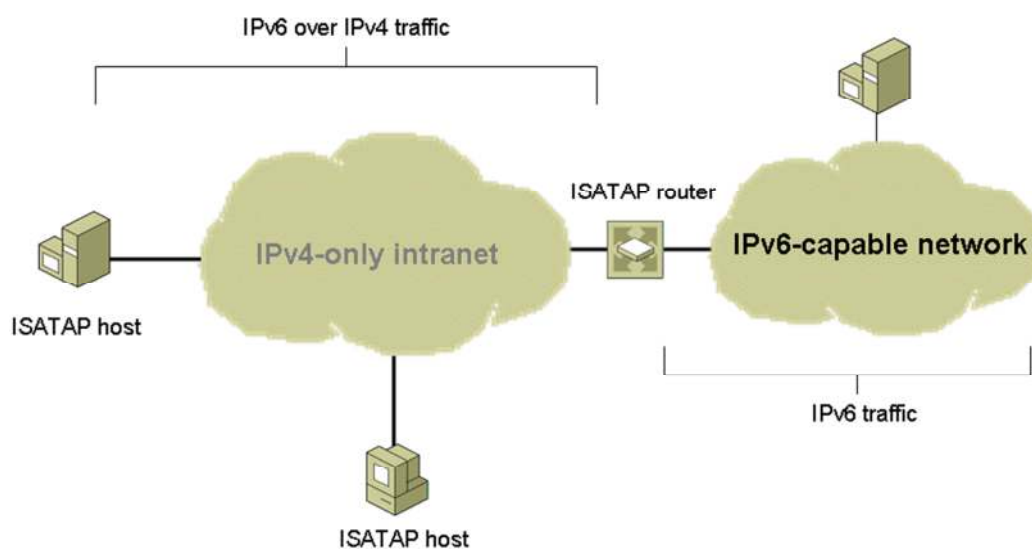
7. ábra Tipikus Teredo hálózat

Támogatja a Windows XP/2003/2008/Vista/7, de Linux és BSD rendszereken is létezik megvalósítása. [18]

3.1.7. ISATAP (Intra-Site Automatic Tunnel Addressing Protocol)

Az RFC 5214 [19] írja le, ami az RFC 4214-et [20] váltotta. Segítségével egy csak IPv4-et támogató route-olt, de NAT-ot nem alkalmazó hálózaton belül oldható meg az IPv6-ot és ISATAP-ot támogató hostok közötti kommunikáció. ISATAP router alkalmazása esetén pedig a hostok képesek kilépni az IPv6-os Internetre, ahol már nincs szükség ISATAP-ra (8. ábra). Ezek alapján

alkalmazása route-olt helyi hálózattal rendelkező vállalkozások esetén javasolt. Otthoni és kirodai környezetben általában nem található router a hálózatban (csak az Internetre csatlakozó router), így itt nem érdemes alkalmazni, bár megfelelő ISATAP router esetén lehetséges. Használható még olyan Internet szolgáltatónál, ahol az előfizetők szolgáltatóhoz csatlakozó útválasztó eszköze mögött nincsenek IPv4-es NAT-olt címekkel rendelkező node-ok, vagy egyáltalán nincsenek node-ok, csak maga a csatlakozó eszköz (pl. számítógép, mobiltelefon). Ilyenkor a szolgáltatónál telepített ISATAP router segítségével biztosítható az IPv6 kezelése abban az esetben is, ha a szolgáltató hálózatában találhatóak csak IPv4-et támogató eszközök.



8. ábra ISATAP hálózat felépítése [13]

A Linux kernel 2.6.25-ös verziójától, valamint a Windows XP/Vista/2003/2008 támogatja az ISATAP alkalmazását [21]. ISATAP router létezik Linuxra [22] és a Windows Server 2008 R2 is nyújt ilyen szolgáltatást.

Az ISATAP címeknek privát IPv4-es címek alkalmazásakor a `::0:5efe:w.x.y.z` –re kell végződnie, míg publikus IPv4-es címek esetében a `::200:5efe:w.x.y.z` –re. A Windows és a Linux is lehetővé teszi egy IPv6-os cím megadását oly módon, hogy a cím végét az IPv4-esnél megszokott módon adjuk meg. Például a `::0:5efe:172.16.0.2` megfelel a `::0:5efe:ac10:2` címnek. Így az ISATAP bevezetése maximum két darab /64-es alhálózatot foglal el a teljes IPv6-os címtartományunkból.

Egy ISATAP kliens az ISATAP routert egy az „ISATAP” nevű hosthoz tartozó IPv4-es cím DNS lekérdezésével találja meg.

A függelékben megtalálható az általam virtualizált környezetben megvalósított és kipróbált lehetséges ISATAP alkalmazások pontos leírása:

- ISATAP hálózat létrehozása Windows Server 2008 R2 ISATAP routerrel
- ISATAP hálózat létrehozása Linux ISATAP routerrel

- Linux ISATAP kliens telepítése és beállítása

3.1.8. SIIT (Stateless IP/ICMP Translation)

Az RFC 6145 írja le [23]. Az IPv4 és az IPv6 protokollok fejléce közötti átalakítást végzi. Tulajdonképpen egy 1:1-es stateless NAT a két protokoll között. Éppen ezért nagy az IP címigénye. Elavult, nem alkalmazott technológia.

3.1.9. NAT-PT/NAPT-PT

RFC 2766 definiálja [24]. Default gatewayként kell beállítani a NAT-PT/NAPT-PT átjárót a hálózatban, így az a DNS kéréseket módosíthatja, és a csak A (Address) rekorddal rendelkező hostokra irányuló lekérdezések esetén létrehoz egy AAAA rekordot is, amit a válaszban visszaad. Főként biztonsági okokból elavult protokoll [25]. Cisco IOS-en és Linuxon is létezik implementációja [26].

3.1.10. Application Level Gateway/Proxy

A legtöbbben ismerik a web proxy kifejezést, de ugyanilyen elven működő alkalmazás szintű átjárók, tűzfalak is léteznek, például ftp, smtp, pop3-ra. Jó, azonban költséges megoldás, hiszen minden támogatott protokollra el kell készíteni a megfelelő gatewayt. A legismertebb proxy a Squid web proxy [27]. Segítségével http, https, ftp protokollok esetében biztosítható az IPv4/IPv6 átjárás.

3.1.11. PortProxy

Az IPv4-es és IPv6-os nodeok közé helyezve, a DNS bejegyzést módosítva megoldható vele a címek átalakítása a két protokoll között, így a csak az egyik protokollt támogató kliens elérheti a csak másikat támogató szerveret. Fontos, hogy nem ALG, tehát ha az alkalmazás valamiért a magasabb szintű csomagban a címet is elküldi, akkor nem fog működni a PortProxyval. A Windows 2008/Vista/7 nyújt ilyen szolgáltatást.

4. DNS64 és NAT64 implementációk

4.1. Korábbi vizsgálatok

2012 során több publikáció is megjelent a DNS64 és a NAT64 teljesítőképességének vizsgálatáról.

Készült vizsgálat Linux rendszeren futtatott TAYGA NAT64, valamint TODD DNS64 megvalósítások összehasonlításáról a NAT44-el. [28] Összehasonlításra került egy saját fejlesztésű http ALG az Ecdsys NAT64 és Unbound+Ecdsys patch DNS64 megvalósításával. [29] És készült Linuxon futtatott DNS-ALG+Tomicki NAT-PT, Bind9 DNS64+Ecdsys NAT64, valamint Apache web proxy összehasonlítás is. [30]

Mindezek a mérések nem tették összehasonlíthatóvá a különböző NAT64 és DNS64 megvalósítások teljesítményét, mert az elvégzett mérések - a mérési módszerekből adódóan - minden esetben csak az adott DNS64 és NAT64 implementációk összteljesítményét mérték. Ugyan mindkét szolgáltatásra egyszerre van szükség a hálózatban, ugyanakkor a két szolgáltatás nem feltétlenül működik egy eszközön, így az egyes megvalósítások teljesítménye külön-külön is fontos információkat nyújt a hálózat tervezésekor.

Linuxon futtatott BIND9 DNS64 és a Tayga NAT64 implementációk vizsgálata történt már a győri Széchenyi István egyetemen, azonban szükség van több operációs rendszer és több DNS64/NAT64 megvalósítás összehasonlító vizsgálatára is. [31]

4.2. Implementációk kiválasztása

A kiválasztás fontos szempontja volt, hogy több szabadon elérhető és terjeszthető (GPL) DNS64 és NAT64 implementáció vizsgálata megtörténjen. Ráadásul minden fontosabb szabadon elérhető operációs rendszeren történjen meg ez a vizsgálat. Így az elvégzett munka eredményei a hálózati szakemberek számára a gyakorlatban is jól használhatóak, segítséget nyújtanak az IPv6 bevezetés tervezése során.

- Ecdsys NAT64

Az első NAT64 implementáció, amihez tartozik egy Unbounddal alkalmazható DNS64 patch is, így vizsgálata nem maradhatott volna ki, azonban stabilitása sajnos nem megfelelő. [31] Működéséhez kernel patchre is szükség van, de a patch az újabb kernelekhez nem érhető el. A projekt weboldala hosszú ideje nem frissül, az oldalon megtalálható legutolsó szoftververzió 2010.11.17-es dátummal került fel. [32] Ezek alapján alkalmazása nem javasolt, így vizsgálatát sem érdemes elvégezni.

- TAYGA NAT64

Egy Linuxra készült, nem kernelben működő stateless NAT64 implementáció. Alkotóinak célja egy olyan szoftver elkészítése volt, mely üzembiztosan használható NAT64 szolgáltatás nyújtására olyan helyeken, ahol egy dedikált NAT64 eszköz alkalmazása túlzás. [33]

Stateless működéséből fakadóan az IPv4 IPv6 megfeleltetést csak egy-egyben képes megvalósítani. Ezért alkalmazásakor szükség van a Linuxban megtalálható NAT44 (iptables) használatára is. Első lépésben a TAYGA a kienstől beérkező IPv6-os csomagot átalakítja egy privát IP forráscímmel rendelkező IPv4-es csomagra. Második lépésben pedig a Linux kernel NAT44-es modulja végzi a stateful NAT-ot, miközben átalakítja a csomag IPv4-es privát forrás címét egy publikus forrás címre (source nat, SNAT, port address translation), ami általában a NAT64 gateway nyilvános IPv4 címe. Ehhez a folyamathoz a TAYGA-nak szüksége van egy elegendően nagy privát IPv4-es címtartományra. Ha egy válaszcso­mag érkezik az Internet irányából a NAT64 gatewayre, úgy elsőként a stateful NAT44 segítségével a csomag IPv4-es célcíme és portja a megfelelő privát címre kerül átalakításra. Majd a stateless NAT64 segítségével a csomag átalakításra kerül IPv6-os csomagra, melynek célcímét a TAYGA saját nyilvántartásából nyeri ki.

- PF NAT64

Az OpenBSD 5.1 verziójában [34] található „Packet Filter” (PF) támogatja a stateful NAT64-et. Ez a támogatás az Ecdsys NAT64 implementációjának beépítésével, továbbfejlesztésével, és jelentős hibajavításokkal készült el. [35] A PF 2001 óta, az OpenBSD v3.0 megjelenésétől kezdődően része az OpenBSD rendszernek. [36] A PF a modern BSD rendszerekben található eszköz, melynek segítségével a csomagok manipulációja és szűrése nagyon sokrétűen oldható meg. Többek közt támogat különféle NAT-okat, load balancingot, logolást, és egy időben képes stateful és stateless csomagszűréseket végrehajtani. Az OpenBSD rendszerben megtalálható PF már hosszú ideje képes stateful NAT44-re, de az OpenBSD 5.1-től kezdődően már stateful NAT64 is megvalósítható segítségével. A PF-ben a NAT64 funkció „address family translation” néven található meg.

A PF stateful módban az address family translation segítségével több IPv6 kienst tud egy publikus IPv4-es cím mögé rejteni. A stateful módhoz a PF-nek egy állapotáblát (ún. state table) kell kezelnie, hogy a válaszként érkező IPv4-es csomagokhoz megtalálja a megfelelő IPv6-os állomás célcímét. A stateful módnak köszönhetően nincs szükség az egymás utáni stateless és stateful NAT alkalmazására, ráadásul egy állapotábla kezelése elégséges a megfelelő működéshez, szemben a TAYGA esetében alkalmazott megoldással. Ezek következtében a PF kisebb terhelést produkál a NAT64 gatewayen.

- BIND9 DNS64

A BIND a legelterjedtebb DNS szerver, mely a 9.8-as verziójától kezdődően tartalmazza a DNS64 támogatást is. [37] A BIND egy meglehetősen nagy és komplex szoftver, ami szinte minden DNS-sel kapcsolatos funkciót megvalósít (authoritatív, rekurzív, dinamikus, DNSSEC, stb.).

- TOTD DNS64

A BIND-del szemben a TOTD nem egy teljes értékű DNS szerver. Csak a DNS64 szolgáltatás megvalósítására készítették fejlesztői. [38] Annak ellenére, hogy fejlesztése viszonylag régóta nem folyik, kis mérete miatt jól alkalmazható, biztonságos program.

A BIND9 és a TOTD vizsgálata mind Linux, OpenBSD, és FreeBSD operációs rendszeren megtörtént.

5. DNS64 mérés

5.1. A mérési összeállítás

A vizsgálatok célja a különböző DNS64 implementációk teljesítményének vizsgálata és összehasonlítása volt. A teljesítményen túlmenően nagyon fontos az egyes szoftverek stabilitása is, így a munka ennek megfigyelésére is kiterjedt mind alacsony, mind pedig extrém nagy terhelés mellett.

5.1.1. Hálózat felépítése

A mérést a Széchenyi István Egyetem Távközlés-Informatikai Laborjában végeztem. A 9. ábra tartalmazza mérés során alkalmazott hálózati topológiát.

A számítógépek mindegyike 1000Base-TX, 1Gb sebességű összeköttetéssel, az ábrán látható módon két darab 3com Baseline 2948-SFP Plus switch segítségével kapcsolódik egymáshoz.

A mérések elvégzéséhez szükség volt egy olyan címtartományra, ami:

- Szisztematikusan hivatkozható
- Csak IPv4-es címekre lehet feloldani (csak A rekordok)
- Más hálózatoktól függő késleltetésektől mentes

Erre a célra saját címtartományt hoztunk létre a 10-{0..10}-{0..255}-{0..255}.zonat.tilb.sze.hu névtérben. Ezek a nevek a 10.0.0.0 – 10.10.255.255 IPv4 címekre mutattak a 192.168.100.105-ös címen elérhető, az ábra felső részén látható teacherb nevű számítógépen futó névszerver segítségével. Az ábra közepén látható DNS64 szerver végezte az IPv4-es címekhez tartozó A rekordokból az IPv6-os AAAA szintetizálását a 2001:738:2c01:8001:ffff:ffff:0a00:0000 – 2001:738:2c01:8001:ffff:ffff:0a0a:ffff címtartományra.

Az ábra alján lévő csak IPv6-os címmel rendelkező 8 darab azonos konfigurációjú Dell munkaállomás szimulálta a nagyszámú klienst, létrehozva a megfelelő terhelést a mérés elvégzéséhez.

5.1.2. Számítógép konfigurációk

A mérés elvégzéséhez egy olyan DNS64 szerver számítógépre volt szükség, ami a rendelkezésre álló számítógépekkel a mérés során túlterhelhető. A processzornak és memóriának a rendelkezésre álló legkisebb teljesítményű és méretű került kiválasztásra. Hálózati kártyaként két 1000Base-TX PCI csatlakozójú kártya került alkalmazásra. A PCI felület ugyan limitálja a maximális hálózati forgalmat, azonban a mérés során ez nem okoz problémát. A DNS64 kiszolgáló konfigurációja a következő:

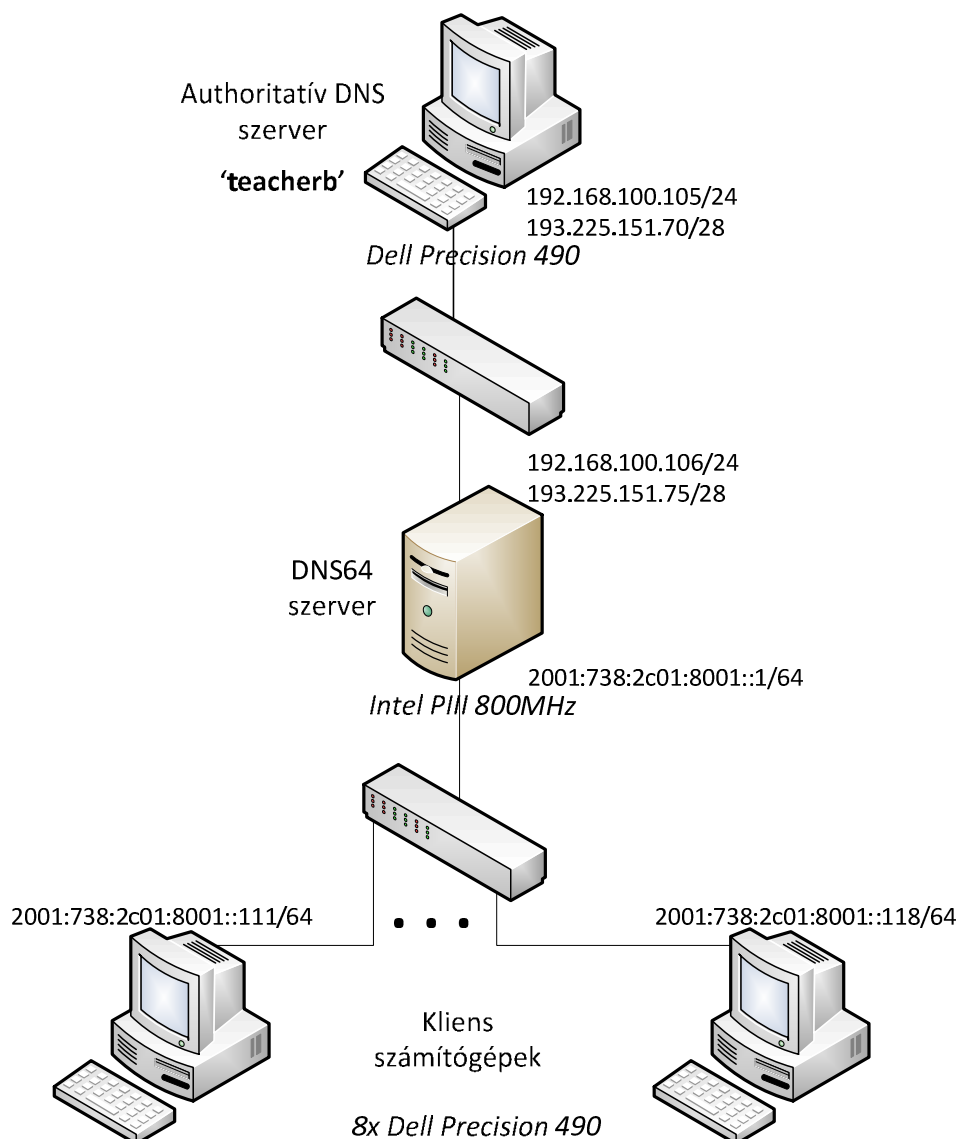
- Intel D815EE2U alaplap
- 800 MHz Intel Pentium III (Coppermine) processzor
- 128 MB, 133 MHz SDRAM
- Két darab 3Com 3c940 Gigabit Ethernet PCI hálózati kártya

Az összeállított hálózati konfiguráció manapság már nem mondható tipikusnak, hiszen minden korszerű számítógép lényegesen nagyobb teljesítményt produkál, ugyanakkor a rendelkezésre álló, a kliensek terhelését szimuláló eszközökkel csak így volt megoldható a megfelelő mérés elvégzése. A vizsgálat célja ráadásul nem a hardverelemek teljesítményének mérése, hanem a különböző szoftverek teljesítményének és megbízhatóságának vizsgálata volt ugyanazon a hardverkörnyezet esetében.

A 9. ábra tetején látható DNS szerver, valamint az ábra alján lévő, a terhelést előállító klienseket szimuláló 8 darab munkaállomás konfigurációja megegyezik. Ezek Dell Precision Workstation 490 típusú munkaállomások a következők szerint:

- DELL 0GU083, Intel 5000X chipset alaplap
- Kettő Intel Xeon 5130 2 GHz dupla magos processzor
- 2x2 GB + 2x1 GB 533 MHz DDR2 SDRAM (dual channel)
- Broadcom NetXtreme BCM5752 Gigabit Ethernet PCI Express hálózati kártya

Mindegyik számítógépen Debian Squeeze 6.0.3 Linux operációs rendszer futott. A Linuxon működő DNS64/NAT64 szolgáltatások mérése esetén a tesztszámítógépen is. Míg a többi mérésben ezen a számítógépen OpenBSD 5.1 és FreeBSD 9.0 verzió került telepítésre. A Linux és a Debian rendszer dual boot megoldással egy diszken, míg a FreeBSD egy külön diszken került telepítésre, azonban a diszk típusa mindkét esetben megegyezett.



9. ábra DNS64 mérés során alkalmazott hálózat felépítése

5.2. Mérési módszerek

5.2.1. IPv4 DNS szerver

A mérés során szükséges autoritatív DNS szerver funkcióját a 9. ábra tetején látható 192.168.100.105-ös IP című, teacherb.tilb.sze.hu DNS nevű számítógép töltötte be. A DNS szolgáltatást a Debain 6.0.3 disztribúcióban megtalálható BIND 9.7.3 nyújtotta. Ezen a szerveren semmiféle speciális beállítás elvégzésére nem volt szükség, kizárólag a megfelelő zóna állomány létrehozása és a BIND-be töltésére volt szükség.

A tilb.sze.hu zóna autoritatív szerverén delegálásra került a zonat.tilb.sze.hu zóna a teacherb számítógépre. Ezek után a már említett 10.0.0.0 – 10.10.255.255 IPv4 címtartományhoz a zóna állományt egy script segítségével lehet a leggyorsabban előállítani (10. ábra).

```
#!/bin/bash
cat > db.zonat.tilb.sze.hu << EOF
\$ORIGIN zonat.tilb.sze.hu.
\$TTL 1
@ IN SOA teacherb.tilb.sze.hu. kt.tilb.sze.hu. (
                2012081701 ; Serial
                28800 ; Refresh
                7200 ; Retry
                604800 ; Expire
                2 ) ; Min TTL
@ 86400 IN NS teacherb.tilb.sze.hu.
EOF

for a in {0..10}
do
    for b in {0..255}
    do
        echo '$'GENERATE 0-255 10-$a-$b-$ IN A \
            10.$a.$b.$ >> db.zonat.tilb.sze.hu
    done
done
echo "" >> db.zonat.tilb.sze.hu
```

10. ábra DNS zóna generáló script

A script eleje a SOA rekordot és egy NS rekordot hoz létre, majd a ciklus magjában kerül létrehozásra az érdeklő 11*256 darab A rekordokhoz szükséges sor. A „\$GENERATE 0-255 10-x-x-\$ IN A 10.x.x.\$” sorok segítségével elkerülhető, hogy 256-szor legyen szükség az „A” rekordokat tartalmazó sorok leírására, segítségével lerövidíthető a zóna állomány, mert a zóna betöltésekor a BIND egy-egy ilyen sort úgy értelmez, mintha a megfelelő 256 darab A rekordot olvasta volna be. A DNS64 vizsgálatához fontos, hogy a használt címekhez ne legyenek AAAA rekordok, csak az IPv4-es címet tartalmazó A-k. Ezért a zónát létrehozó script is csak A rekordokat generált.

5.2.2. DNS64 szerver

A két kiválasztott DNS64 implementáció (BIND9.8 és TOTD) mindhárom operációs rendszeren vizsgálatra került. A következőkben a vizsgálatnál alkalmazott beállításokat ismertetem részletesen.

Linux

A 9. ábra tartalmazza a hálózati vezérlőkhöz rendelt IP címeket. Az eth1, mely csak IPv6-os címmel rendelkezett, csatlakozott a szintén csak IPv6-os kliensekhez. Az eth2 pedig mind publikus, mind pedig privát IPv4-es címmel is rendelkezett. A kliensek IPv6 SLAAC (Stateless Address Autoconfiguration) beállításához a radvd (Router Advertisement Daemon) került telepítésre a teszt számítógépen. A 11. ábra tartalmazza a radvd alkalmazott beállításait.

```

interface eth2
{
    AdvSendAdvert on;
    AdvManagedFlag off;
    AdvSendAdvert on;
    prefix 2001:738:2c01:8001::/64
    {
        AdvOnLink off;
    };
    RDNSS 2001:738:2c01:8001::1 {};
};

```

11. ábra /etc/radvd.conf beállítások

A sok beérkező DNS64 kérés miatt a connection táblázat hamar megtelhet, ezért nagy forgalmú szerverek esetében szükség van a connection tracking kikapcsolására, vagy a táblázat méretének növelésére, vagy a hozzá tartozó timeout csökkentésére. A mérés során a timeout értéke 1 másodpercre került beállításra a 12. ábra szerint.

```

echo "1" > /proc/sys/net/netfilter/nf_conntrack_udp_timeout

```

12. ábra connection tracking timeout beállítása

OpenBSD és FreeBSD

A két operációs rendszer nagymértékű hasonlósága miatt, azok beállítása is megegyezett. A hálózati kártyákhoz rendelt IP címek megegyeznek a Linuxnál alkalmazottakkal. (9. ábra) Az SLAAC alkalmazásához szükséges beállításokat pedig a 13. ábra tartalmazza.

```

default:\
:chlim#64:raflags#0:rltime#1800:rttime#0:retrans#0:\
:pinfolags="1a":vltime#2592000:pltime#604800:mtu#0:
sk1:\
:addr="2001:738:2c01:8001::":prefixlen#64:tc=default:

```

13. ábra /etc/rtadvd.conf beállítás

A nagy forgalmú DNS szerverek miatt szükséges módosítást a 14. ábra /etc/pf.conf beállítás tartalmazza.

```

set timeout interval 1
set limit states 40000

pass in on sk0 inet proto udp from any port 53 to 193.225.151.75 no state
pass out on sk0 inet proto udp from 193.225.151.75 to any port 53 no state
pass in on sk1 inet6 proto udp from any to 2001:738:2c01:8001::1 port 53 no state
pass out on sk1 inet6 proto udp from 2001:738:2c01:8001::1 port 53 to any no state

```

14. ábra /etc/pf.conf beállítás

Önmagában az első, vagy a második sor, vagy az utolsó négy sor együttes alkalmazása is elég lett volna a Linuxnál alkalmazott megoldáshoz hasonlóan, de a BSD PF nagyfokú rugalmasságát szerettem volna kihasználni a mérések során, így több megoldást is kerestem a problémára.

BIND 9.8 DNS resolver DNS64

A Linux és az OpenBSD vizsgált verziói még nem tartalmazták a DNS64 támogatást is nyújtó BIND 9.8-as verzióját, így a két operációs rendszer esetében forráskódból fordítottam le a legújabb BIND verziót. A FreeBSD 9.0 már a BIND 9.8.1-P1 verzióját tartalmazza, így nem volt szükség a BIND lefordítására.

Egyik esetben sem volt szükség a BIND konfigurációjának módosítására, kizárólag a DNS64 beállítást végeztem el a konfigurációs állományban (15. ábra). A mérések során nem a well-known prefix került alkalmazásra, hanem egy hálózatspecifikus tartomány, amely a hálózat végéről került kiválasztásra, így beállításra is ez került.

```
dns64 2001:738:2c01:8001:ffff:ffff::/96 {  
};
```

15. ábra DNS64 beállítás a named.conf-ban

TOTD DNS64

A TOTD nem egy teljes értékű DNS szerver, kizárólag forwarderként tud működni. A mérés elvégzéséhez a forwarder paraméter segítségével recurserként a létrehozott teszt zóna (zonat.tilb.sze.hu) autoritativ szervere (teacherb) került beállításra (16. ábra). Valós környezetben ez szinte biztosan nem fordul elő, azonban ezzel a módszerrel elkerülhető volt, hogy nem befolyásolható hálózati szakaszok torzítsák az elvégzett mérések eredményeit.

```
forwarder 192.168.100.105 port 53  
prefix 2001:738:2c01:8001:ffff::  
retry 300
```

16. ábra /etc/totd.conf beállítás

BIND 9.8 DNS forwarder DNS64

A BIND és a TOTD eredményeinek korrekt összehasonlításához szükséges, hogy azonos feltételekkel történjenek a mérések. Ezért a BIND mérését a BIND-et forwarderként konfigurálva is elvégeztem. Ehhez a beállításokat a named.conf állományban kell megtenni (17. ábra).

```
forwarders { 192.168.100.105; };  
forward only;
```

17. ábra BIND forwarder konfiguráció named.conf-ban

Kliens számítógépek

A 8 darab azonos konfigurációjú számítógépen a DNS szerver beállítását végeztem el a /etc/resolv.conf állományban. Erre a beállításra csak a két BSD rendszer mérésénél volt szükség, mert a Linux esetében a radvd daemon segítségével a DNS szerver beállítása automatikusan megtörtént.

```
nameserver 2001:738:2c01:8001::1
```

18. ábra Névkiszolgáló beállítás /etc/resolv.conf állományban

DNS64 mérés

A mérések során a processzor kihasználtság és a memórafoglalás is folyamatos rögzítésre került a DNS64 tesztserveren. A mérés megfelelő működésének ellenőrzéséhez ezeket a paramétereket a teacherb nevű DNS serveren is folyamatosan rögzítettem, hogy látható legyen, ha esetleg a terhelést a nem vizsgált DNS server nem viselné el, ezáltal a mérési eredményeket befolyásolná.

A mérések 1, 2, 4 majd 8 darab kliens egyidejű működése mellett kerültek elvégzésre, és az egyes mérések között a vizsgált DNS64 server újraindítása is megtörtént, hogy biztosítható legyen a felhasznált memória minél pontosabb mérése. Azért, hogy a DNS caching funkciója ne befolyásolja a mérést, minden kliensnek külön névtartományt kellett lekérdeznie a teszt céljára létrehozott zonat.tilb.sz.hu zónából, ráadásul mindegyik címet csak egyszer.

```
#!/bin/bash
i=`cat /etc/hostname|grep -o .$.`
rm dns64-$i.txt
do
  for b in {0..255}
  do
    /usr/bin/time -f "%E" -o dns64-$i.txt -a ./dns-st-c.sh $i $b
  done
done
```

19. ábra dns-test.sh mérő script

A megfelelő pontosságú adatokhoz 256 mérés elvégzése után kerül csak rögzítésre a végrehajtási idő, és egy mérési ciklus 256-szor kerül végrehajtásra. (Az egy mérési ciklust végrehajtó script a 20. ábraán látható.) Így összesen 2^{16} DNS kérés kerül mérésre, és minden egyes mérési adat ebből 256 DNS kérés végrehajtási idejét tartalmazza. A DNS kéréshez a Linux host [39] parancsa, az időméréshez pedig a GNU time [40] parancsa került felhasználásra. A kliensek egyidejű indítása (a 2, 4, 8 darab kliens esetében) a KDE terminál emulátorának, a KONSOLE-nak a „Send Input to All Sessions” funkciójával történt.

```
#!/bin/bash
for c in {0..252..4} # that is 64 iterations.
do
  host 10-$1-$2-$c.zonat.tilb.sze.hu &
  host 10-$1-$2-$((c+1)).zonat.tilb.sze.hu &
  host 10-$1-$2-$((c+2)).zonat.tilb.sze.hu &
  host 10-$1-$2-$((c+3)).zonat.tilb.sze.hu &
done
```

20. ábra dns-st-c.sh mérő script

A mérést végző scriptben (20. ábra) lévő ciklus 64-szer hajtódik végre, és a ciklus magjában 4 párhuzamos DNS kérés van. Így összesen 256 kérést hajt végre a script, amit a 19. ábra scriptje 256 alkalommal hív meg.

A könnyű összehasonlíthatóság érdekében a mérés során a terhelést produkáló kliensek számának folyamatos növelése történt 1, 2, 4, 8 darabra. A mérési részeredmények pedig minden kliensen a dns64-X.txt nevű állományban rögzítésre kerültek, ahol X a kliens sorszámát jelöli.

A DNS64 szolgáltatást nyújtó tesztszámítógép processzorkihasználtságának és szabad memóriájának mérése Linux esetében a `dstat -t -c -m -l -p --unix --output load.csv` paranccsal, míg BSD rendszereken a `vmstat -w 1 >load.txt` paranccsal történt.

5.3. Mérési eredmények

5.3.1. BIND rezolverként Linuxon futtatva

A mérési eredményeket az 1. táblázat mutatja. A táblázat első sora a terhelést produkáló kliensek számát mutatja. Tehát, hogy egy időben hány kliens hajtotta végre a 19. ábrán bemutatott scriptet. Kétszer annyi kliens, kétszeres terhelést produkál. A 2. sorban látható 256 host parancs átlagos végrehajtási ideje. Mivel a mérés során kliensenként 256 eredmény keletkezik, így ebben a sorban a mért válaszadási idők átlaga került megjelenítésre. A 3. sorban a végrehajtási idők szórása, majd a 4. sorban ezeknek legmagasabb mért értéke látható. Megfigyelhető, hogy a szórás minden esetben alacsony értékű, valamint a maximum érték nem tér el nagymértékben az átlagos értéktől. Az 5. sorban a DNS64 szolgáltatást nyújtó tesztszámítógép processzorának terheltsége látható. A 6. sor tartalmazza a processzorterhelési értékek szórását. A 7. sor a memóiafelhasználást mutatja. Ez a paraméter a mérés megkezdésekor és befejezésekor a tesztszámítógépen rendelkezésre álló szabad memória mérete alapján került kiszámításra. Mivel a számítógépen más szolgáltatások is futnak, melyek a szabad memória méretét befolyásolják, ez az érték nem határozható meg teljes pontossággal. A 7. sor az egy másodperc alatt átlagosan kiszolgált kérések számát mutatja. Kiszámítása a kliensek száma és a 256 host parancs végrehajtási ideje alapján történt.

1	Kliensek száma	1	2	4	8
2	256 host parancs végrehajtási ideje [ms]	1,242	1,862	3,748	7,541
3	szórása	0,018	0,050	0,076	0,282
4	és maximuma	1,550	2,260	3,970	12,690
5	CPU kihasználtsága [%]	67,66	96,63	100	100
6	és szórása	1,2	2,3	0	0
7	Memória foglalása [MB]	36	49	52	50
8	Kiszolgált kérések száma [db/mp]	206	275	273	272

1. táblázat DNS64 teljesítmény BIND, Linux, Rezolver

A mérési eredmények alapján kijelenthető, hogy:

- A terhelés növekedésével (kliensek számának növelésével) nem lép fel nagymértékű teljesítménycsökkenés, a túlterhelés alatt nem omlik össze a rendszer. Hiába éri el a 100%-ot a processzorkihasználtság, a rendszer megközelítőleg a maximális ~275 klienst szolgálja ki másodpercenként, a válaszidő is megközelítőleg lineárisan nő a túlterhelést előidéző 2, 4, 8 darab kliens esetében.
- A rendszer maximális teljesítőképességét körülbelül két kliens esetén elérte. 275 kérés/mp feletti értéket nem sikerült elérni, mely értéknél már a processzor is közel teljes mértékben kihasználásra került. A kliensek számának további növelése a teljesítmény növekedése helyett stagnálást, kismértékű csökkenést eredményezett. Aminek oka, hogy a tesztelés során használt script nem generált újabb kérdéseket, amíg nem kapott választ az előző kérésekre, a rendszer viszont már nem tudta a kéréseket gyorsabban kiszolgálni.
- A memóriefoglalás viszonylag alacsony értéket mutatott. Valószínűsíthető, hogy ha a tesztrendszer képes lett volna több kérés kiszolgálására, akkor a memóriefoglalás is növekedett volna a cache következtében, azonban ez sem eredményezett volna jelentősen nagyobb memória felhasználást.

A teszt egy nagyon fontos eredményt produkált, hiszen a rendszer a sokszoros túlterhelés esetén is megbízhatóan teljesített, nem tett kiszámíthatatlan dolgokat, nem omlott össze, teljesítménye csak nagyon kis mértékben, fokozatosan csökkent (, teljesítette a graceful degradation követelményt). Így biztonsággal használható akár nagy forgalmú, nagy megbízhatóságot kívánó szolgáltatói környezetben is.

A rendszer stabilitásáról fontos információt nyújtanak a 8 klienssel elért többszörös túlterhelés esetén produkált válaszidők is. A válaszidők szórása (0,282) kevesebb, mint 4%-a az átlagos válaszidőnek (7,541), ráadásul a maximális válaszidő (12,69) sem éri el az átlagos érték kétszeresét. Kijelenthető, hogy a rendszer stabil, megfelelően teljesít akár többszörös túlterhelés esetén is.

Ezek alapján a BIND Linux alatt futtatva megfelelő választás DNS64 szolgáltatás nyújtásához.

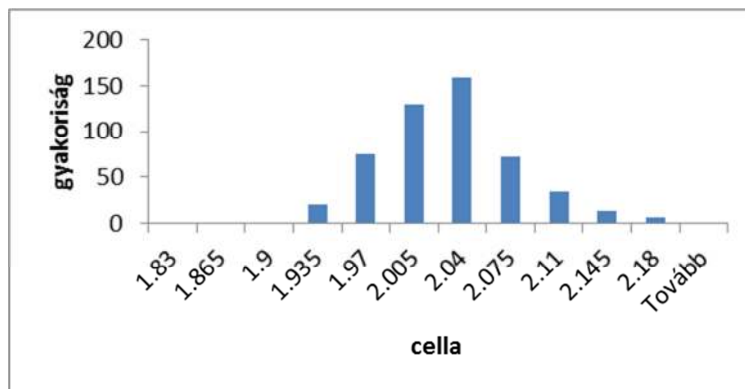
5.3.2. BIND rezolverként OpenBSD-n futtatva

A mérési eredmények összesítését a 2. táblázat tartalmazza.

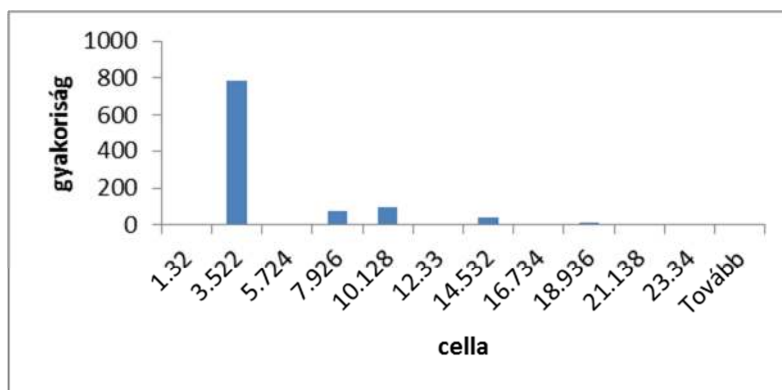
1	Kliensek száma	1	2	4	8
2	256 host parancs végrehajtási ideje [ms]	1,259	2,013	4,194	7,626
3	szórása	0,009	0,051	3,564	7,305
4	és maximuma	1,300	2,180	23,340	52,690
5	CPU kihasználtsága [%]	72,94	96,90	97,02	98,12
6	és szórása	1,8	2,2	5,5	5,6
7	Memória foglalása [MB]	35	45	45	45
8	Kiszolgált kérések száma [db/mp]	203	254	244	269

2. táblázat DNS64 teljesítmény BIND, OpenBSD, Rezolver

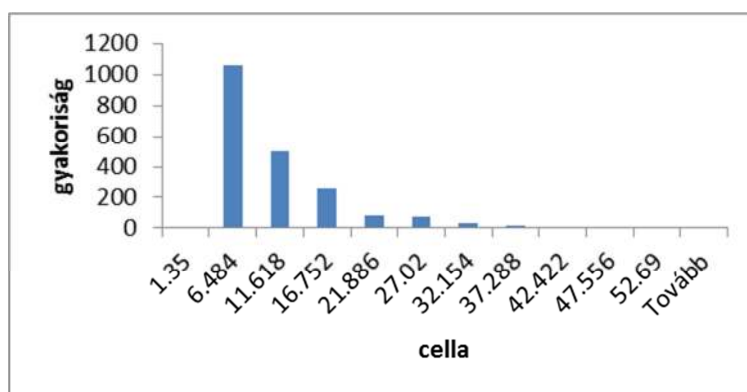
Az átlagos válaszidők megközelítik a Linux esetében mért értékeket, azonban a 8 kliensnél mért végrehajtási idő szórása (7,306) megközelíti az átlagot (7,626), ráadásul a maximum érték (52,69) az átlagos értéknek közel hétszerese. A további elemzések elvégzéséhez a mért értékeket hisztogramon is ábrázoltam (21. ábra, 22. ábra, 23. ábra).



21. ábra 256 darab host parancs átlagos végrehajtási ideje BIND, OpenBSD, 2 kliens esetén



22. ábra 256 darab host parancs átlagos végrehajtási ideje BIND, OpenBSD, 4 kliens esetén



23. ábra 256 darab host parancs átlagos végrehajtási ideje BIND, OpenBSD, 8 kliens esetén

A 21. ábra mutatja a két kliens esetén mért értékek eloszlását. Jól látszik, hogy az átlagos érték (2,013) körül az értékek egy viszonylag keskeny tartományban helyezkednek el. A következő két ábrán a 4 és 8 darab klienssel produkált nagymértékű túlterhelés eredményei láthatóak. Jól megfigyelhető, hogy az átlagokhoz (4,194 és 7,626) képest az értékek jóval szélesebb tartományban mozognak. (Mindhárom ábra tartalmazza az összes mért értéket, így rendre $2 \cdot 256 = 512$, $4 \cdot 256 = 1024$, $8 \cdot 256 = 2048$ darab érték ábrázolása szerepel rajtuk.)

A túlterhelés esetén tapasztalható kiszámíthatatlan viselkedése miatt a BIND alkalmazása OpenBSD alatt nem ajánlott DNS64 szolgáltatás nyújtására olyan környezetekben, ahol előfordulhat a terhelés túlzott növekedése.

5.3.3. BIND rezolverként FreeBSD-n futtatva

A mérési eredmények összesítését a 3. táblázat tartalmazza.

1	Kliensek száma	1	2	4	8
2	256 host parancs végrehajtási ideje [ms]	1,815	2,979	5,900	11,923
3	szórása	0,018	0,034	0,075	0,090
4	és maximuma	2,000	3,070	6,380	12,140
5	CPU kihasználtsága [%]	78,58	97,51	100	100
6	és szórása	1,8	1,6	0,0	0,0
7	Memória foglalása [MB]	30	36	42	53
8	Kiszolgált kérések száma [db/mp]	141	172	174	172

3. táblázat DNS64 teljesítmény BIND, FreeBSD, Rezolver

A legstabilabb, legjobban kiszámítható eredményt ez az összeállítás produkálta, azonban a legrosszabb átlagos válaszidők is itt figyelhetők meg. Ennek a rossz teljesítménynek legvalószínűbb oka, hogy a FreeBSD biztonsági okokból jailben futtatja a BIND szolgáltatást [41].

A FreeBSD-n futtatott BIND megfelelő választás olyan helyekre, ahol a biztonság akár a teljesítmény rovására is mehet. A Linuxon futtatott BIND konfigurációval összehasonlítva az

átlagosan kiszolgált kérések száma másodpercenként megközelítőleg harmadával lett kevesebb (141/206=68,45%; 172/275=62,55%; 174/273=63,74; 172/272=63,24%).

5.3.4. BIND forwarderként Linuxon futtatva

A mérési eredmények összesítését a 4. táblázat tartalmazza.

1	Kliensek száma	1	2	4	8
2	256 host parancs végrehajtási ideje [ms]	1,127	1,542	3,183	6,318
3	szórása	0,049	0,037	0,081	0,104
4	és maximuma	1,650	1,680	3,310	6,470
5	CPU kihasználtsága [%]	61,77	95,56	100	100
6	és szórása	4,1	2,3	0,0	0,0
7	Memória foglalása [MB]	40	58	57	57
8	Kiszolgált kérések száma [db/mp]	227	332	322	324

4. táblázat DNS64 teljesítmény BIND, Linux, Forwarder

Az ugyanilyen konfigurációban működő, de rezolverként konfigurált megoldással szemben megközelítőleg 20%-os teljesítménynövekedést produkált (227/206=110,19%; 332/275=120,73%; 322/273=121,61%; 324/272=119,12%) alacsonyabb processzorkihasználtság mellett. Ennek oka, hogy az érdemi munkát a teacherb nevű, nagyobb teljesítményű DNS szerver végezte.

5.3.5. BIND forwarderként OpenBSD-n futtatva

A mérési eredmények összesítését az 5. táblázat tartalmazza.

1	Kliensek száma	1	2	4	8
2	256 host parancs végrehajtási ideje [ms]	1,211	1,749	3,633	7,291
3	szórása	0,008	0,051	2,900	7,800
4	és maximuma	1,230	1,910	17,510	37,890
5	CPU kihasználtsága [%]	67,46	97,21	98,11	98,48
6	és szórása	1,9	2,4	5,4	6,0
7	Memória foglalása [MB]	37	52	49	47
8	Kiszolgált kérések száma [db/mp]	211	293	282	281

5. táblázat DNS64 teljesítmény BIND, OpenBSD, Forwarder

A Linux rendszernél végzett összehasonlításhoz hasonlóan itt is nőtt a másodpercenként kiszolgált kérések száma. A 4 és 8 darab kliens esetén mért végrehajtási idők szórása és maximuma ugyanúgy nem megfelelő értékeket produkál, mint a rezolver esetében.

5.3.6. BIND forwarderként FreeBSD-n futtatva

A mérési eredmények összesítését a 6. táblázat tartalmazza.

1	Kliensek száma	1	2	4	8
2	256 host parancs végrehajtási ideje [ms]	1,711	2,831	5,475	11,126
3	szórása	0,014	0,037	0,082	0,099
4	és maximuma	1,790	2,950	5,590	11,390
5	CPU kihasználtsága [%]	77,15	97,32	100	100
6	és szórása	1,8	1,5	0,0	0,0
7	Memória foglalása [MB]	31	37	39	37
8	Kiszolgált kérések száma [db/mp]	150	181	187	184

6. táblázat DNS64 teljesítmény BIND, FreeBSD, Forwarder

Az azonos konfigurációban futtatott, de rezolverként konfigurált BIND-hez viszonyítva itt is megfigyelhető a stabilitás és a kismértékű teljesítménynövekedés ($150/141=106,38\%$; $181/172=105,23\%$; $187/174=107,47\%$; $184/172=106,98\%$).

5.3.7. TOTD forwarderként Linuxon futtatva

A mérési eredmények összesítését a 7. táblázat tartalmazza.

1	Kliensek száma	1	2	4	8
2	256 host parancs végrehajtási ideje [ms]	0,791	1,310	2,158	4,348
3	szórása	0,038	3,863	3,754	5,301
4	és maximuma	1,370	64,950	63,730	68,540
5	CPU kihasználtsága [%]	38,00	58,05	80,16	84,79
6	és szórása	2,4	27,1	27,5	29,2
7	Memória foglalása [MB]	1,0	1,1	1,6	0,8
8	Kiszolgált kérések száma [db/mp]	324	391	474	471

7. táblázat DNS64 teljesítmény TOTD, Linux, Forwarder

A TOTD Linuxon futtatva rendkívül kevés memóriával beéri, és sebessége jelentősen felülmúlja ($324/227=142,73\%$; $391/332=117,77\%$; $474/322=147,20\%$; $471/324=145,37\%$) a Linuxon forwarderként futtatott BIND-del produkáltat, miközben processzor kihasználása megközelítőleg azonos azzal. Az alacsony memóriaigény a cache hiányának következménye, ami azonban azt is eredményezi, hogy nagyobb hálózati forgalmat generál, hiszen minden kérést továbbít a rezolver szervernek. A mérési eljárás miatt a cache egyik esetben sem került kihasználásra, azonban a BIND

ettől függetlenül letárolta benne a rezolúciók eredményét. Valós alkalmazás estén azonban a cache funkció mindenképpen előnyt jelent.

Egy kliens esetén a TOTD kiválóan működik, de a többi esetben, nagyobb terhelésnél már a válaszidő több mint egy perces, ami nem elfogadható egy DNS szerver esetében. Magas terhelés esetén a TOTD többször is megállt egy perces időtartamokra, majd az egy perc leteltével folytatta működését. Ráadásul hasonló viselkedést produkált mindhárom vizsgált operációs rendszeren futtatva. Így a TOTD ugyan gyors és kicsi a memóriaigénye, de csak kisforgalmú, vagy korlátozott forgalmú helyeken alkalmazható biztonsággal. Előnyei miatt mindenképpen érdemes a programkódot áttanulmányozni, a probléma okát feltárni és kijavítani.

5.3.8. TOTD forwarderként OpenBSD-n futtatva

A mérési eredmények összesítését a 8. táblázat tartalmazza.

1	Kliensek száma	1	2	4	8
2	256 host parancs végrehajtási ideje [ms]	0,808	1,118	2,276	4,445
3	szórása	0,010	0,141	2,724	5,468
4	és maximuma	0,830	3,060	28,560	41,080
5	CPU kihasználtsága [%]	48,99	79,98	76,51	78,34
6	és szórása	2,2	6,6	13,4	11,4
7	Memória foglalása [MB]	0,2	0,1	1,1	0,4
8	Kiszolgált kérések száma [db/mp]	317	458	450	461

8. táblázat DNS64 teljesítmény TOTD, OpenBSD, Forwarder

A Linuxon futtatott TOTD-vel összehasonlítva közel azonos eredményeket produkált, azonban stabilitása felülmúlta azt. 1 és 2 darab kliens esetén a végrehajtási idő szórása és maximuma is normális értéket mutat, míg 4 és 8 darab kliensnél már itt is megjelentek a használhatatlan értékek. Ezek azonban már nem érik el az egy perces értéket, mint a Linux esetében. A közel fél perces válaszidő miatt alkalmazása szintén csak korlátozott terhelés mellett javasolt, azonban stabilitása miatt a TOTD futtatása OpenBSD rendszeren javallott a Linux helyett.

5.3.9. TOTD forwarderként FreeBSD-n futtatva

A mérési eredmények összesítését a 9. táblázat tartalmazza.

1	Kliensek száma	1	2	4	8
2	256 host parancs végrehajtási ideje [ms]	0,923	1,127	2,580	4,750
3	szórása	0,676	0,061	6,356	6,181
4	és maximuma	5,800	1,300	65,650	69,700
5	CPU kihasználtsága [%]	45,01	78,66	69,54	83,23
6	és szórása	6,4	5,2	41,2	34,2
7	Memória foglalása [MB]	3,0	0,7	0,8	0,9
8	Kiszolgált kérések száma [db/mp]	277	454	397	431

9. táblázat DNS64 teljesítmény TOTD, FreeBSD, Forwarder

A stabilitás ugyanúgy itt is problémát okoz, mint a másik két operációs rendszer esetében, így alkalmazása itt is csak korlátozott terhelés esetén javasolt.

5.4. Értékelés

A mérések eredményeit összehasonlítva a BIND rezolverként legjobban Linux-on futtatva teljesített. Sebességben ugyanakkor az OpenBSD-n futó BIND hasonlóan teljesített, de jelentős terhelés esetén kiszámíthatatlanná, ezáltal használhatatlanná vált. A TOTD nagy terhelés esetén kiszámíthatatlanul viselkedik, így olyan helyeken érdemes megfontolni használatát, ahol a terhelés limitált. Ilyenek lehetnek az embedded rendszerek, mint amilyen egy otthoni Wi-Fi router is. De egyszerű telepítése és kis erőforrásigénye miatt érdemes megfontolni a kliens oldali alkalmazását is.

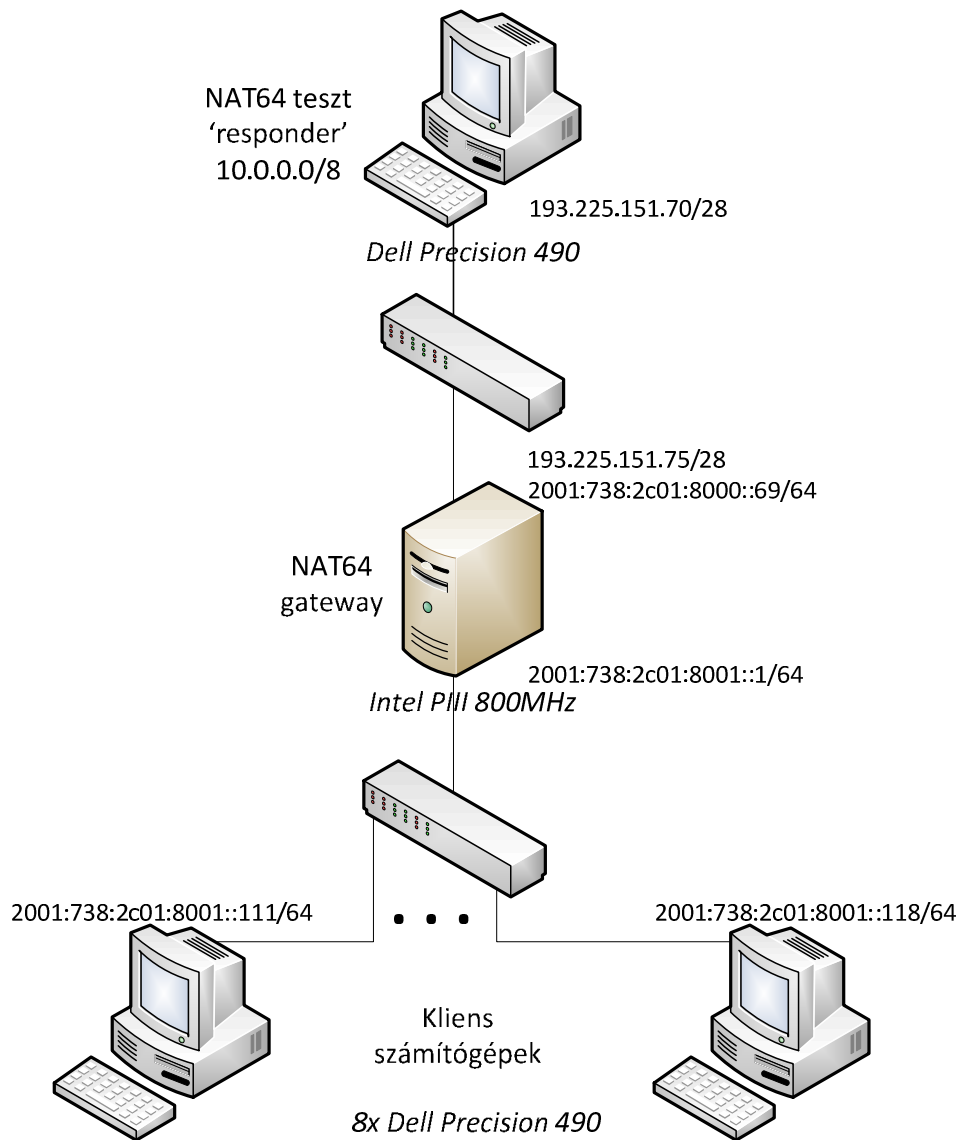
6. NAT64 mérés

6.1. A mérési összeállítás

A hálózat fizikai felépítése megegyezik a DNS64 mérésnél alkalmazottal. Eltérés az egyes számítógépek funkcióiban és a címkiosztásban van. A 24. ábra tartalmazza a hálózati topológiát.

6.2. Mérési módszerek

A 24. ábra tartalmazza a hálózati vezérlőkhöz rendelt IP címeket. Az eth1, mely csak IPv6-os címmel rendelkezik, csatlakozott a szintén csak IPv6-os kliensekhez. Az eth2 pedig mind publikus IPv4, mind pedig globális IPv6-os címmel is rendelkezik.



24. ábra NAT64 mérés során alkalmazott hálózat felépítése

6.2.1. Linux alapú TAYGA

A 24. ábra tartalmazza a hálózati vezérlők pontos beállításait a Linux tesztszámítógépen.

```
#Belso
auto eth1
iface eth1 inet6 static
address 2001:738:2c01:8001::1
netmask 64
up sleep 1
up echo 0 > /proc/sys/net/ipv6/conf/eth1/autoconf
up echo 0 > /proc/sys/net/ipv6/conf/eth1/accept_ra
post-up sleep 1
post-up /root/nat64-config.sh

#Kulso
auto eth2
iface eth2 inet static
address 193.225.151.75
netmask 255.255.255.240
gateway 193.225.151.65
iface eth2 inet6 static
address 2001:738:2c01:8000::69
netmask 64
gateway 2001:738:2c01:8000::1
pre-up echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
pre-up echo 0 > /proc/sys/net/ipv6/conf/eth2/autoconf
pre-up echo 0 > /proc/sys/net/ipv6/conf/eth2/accept_ra
```

25. ábra Linux hálózati vezérlők beállítása

A kliensek IPv6 SLAAC (Stateless Address Autoconfiguration) beállításához a radvd (Router Advertisement Daemon) került telepítésre a teszt számítógépen a DNS64 méréssel megegyező módon. A 11. ábra tartalmazza a radvd alkalmazott beállításait.

Ezután a TAYGA telepítésére, majd a 26. ábra szerinti konfigurálására került sor.

```
tun-device nat64
ipv4-addr 172.16.0.1
dynamic-pool 172.16.0.0/12
prefix 2001:738:2c01:8001:ffff:ffff::/96
```

26. ábra a TAYGA beállítása a /etc/tayga.conf állományban

A 27. ábraán látható script segítségével lehetett a TAYGA beállításának automatizálását elvégezni. Itt került engedélyezésre az IPv4 és IPv6 routing, valamint a NAT beállítások elvégzése is itt történt.

```
#!/bin/bash
tayga --mktun
ip link set nat64 up
#Privát IPv4 címtartomány stateless NAT-hoz
ip addr add 172.16.0.1 dev nat64
ip route add 172.16.0.0/12 dev nat64
#NAT64 számára hálózat specifikus IPv6 címtartomány
ip addr add 2001:738:2c01:8001::2 dev nat64
ip route add 2001:738:2c01:8001:ffff:ffff:/96 dev nat64
tayga
#Routing engedélyezése
echo 1 > /proc/sys/net/ipv4/ip_forward
```

```
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
#Stateful NAT44 engedélyezése
iptables -t nat -A POSTROUTING -o eth2 -j MASQUERADE
#A mérés során felhasznált célcímek a responder felé elérhetőek
ip route add 10.0.0.0/8 via 193.225.151.70
```

27. ábra /root/nat64-config.sh shell script

A mérés alatt a NAT64 tesztszámítógép “Neighbour table overflow” hibákat adott, a problémát a 28. ábraán látható parancsok kiadásával lehetett megszüntetni. IPv6 esetében a neighbour táblázat segít megtalálni egy IPv6-os host layer 2-es (MAC) címét, míg IPv4 esetén ezt a funkciót az ARP cache látja el.

```
cd /proc/sys/net/ipv6/neigh/default/
echo 4096 > gc_thresh1
echo 8196 > gc_thresh2
echo 16384 > gc_thresh3
```

28. ábra IPv6 neighbour táblázat beállítása

6.2.2. OpenBSD Packet Filter

A NAT64 mérések során a pontos összehasonlítás érdekében dual boot segítségével ugyanazon konfiguráción futott a TAYGA és a PF. A hálózati kártyák beállításai is megegyeztek mindkét esetben. A 29. ábra és 30. ábra tartalmazza az OpenBSD hálózati vezérlőinek beállításait.

```
inet 193.225.151.75 255.255.255.240
inet6 alias 2001:738:2c01:8000::69
!route add -inet default 193.225.151.65
!route add -inet6 default 2001:738:2c01:8000::1
!route add -inet 10.0.0.0/8 193.225.151.70
```

29. ábra /etc/hostname.sk0

```
inet6 alias 2001:738:2c01:8001::1 64
```

30. ábra /etc/hostname.sk1

Az SLAAC alkalmazásához szükséges beállítások megegyeztek a DNS64 mérésnél alkalmazottakkal (13. ábra).

A NAT64 működéséhez szükséges beállítások PF használatakor lényegesen egyszerűbbek, mint a TAYGA esetében. A 31. ábra mutatja ezeket a beállításokat. Az első sor állítja be a NAT alkalmazásához is szükséges state table maximális méretét 40.000 sorra. Kisforgalmú hálózatokban erre nincs szükség, míg nagy forgalmú hálózatok esetében még a 40.000-es értéknél is magasabb beállításra lehet szükség. A mérések során ekkora terheléshez ez az érték bizonyult célravezetőnek. A második sor valósítja meg a tulajdonképpeni NAT64 funkciót, az address family translationt.

```
set limit states 40000
pass in on sk1 inet6 from any to 2001:738:2c01:8001:ffff:ffff::/96 af-to inet from 193.225.151.75
```

31. ábra /etc/pf.conf NAT64 beállítása

Végül a megfelelő működéshez engedélyezni kellett a rtadvd futását, és az IPv4 és IPv6 routingot is (32. ábra és 33. ábra).

```
rtadvd_flags="sk1" # for normal use: list of interfaces
```

32. ábra rtadvd engedélyezése a /etc/rc.conf állományban

```
net.inet.ip.forwarding=1 # 1=Permit forwarding (routing) of IPv4 packets
net.inet6.ip6.forwarding=1 # 1=Permit forwarding (routing) of IPv6 packets
```

33. ábra IPv4 és IPv6 routing engedélyezése a /etc/sysctl.conf állományban

6.2.3. Responder számítógép

A mérés elvégzéséhez szükséges egy olyan számítógép, amely minden beérkező kérésre válaszol. Ezt a válaszoló szerepet a 24. ábra tetején lévő munkaállomás töltötte be. Ez a számítógép egy hálózaton került elhelyezésre a NAT64 teszt konfigurációval, és a teszt során használt 10.0.0.0/8-as IPv4 címek felé irányítása a NAT64 gatewayen statikus routing bejegyzéssel történt (27. ábra és 29. ábra). Ezen a responder számítógépen egy NAT bejegyzéssel minden 10.0.0.0/8-as címtartományba érkező csomag átirányításra (DNAT) került a számítógép publikus címére, így a számítógép minden ezekre a címekre érkező csomagra válaszolt (34. ábra).

```
iptables -t nat -A PREROUTING -d 10.0.0.0/8 -j DNAT --to-destination 193.225.151.70
```

34. ábra iptables beállítás a responder számítógépen

6.2.4. Kliens számítógépek

A mérések során szükséges terhelést produkáló számítógépeken a hálózat specifikus NAT64 címtartomány, a 2001:738:2c01:8001:ffff:ffff::/96 egy minden kliensen kiadott paranccsal került manuális irányításra NAT64 gateway irányába (35. ábra). A well-known NAT64 címtartomány, vagy egy nem a 2001:738:2c01:8001::/64 címtartományba eső hálózat specifikus NAT64 címtartomány alkalmazása esetén erre nincs szükség, hiszen a mérési hálózatban a default gateway és a NAT64 gateway ugyanaz a számítógép.

```
route add -A inet6 2001:738:2c01:8001:ffff:ffff::/96 gw 2001:738:2c01:8001::1
```

35. ábra NAT64 címtartomány útválasztás beállítása a klienseken

6.2.5. NAT64 mérés

A mérési sorozat a DNS64 mérésével megegyező módon 1, 2, 4, majd 8 darab kliens egyidejű futtatásával történt. A 36. ábra tartalmazza a mérések során, a kliens számítógépeken futtatott scriptet.

```
#!/bin/bash
i=`cat /etc/hostname | grep -o .${`
for b in {0..255}
do
  rm -r $b
  mkdir $b
  for c in {0..63..4}
  do
    ping6 -c11 -i0 -q 2001:738:2c01:8001:ffff:ffff:10.$i.$b.$c \
      >> $b/nat64p-10-$i-$b-$c &
    ping6 -c11 -i0 -q 2001:738:2c01:8001:ffff:ffff:10.$i.$b.$((c+1)) \
      >> $b/nat64p-10-$i-$b-$((c+1)) &
    ping6 -c11 -i0 -q 2001:738:2c01:8001:ffff:ffff:10.$i.$b.$((c+2)) \
      >> $b/nat64p-10-$i-$b-$((c+2)) &
    ping6 -c11 -i0 -q 2001:738:2c01:8001:ffff:ffff:10.$i.$b.$((c+3)) \
      >> $b/nat64p-10-$i-$b-$((c+3)) &
  done
done
```

36. ábra ping-test-4.sh teszt shell script

Egy teszt során egy kliens összesen $256 \cdot 64 \cdot 4 \cdot 11 = 720.896$ darab ping csomagot küldött el az összesen $256 \cdot 64 \cdot 4 = 65.536$ IP címre. Egy-egy célcímre összesen 11 darab csomagot. A megfelelően nagy terhelés előállítása érdekében párhuzamosan négy ping parancs futtatása történt a ciklus magjában, majd az utolsó befejezése után folytatódott a ciklus végrehajtása.

6.3. Mérési eredmények

6.3.1. Linux alapú TAYGA

A mérési eredmények összesítését a 10. táblázat tartalmazza.

1	Kliensek száma	1	2	4	8
2	Csomagvesztés [%]	0,01	0,01	0,03	0,03
3	Válaszidő [ms]	0,447	0,957	1,986	4,406
4	szórása	0,103	0,158	0,321	0,474
5	és maximuma	5,202	5,438	8,057	13,965
6	CPU kihasználtsága [%]	69,8	84,3	97,8	100
7	és szórása	3,3	1,7	2,2	0,1
8	Memória foglalása [MB]	10,8	11,4	11,9	12,9
9	Forgalom [csomag/másodperc]	5721	6614	7085	6890

10. táblázat TAYGA NAT64 teljesítmény

A táblázat első sora a terhelést produkáló kliensek számát mutatja. Tehát, hogy egy időben hány kliens hajtotta végre a 36. ábraá19. ábrán bemutatott scriptet. Kétszer annyi kliens kétszeres terhelést produkál. A második sor tartalmazza a produkált csomagvesztések százalékos arányát. A harmadik sor mutatja a mért válaszidőt (RTT), a 4. sor a válaszidők szórását, majd az 5. sor a

legmagasabb mért válaszidőt. A processzor kihasználtságát a 6. sor, és a kihasználtság szórását a 7. sor tartalmazza. A felhasznált memóriát méretét a 8. sor tartalmazza. Az utolsó sorban a másodpercenként elküldött csomagok száma olvasható le.

A mérési eredmények alapján kijelenthető, hogy:

- Minden terhelési szinten rendkívül alacsony értéket produkált a csomagvesztés.
- A terhelés növekedésével (kliensek számának növelésével) nem lép fel nagymértékű teljesítménycsökkenés, a túlterhelés alatt nem omlik össze a rendszer. Hiába éri el a 100%-ot a processzorkihasználtság, a válaszidő megközelítőleg a terhelés növekedésével arányosan nő. ($0,957/0,447=2,14$; $1,986/0,957=2,08$; $4,406/1,986=2,22$)
- A kiszolgált kérések száma folyamatosan nő, addig a pontig, amíg rendelkezésre áll szabad processzorkapacitás. ($6614/5721=115,61\%$; $6614/7085=107,12\%$) A rendszer maximális teljesítőképességét körülbelül négy kliens esetén érte el. A kliensek számának további növelése a teljesítmény növekedése helyett közel 3% ($6890/7085=2,75\%$) csökkenést eredményezett. A tesztelés során használt script nem generált újabb kérdéseket, amíg nem kapott választ az előző kérésekre.
- A memóriefoglalás viszonylag alacsony értéket mutatott, és a terhelés növekedésével csak kis mértékben nőtt.

A teszt egy nagyon fontos eredményt produkált, hiszen a rendszer a sokszoros túlterhelés esetén is megbízhatóan teljesített, nem tett kiszámíthatatlan dolgokat, nem omlott össze, teljesítménye csak nagyon kis mértékben, csökkent. A mért válaszidők a terheléssel arányosan közel lineárisan nőttek (, ezáltal teljesítetve a graceful degradation követelményt). Így a TAYGA biztonsággal használható akár nagy forgalmú, nagy megbízhatóságot kívánó szolgáltatói környezetben is.

6.3.2. OpenBSD Packet Filter

A mérési eredmények összesítését a 11. táblázat 11. táblázat PF NAT64 teljesítmény tartalmazza.

1	Kliensek száma	1	2	4	8
2	Csomagvesztés [%]	0,02	0,02	0,02	0,02
3	Válaszidő [ms]	0,405	0,486	0,606	1,194
4	szórása	0,050	0,073	0,127	0,250
5	és maximuma	1,770	1,433	3,904	7,055
6	CPU kihasználtsága [%]	31,7	50,1	80,1	90,3
7	és szórása	5,4	5,0	5,1	4,7
8	Memória foglalása [MB]	2,4	3,5	5,5	7,9
9	Forgalom [csomag/másodperc]	5909	11091	18367	22886

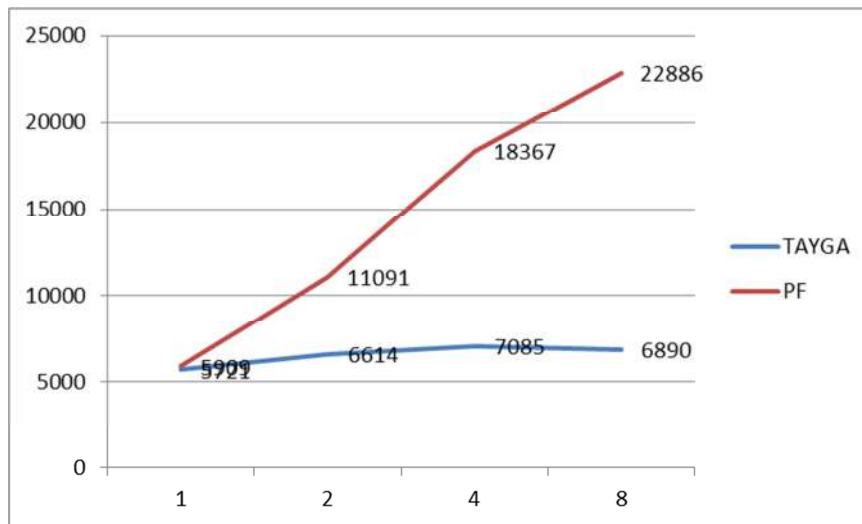
11. táblázat PF NAT64 teljesítmény

A mérési eredmények alapján kijelenthető, hogy:

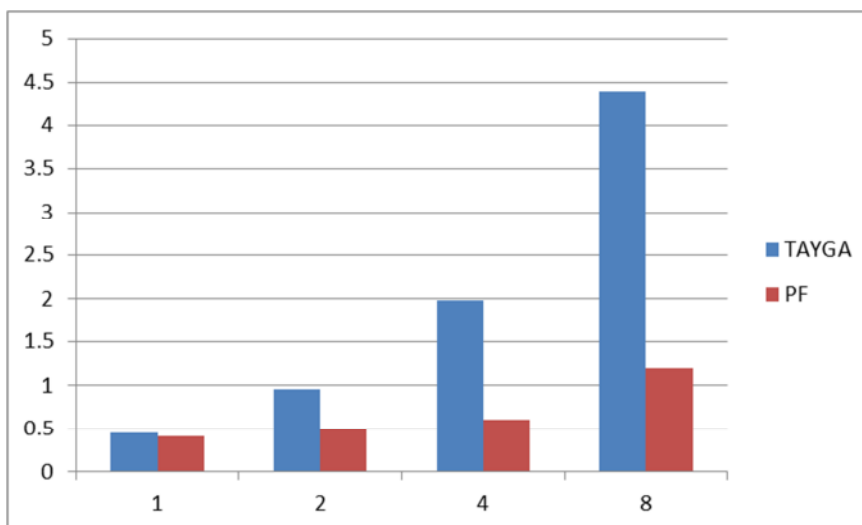
- Minden terhelési szinten rendkívül alacsony értéket produkált a csomagvesztés.
- A terhelés növekedésével a válaszidő közel 20%-al nő, amíg a processzor kihasználtsága nem közelíti meg a 90%-ot ($0,486/0,405=1,2$; $0,606/0,486=1,25$) 4-ről 8 darab kliensre viszont már itt is közel duplázódik a válaszidő ($1,194/0,606=1,97$).
- A kiszolgált kérések száma nagymértékben nő ($11091/5909=187,7\%$; $18367/11091=165,6\%$; $22886/18367=124,6\%$) a terhelés növelésével.
- A mérés során produkált maximális terhelés mellett is csak 90%-os a processzorkihasználtság, így valószínűleg a 8 kliensnél mért másodpercenként 22.886 darabnál is több csomag továbbítására képes a rendszer.
- A memóriefoglalás alacsony értéket mutatott, és a terhelés növekedésével csak kis mértékben nőtt.

6.4. Értékelés

A két mérési sorozat eredményeit összehasonlítva egy kliens esetén a TAYGA és a PF megközelítőleg azonos eredményt produkál, azonban a TAYGA ezt több mint kétszeres processzoridő felhasználásával éri el. A terhelést tovább növelve a PF behozhatatlan előnyt produkálva 8 kliensnél több, mint háromszoros teljesítményével ($22886/6890=332,16\%$), és közel negyedakkora ($1,194/4,406=27,01\%$) válaszidejével egyértelmű választás (37. ábra és 38. ábra).



37. ábra Másodpercenként továbbított csomagok száma



38. ábra Válaszidő [ms] a kliensek számának függvényében

7. Összefoglalás

A dolgozat elkészítése során megvizsgáltam az összes elterjedtebb IPv6 áttérési technikát. Mindegyik esetében részletesen, a gyakorlatban is megvizsgáltam a lehetséges alkalmazási lehetőségeket és áttanulmányoztam a kapcsolódó szakirodalmat. Az ismertetett technikák közül egyértelműen a DNS64/NAT64 használható fel a legsokoldalúbban az áttérés jelenlegi fázisában. Így az elérhető és szabadon terjeszthető DNS64 és NAT64 implementációk kiválasztásával folytattam a munkát. Sajnos ma még nem érhető el sok működő DNS64/NAT64 megvalósítás annak ellenére, hogy az IPv6 protokoll bevezetése már nem halasztható tovább. A keresés során jutottam el a BIND9 és a TOTD DNS64 szolgáltatást nyújtó alkalmazásokhoz, valamint a TAYGA és az OpenBSD Packet Filter NAT64 szolgáltatásához. A vizsgálatok elvégzéséhez kiválasztottam az elterjedtebb, szabad szoftverként hozzáférhető operációs rendszereket. Így került a legnépszerűbb Linux Debian disztribúciója, az OpenBSD és a FreeBSD legfrissebb verziója a mérés során használt operációs rendszerek közé. A mérések során megbizonyosodtam a különféle operációs rendszerek, valamint DNS64/NAT64 implementációk kombinációinak megbízhatóságáról, teljesítményéről, és a terheléses vizsgálatok során fény derült arra is, hogy jelentős túlterhelés mellett hogyan viselkednek az egyes kombinációk. A mérések alapján legmegbízhatóbb, és leggyorsabb DNS64 szerver a Linuxon futtatott BIND, abban az esetben, ha teljes értékű DNS szerverre van szükségünk és nem elégszünk meg egy forwarderrel. Ha elég a forwarderként működő DNS64 szerver, és korlátozott a szerverre jutó terhelés, úgy sebessége miatt a TOTD az ideális választás. Egyszerűsége és alacsony memóriaigénye miatt a TOTD tökéletes megoldás embedded rendszerekbe is.

A NAT64 alatt mutatott teljesítményével az OpenBSD Packet Filtere a legjobb döntés NAT64 átjáró üzemeltetésére. Sajnos a BIND OpenBSD alatt nem hozott kielégítő eredményt, így egy OpenBSD-n működő, kombinált DNS64/NAT64 implementáció nem nevezhető jó döntésnek. Ebben az esetben a TAYGA jelentős teljesítménybeli hátránya ellenére, a BIND robusztusságának köszönhetően a Linux alapú TAYGA+BIND a legmegbízhatóbb megoldás. Ha nagyméretű hálózathoz keressük az ideális megoldást, ott a router és a DNS szerver funkció és a két féle eszköz üzemeltetése is kettéválik. Ilyen esetekben érdemes a Linux+TAYGA DNS64, és az OpenBSD PF megoldását kombinálni. Természetesen mindegyik esetben figyelembe kell venni a már meglévő hálózati eszközöket és topológiát, hiszen a meglévő eszközök közé kell beintegrálni az új szolgáltatásokat.

8. Irodalomjegyzék

- [1] S. E. Deering és R. M. Hinden, „RFC 2460 - Internet Protocol, Version 6 (IPv6) Specification,” 12 1998. [Online]. Available: <http://www.ietf.org/rfc/rfc2460.txt>. [Hozzáférés dátuma: 28 10 2012].
- [2] G. Huston, „IPv4 Address Report,” 2 11 2012. [Online]. Available: <http://www.potaroo.net/tools/ipv4/index.html>. [Hozzáférés dátuma: 3 11 2012].
- [3] RIPE NCC, „RIPE NCC Begins to Allocate IPv4 Address Space From the Last /8,” 14 9 2012. [Online]. Available: <http://www.ripe.net/internet-coordination/news/ripe-ncc-begins-to-allocate-ipv4-address-space-from-the-last-8>. [Hozzáférés dátuma: 7 11 2012].
- [4] M. Bagnulo, A. Sullivan, P. Matthews és I. v. Beijnum, „RFC 6147 - DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers,” 4 2011. [Online]. Available: <http://www.ietf.org/rfc/rfc6147.txt>. [Hozzáférés dátuma: 28 10 2012].
- [5] M. Bagnulo, P. Matthews és I. v. Beijnum, „RFC 6146 - Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers,” 4 2011. [Online]. Available: <http://www.ietf.org/rfc/rfc6146.txt>. [Hozzáférés dátuma: 28 10 2012].
- [6] C. Bao, C. Huitema, M. Bagnulo, M. Boucadair és X. Li, „RFC 6052 - IPv6 Addressing of IPv4/IPv6 Translators,” 10 2010. [Online]. Available: <http://www.ietf.org/rfc/rfc6052.txt>. [Hozzáférés dátuma: 28 10 2012].
- [7] M. Bagnulo és A. García-Martínez, „The NAT64/DNS64 Tool Suite for IPv6 Transition,” *IEEE Communications Magazine*, pp. július 177-183., 2012.
- [8] R. Tuminauskas, „Experimental NAT64/DNS64 Service,” 1 6 2011. [Online]. Available: <https://labs.ripe.net/Members/raimis/experimental-nat64-dns64-service>. [Hozzáférés dátuma: 28 10 2012].
- [9] IANA, „IPv6 Global Unicast Address Assignments,” 1 10 2012. [Online]. Available: <http://www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xml>. [Hozzáférés dátuma: 28 10 2012].
- [10] IANA, „IANA IPv4 Address Space Registry,” 15 10 2012. [Online]. Available:

<http://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xml>. [Hozzáférés dátuma: 28 10 2012].

- [11] A. Durand, R. Droms, J. Woodyatt és Y. L. Lee, „RFC 6333 - Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion,” 8 2011. [Online]. Available: <http://www.rfc-editor.org/rfc/rfc6333.txt>. [Hozzáférés dátuma: 7 11 2012].
- [12] B. E. Carpenter és K. Moore, „RFC 3056 - Connection of IPv6 Domains via IPv4 Clouds,” 2 2001. [Online]. Available: <http://www.ietf.org/rfc/rfc3056.txt>. [Hozzáférés dátuma: 24 4 2012].
- [13] Microsoft Corporation, Windows Server 2008 IPv6 Transition Technologies, 2008.
- [14] C. Huitema, „RFC 3068 - An Anycast Prefix for 6to4 Relay Routers,” 6 2001. [Online]. Available: <http://www.ietf.org/rfc/rfc3068.txt>. [Hozzáférés dátuma: 24 4 2012].
- [15] R. Despres, „RFC 5569 - IPv6 Rapid Deployment on IPv4 Infrastructures (6rd),” 1 2010. [Online]. Available: <http://www.ietf.org/rfc/rfc5569.txt>. [Hozzáférés dátuma: 28 10 2012].
- [16] C. Huitema, „RFC 4380 - Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs),” 2 2006. [Online]. Available: <http://www.ietf.org/rfc/rfc4380.txt>. [Hozzáférés dátuma: 28 10 2012].
- [17] D. Thaler, „RFC 6081 - Teredo Extensions,” 1 2011. [Online]. Available: <http://www.ietf.org/rfc/rfc6081.txt>. [Hozzáférés dátuma: 28 10 2012].
- [18] R. Denis-Courmont, „Miredo: Teredo IPv6 tunneling for Linux and BSD,” 3 3 2012. [Online]. Available: <http://www.remlab.net/miredo/>. [Hozzáférés dátuma: 28 10 2012].
- [19] F. L. Templin, T. Gleeson és D. Thaler, „RFC 5214 - Intra-Site Automatic Tunnel Addressing Protocol (ISATAP),” 3 2008. [Online]. Available: <http://www.ietf.org/rfc/rfc5214.txt>. [Hozzáférés dátuma: 16 4 2012].
- [20] F. L. Templin, T. Gleeson, M. Talwar és D. Thaler, „RFC 4214 - Intra-Site Automatic Tunnel Addressing Protocol (ISATAP),” 10 2005. [Online]. Available: <http://www.ietf.org/rfc/rfc4214.txt>. [Hozzáférés dátuma: 16 4 2012].
- [21] „Intra-Site Automatic Tunnel Addressing Protocol (ISATAP),” 2012. [Online]. Available: <http://isatap.org/>. [Hozzáférés dátuma: 16 4 2012].
- [22] S. Hlusiak, „Setting up an ISATAP router with Linux,” 2010. [Online]. Available: <http://www.saschahlusiak.de/linux/isatap.htm>. [Hozzáférés dátuma: 28 10 2012].

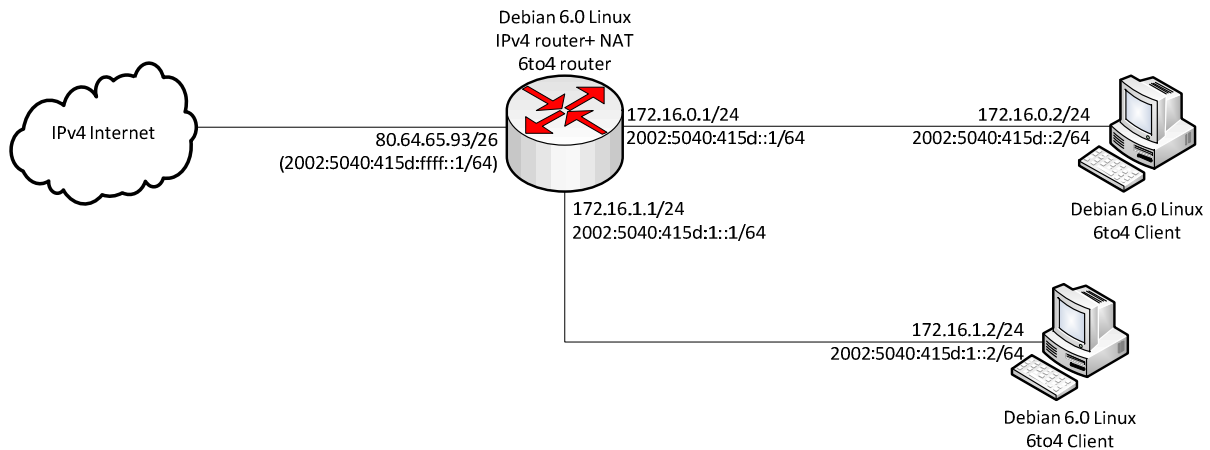
- [23] L. Xing, C. Bao és F. Baker, „RFC 6145 - IP/ICMP Translation Algorithm,” 4 2011. [Online]. Available: <http://www.ietf.org/rfc/rfc6145.txt>. [Hozzáférés dátuma: 28 10 2012].
- [24] G. Tsirtsis és P. Srisuresh, „RFC 2766 - Network Address Translation - Protocol Translation (NAT-PT),” 2 2000. [Online]. Available: <http://www.ietf.org/rfc/rfc2766.txt>. [Hozzáférés dátuma: 28 10 2012].
- [25] C. Aoun és E. B. Davies, „RFC 4966 - Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status,” 7 2007. [Online]. Available: <http://www.ietf.org/rfc/rfc4966.txt>. [Hozzáférés dátuma: 28 10 2012].
- [26] L. Tomicki, „NAT-PT - Download,” 4 12 2010. [Online]. Available: <http://tomicki.net/naptd.download.php>. [Hozzáférés dátuma: 28 10 2012].
- [27] „Squid: Optimising Web Delivery,” 29 8 2012. [Online]. Available: <http://www.squid-cache.org/>. [Hozzáférés dátuma: 28 10 2012].
- [28] J. O. L. Kenneth és E. S. Y. William, „Performance of NAT64 versus NAT44 in the Context of IPv6 Migration,” *Proceedings of the International MultiConference of Engineers and Computer Scientists*, pp. 638-645, 2012.
- [29] C. Pérez Monte, M. Inés Robles, G. Mercado és C. Taffernaberry, „Implementation and Evaluation of Protocols Translating Methods for IPv4 to IPv6 Transition,” *Journal of Computer Science & Technology*, %1. kötet12, %1. szám2, pp. 64-70, 2012.
- [30] S.-y. Yu és B. E. Carpenter, „Measuring IPv4 – IPv6 translation technique,” Auckland, 2012.
- [31] G. Lencse és G. Takács, „Performance Analysis of DNS64 and NAT64 Solutions,” *Infocommunications Journal*, pp. 29-36., 6 2012.
- [32] Viagénie, „Ecdysis: open-source implementation of a NAT64 gateway,” 17 11 2010. [Online]. Available: <http://ecdysis.viagenie.ca/download.html>. [Hozzáférés dátuma: 30 10 2012].
- [33] N. Lutchansky, „TAYGA Simple, no-fuss NAT64 for Linux,” 11 6 2011. [Online]. Available: <http://www.litech.org/tayga/>. [Hozzáférés dátuma: 30 10 2012].
- [34] T. d. Raadt, „The OpenBSD 5.1 Release,” 1 5 2012. [Online]. Available: <http://www.openbsd.org/51.html>. [Hozzáférés dátuma: 30 10 2012].
- [35] S. Perreault, „[Ecdysis-discuss] NAT64 in OpenBSD,” 16 10 2011. [Online]. Available: <http://www.viagenie.ca/pipermail/ecdiscuss/2011-October/000173.html>. [Hozzáférés

dátuma: 30 10 2012].

- [36] P. N. Hansteen, The Book of PF, 2nd Edition - A No-Nonsense Guide to the OpenBSD Firewall, 2nd szerk., San Francisco: William Pollock, 2010.
- [37] Internet Systems Consortium, „BIND,” 9 10 2012. [Online]. Available: <http://www.isc.org/software/bind>. [Hozzáférés dátuma: 30 10 2012].
- [38] F. W. Dillema, „The totd ('Trick Or Treat Daemon'),” 25 3 2009. [Online]. Available: <http://www.dillema.net/software/totd.html>. [Hozzáférés dátuma: 30 10 2012].
- [39] J. Haas, „Linux / Unix Command: host,” 2012. [Online]. Available: http://linux.about.com/library/cmd/blcmdl1_host.htm. [Hozzáférés dátuma: 7 11 2012].
- [40] jonas, „time,” 11 12 1999. [Online]. Available: <http://www.gnu.org/software/time/>. [Hozzáférés dátuma: 7 11 2012].
- [41] „FreeBSD Handbook Chapter 16 Jails,” [Online]. Available: http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/jails.html. [Hozzáférés dátuma: 2 11 2012].
- [42] Debian, „Debian IPv6 Project,” 11 4 2012. [Online]. Available: <http://wiki.debian.org/DebianIPv6>. [Hozzáférés dátuma: 30 4 2012].
- [43] D. Shinder, „Configuring an ISATAP Router with Windows Server 2008 R2,” 7 4 2011. [Online]. Available: http://www.windowsnetworking.com/articles_tutorials/configuring-isatap-router-windows-server-2008-r2-part1.html. [Hozzáférés dátuma: 19 4 2012].
- [44] fdz, „Setup of an ISATAP router on Linux,” 19 2 2010. [Online]. Available: http://ipv6-portal.com/index.php?option=com_content&view=article&id=13%3Asetup-of-isatap-router-on-linux&catid=13%3Ahowto&Itemid=4&lang=en. [Hozzáférés dátuma: 23 4 2012].
- [45] cjj, „Setup isatap router on debian,” 11 10 2010. [Online]. Available: <http://chunchung.ccdw.org/2010/10/setup-isatap-router-on-debian.html>. [Hozzáférés dátuma: 23 4 2012].
- [46] U. Möhring, „IPv6 for web hosts,” 1 3 2011. [Online]. Available: <http://scratchpad.ulfmoehring.net/articles/ipv6-for-webhosts>. [Hozzáférés dátuma: 23 4 2012].

9. Függelék

9.1. IPv6 hálózat kialakítása Linux NAT44 és 6to4 router segítségével



39. ábra Debian Internet átjáró 6to4-gyel

Tipikus konfigurációként sűrűn előfordul, hogy az Internet szolgáltató még nem nyújt IPv6-os szolgáltatást, a szolgáltatóhoz csatlakozó router pedig címfordítást használ (39. ábra). Ha a routert megfelelően beállítjuk, akkor ebben az esetben is segíthet a 6to4. Egy nagyobb IPv6 route-olt hálózat is kialakítható statikus, vagy radvd alkalmazásával akár automatikus konfigurációval (62. ábra). Erre a lehetőséget a 6to4 kidolgozásakor előrelátóan betervezett 16 bit hosszúságú subnetID adja (3. ábra). A hálózat kialakításakor az Internetre csatlakozó routeren 6to4 tunnelt, a belső oldalon natív IPv6 hálózatot kell kialakítani. Az IPv6 címkiosztásnál pedig a rendelkezésre álló 65536 subnetből lehet választani. Természetesen akár dinamikus IP címkiosztású PPPOE kapcsolat is használható, ekkor viszont nem kerülhető el a radvd alkalmazása, ráadásul az új csatlakozáskor kapott új IP cím miatt az összes számítógép IPv6-os címe is megváltozik.

A már működő IPv4-es routerből kiindulva első lépésként engedélyezni kell a `/etc/sysctl.conf`-ban az IPv6 routingot (56. ábra).

Ezután be kell állítani a `/etc/network/interfaces` állományban a 6to4 interfészt a hozzá tartozó paraméterekkel és a lokális hálózati interfészekre az IPv6-os címeket (40. ábra). Ezután a klienseken a már csak a kiválasztott IPv6-os címeket kell beállítani a `/etc/network/interfaces` állományban (41. ábra).

```
# The primary network interface
allow-hotplug eth0 eth1 eth2
iface eth0 inet static
    address 80.64.65.93
    netmask 255.255.255.192
    gateway 80.64.65.65
    dns-nameservers 80.64.65.69 80.64.65.66
auto sit0
iface sit0 inet6 static
    address 2002:5040:415d:ffff::1
    netmask 16
    gateway ::192.88.99.1
iface eth1 inet static
    address 172.16.0.1
    netmask 255.255.255.0
iface eth1 inet6 static
    address 2002:5040:415d::1
    netmask 64
iface eth2 inet static
    address 172.16.1.1
    netmask 255.255.255.0
iface eth2 inet6 static
    address 2002:5040:415d:1::1
    netmask 64
```

40. ábra 6to4 router beállítása NAT-olt hálózathoz

```
# The primary network interface
allow-hotplug eth0
iface eth0 inet static
    address 172.16.0.2
    netmask 255.255.255.0
    gateway 172.16.0.1
iface eth0 inet6 static
    address 2002:5040:415d::2
    netmask 64
    gateway 2002:5040:415d::1
```

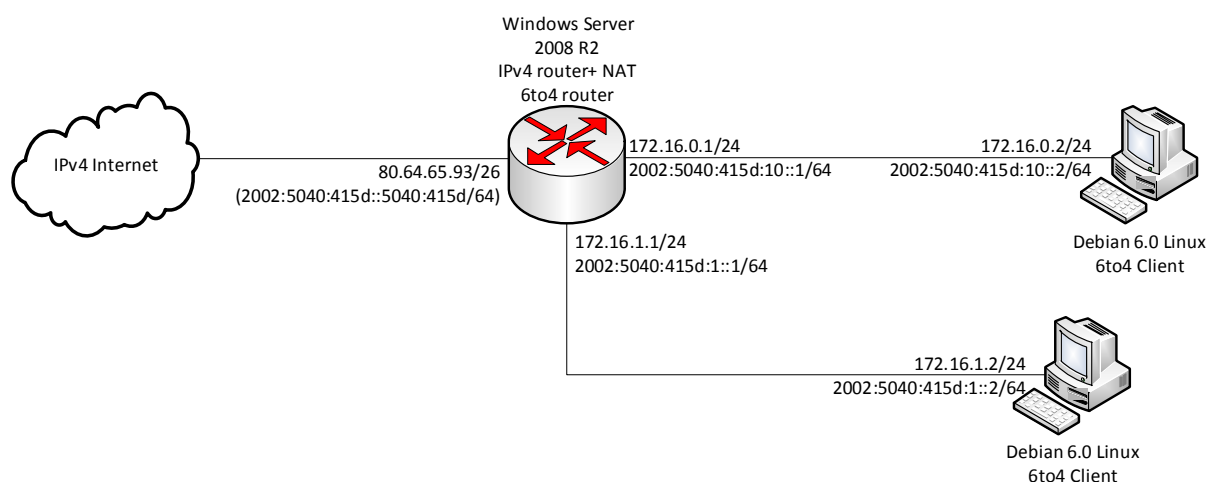
41. ábra IPv6 kliens beállítása

A kliensről a `traceroute6` parancs segítségével ellenőrizhető a beállítások működése (42. ábra).

```
root@DebianIPv6kliens-1:~# traceroute6 2a02:a50::66
traceroute to 2a02:a50::66 (2a02:a50::66), 30 hops max, 80 byte packets
 1  2002:5040:415d::1 (2002:5040:415d::1)  0.824 ms  0.803 ms  0.791 ms
 2  2002:c058:6301::1 (2002:c058:6301::1)  1.648 ms  1.638 ms  1.631 ms
 3  2a02:a50::66 (2a02:a50::66)  2.262 ms  2.256 ms  2.248 ms
root@DebianIPv6kliens-1:~#
```

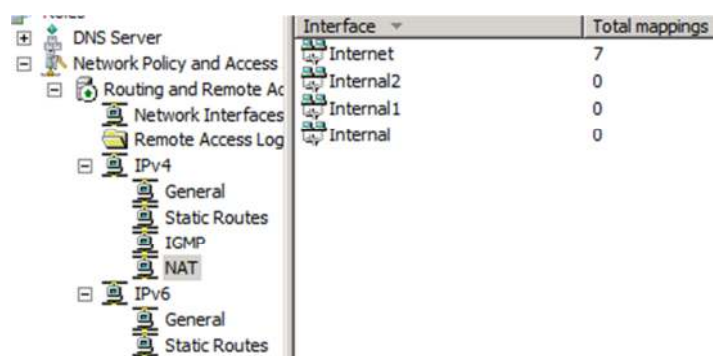
42. ábra `traceroute6` a kliensről

9.2. IPv6 hálózat kialakítása Windows Server 2008 R2 NAT és 6to4 router segítségével



43. ábra Windows 2008 Internet átjáró 6to4-gyel

Sok vállalkozás nem Linux alapon oldja meg telephelyének Internet csatlakozását, hanem Windows Server 2008-at használ erre a célra. Itt is egy már működő IPv4-es NAT-olt hálózathoz indultam ki. Így a Windows Server 2008 R2 konfigurálását nem ismertetem részletesen. Az Internet megosztáshoz jelenleg engedélyezve van a NAT. Valamint DNS és DHCP kiszolgálók működnek még a szerveren. A RRAS NAT beállítása az ábrán látható (44. ábra).



44. ábra RRAS NAT interfészek

A szerver az Internet szolgáltatótól jelen esetben csak egy nyilvános fix IPv4 címet kap, amiből generál magának egy a 6to4 protokollhoz megfelelő IPv6-os címet. Figyelni kell arra, hogy a generált 6to4 IPv6-os cím ne essen egy hálózathoz egyik alhálózattal sem. Ezért nem kerülhetett egyik kliens sem a 2002:5040:415d::/64-es tartományba, hiszen a Windows a 2002:5040:415d::5040:415d címet osztja ki magának. A hálózati szolgáltatás alapértelmezett beállítása miatt, alap esetben nem jelenik meg 6to4 interface. (Az ipconfig parancs segítségével ellenőrizhető.) A netsh interface 6to4 show state paranccsal lekérdezhető az érvényben lévő beállítás (45. ábra).

```
C:\Users\Administrator>netsh interface 6to4 show state
6to4 Service State : default
Undo on Service Stop : default

C:\Users\Administrator>
```

45. ábra 6to4 Interface állapotának lekérdezése

A 6to4 engedélyezése a netsh interface 6to4 set state enabled paranccsal végezhető el (46. ábra). Ezután már az ipconfig is megjeleníti a 6to4 interfészt (47. ábra).

```
C:\Users\Administrator>netsh interface 6to4 set state enabled
Ok.

C:\Users\Administrator>netsh interface 6to4 show state
6to4 Service State : enabled
Undo on Service Stop : default
```

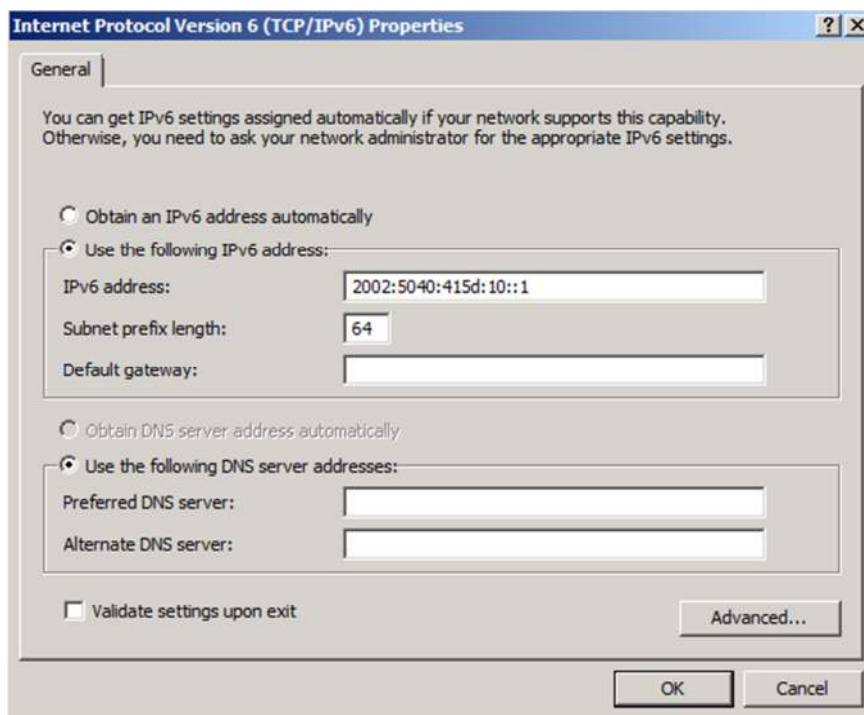
46. ábra 6to4 engedélyezése Windows Server 2008 R2 rendszeren

```
Tunnel adapter 6T04 Adapter:

Connection-specific DNS Suffix . : 
Description . . . . . : Microsoft 6to4 Adapter
Physical Address. . . . . : 00-00-00-00-00-00-E0
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes
IPv6 Address. . . . . : 2002:5040:415d::5040:415d<Preferred>
Default Gateway . . . . . : 2002:c058:6301::c058:6301
DNS Servers . . . . . : 127.0.0.1
NetBIOS over Tcpip. . . . . : Disabled
```

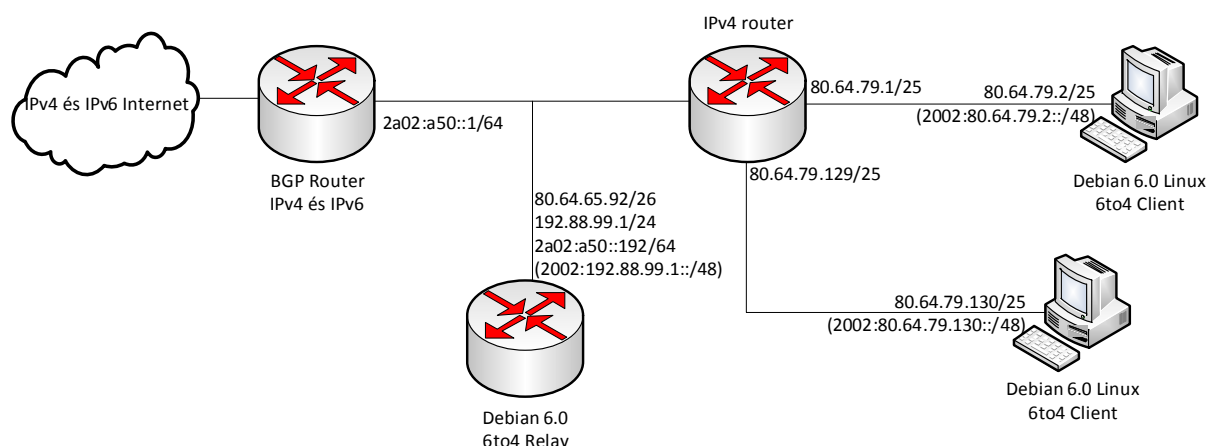
47. ábra 6to4 interface

Ezután a belső hálózati interfészeken be kell állítani az IPv6-os címeket, és már használható is a hálózat (48. ábra).



48. ábra A 172.16.0.1/24-es hálózathoz tartozó IPv6-os cím beállítása

9.3. 6to4 hálózat kialakítása Linux 6to4 relay alkalmazásával



49. ábra Debian 6to4 relayjel megvalósított hálózat

Elsőként a két kliens beállítását érdemes elvégezni. Itt két lehetőségünk van. Az első a már érvénytelenített sit (Simple Internet Transition) tunnel segítségével, a másik a v4tunnel segítségével [42]. A 49. ábraán felül található kliensen a 6to4, míg az alsón a sit tunnel segítségével alakítottam ki a hálózatot. A sit megoldás hátránya, hogy nem lehet tunnelenként külön tűzfalszabályokat létrehozni az ip6tables segítségével. A tunnel létrehozása permanens módon, legegyszerűbben a `/etc/network/interfaces` állományban tehető meg, ahogy az 50. ábraán látható. Fontos, hogy az alhálózati maszk 16 bit hosszúságú legyen, mert a 6to4 hostok csak így érik el egymást. A `gateway` paraméterrel adhatjuk meg a 6to4 relay címét. E cím nélkül is lehetőség van a 6to4 hostok közötti kommunikációra, azonban a nem 6to4 hostok elérése nem fog működni. A példa konfigurációban az RFC 6038-ban 6to4 relay céljára rögzített 192.88.99.1-es anycast címet használtam, így saját 6to4 relay beüzemelése nélkül is lehetőség van a konfiguráció ellenőrzésére.

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
    address 80.64.79.2
    netmask 255.255.255.128
    gateway 80.64.79.1

auto tun6to4
iface tun6to4 inet6 v4tunnel
    address 2002:5040:4f02::1
    netmask 16
    gateway ::192.88.99.1
    endpoint any
    local 80.64.79.2
```

50. ábra `/etc/network/interfaces` v4 tunnellel

A másik kliensen, a sit kialakításához szintén a host IPv6-os címe, a netmask és a relay címe paraméterek szükségesek (51. ábra).

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
    address 80.64.79.130
    netmask 255.255.255.128
    gateway 80.64.79.129
auto sit0
iface sit0 inet6 static
    address 2002:5040:4f82::1
    netmask 16
    gateway ::192.88.99.1
```

51. ábra /etc/network/interfaces sit tunellel

A két kliens újraindítása után ifconfig parancssal már kiíratható az új tunnel interface. Az ifconfig kimenete látható a két ábrán (52. ábra és 53. ábra).

```
tun6to4  Link encap:IPv6-in-IPv4
         inet6 addr:  ::80.64.79.2/128 Scope:Compat
         inet6 addr:  2002:5040:4f02::1/16 Scope:Global
         UP RUNNING NOARP MTU:1480 Metric:1
         RX packets:20 errors:0 dropped:0 overruns:0 frame:0
         TX packets:20 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:2080 (2.0 KiB)  TX bytes:2080 (2.0 KiB)
```

52. ábra Új 6to4 tunnel

```
sit0    Link encap:IPv6-in-IPv4
        inet6 addr:  ::127.0.0.1/96 Scope:Unknown
        inet6 addr:  ::80.64.79.130/96 Scope:Compat
        inet6 addr:  2002:5040:4f82::1/16 Scope:Global
        UP RUNNING NOARP MTU:1480 Metric:1
        RX packets:5 errors:0 dropped:0 overruns:0 frame:0
        TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:520 (520.0 B)  TX bytes:624 (624.0 B)
```

53. ábra Az elavult sit tunnel

A ping6 segítségével ellenőrizve, jól látható, hogy a két számítógép közötti IPv6 kommunikáció működik (54. ábra). Ugyanígy ellenőrizhető a más IPv6-os, nem 6to4 számítógépek elérhetősége is (55. ábra). A második esetben jól megfigyelhető, hogy a saját hálózatban lévő host válaszideje 42.1ms, ami a relayként használt host nagy távolsága miatt van (5. ábra).

```

sit0      Link encap:IPv6-in-IPv4
          inet6 addr:  ::127.0.0.1/96 Scope:Unknown
          inet6 addr:  ::80.64.79.130/96 Scope:Compat
          inet6 addr:  2002:5040:4f82::1/16 Scope:Global
          UP RUNNING NOARP  MTU:1480  Metric:1
          RX packets:5 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:520 (520.0 B)  TX bytes:624 (624.0 B)

root@DebianIPv6kliens-2:~# ping6 2002:5040:4f02::1
PING 2002:5040:4f02::1(2002:5040:4f02::1) 56 data bytes
64 bytes from 2002:5040:4f02::1: icmp_seq=1 ttl=64 time=0.891 ms
64 bytes from 2002:5040:4f02::1: icmp_seq=2 ttl=64 time=0.803 ms
64 bytes from 2002:5040:4f02::1: icmp_seq=3 ttl=64 time=0.995 ms

```

54. ábra Kommunikáció ellenőrzése a második kliensről

```

root@DebianIPv6kliens-2:~# ping6 ripe.net
PING ripe.net(www.ipv6.ripe.net) 56 data bytes
64 bytes from www.ipv6.ripe.net: icmp_seq=1 ttl=52 time=31.5 ms
64 bytes from www.ipv6.ripe.net: icmp_seq=2 ttl=52 time=32.6 ms
^C
--- ripe.net ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 31.561/32.129/32.698/0.596 ms
root@DebianIPv6kliens-2:~# ping6 2a02:a50::1
PING 2a02:a50::1(2a02:a50::1) 56 data bytes
64 bytes from 2a02:a50::1: icmp_seq=1 ttl=55 time=42.1 ms
64 bytes from 2a02:a50::1: icmp_seq=2 ttl=55 time=42.1 ms

```

55. ábra Natív IPv6-os host kapcsolat ellenőrzése

A saját 6to4 relay üzembe helyezéséhez egy már működő natív IPv6-os csatlakozással rendelkező Debian 6.0 konfigurációjának módosítására volt szükség. Első lépésben az IPv6 routing engedélyezése szükséges a `/etc/sysctl.conf` állományban (56. ábra).

```

# Uncomment the next line to enable packet forwarding for IPv6
# Enabling this option disables Stateless Address Autoconfiguration
# based on Router Advertisements for this host
net.ipv6.conf.all.forwarding=1

```

56. ábra IPv6 forwarding engedélyezése

Ezután a 192.88.99.1 anycast IPv4-es címet másodlagosként (alias) hozzáadjuk az Ethernet interfészhez (eth0:1), így a relayt a 6to4 kliensek elérhetik az anycast címen is. Végül létre kell hozni a sit interfészt is, ami a 6to4 kliensek tunneleit végződtetni fogja. A teljes `/etc/network/interfaces` konfigurációs állomány a módosításokkal, az ábrán látható (57. ábra). A sit0 tunnel konfigurációjában, a kliensétől eltérően nincs szükség a gateway megadására, hiszen az átjáró saját maga, IPv6 címként pedig a 192.88.99.1-es 6to4 relay unicast címnek megfelelő 6to4 címet kell beállítani (2002:c058:6301::1).

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0 eth1
iface eth0 inet static
    address 80.64.65.92
    netmask 255.255.255.192
    gateway 80.64.65.65
auto eth0:1
iface eth0:1 inet static
    address 192.88.99.1
    netmask 255.255.255.0
iface eth0 inet6 static
    address 2a02:a50::192
    netmask 64
    gateway 2a02:a50::1
auto sit0
iface sit0 inet6 static
    address 2002:c058:6301::1
    netmask 16
```

57. ábra Debian 6to4 relay /etc/network/interfaces beállításai

A hálózati alrendszer újraindítása után a Linux beolvassa az új konfigurációt, és ha a hálózat routerének routing táblájában is beállításra kerül, hogy a 192.88.99.0/24-es anycast tartomány már ezen az új átjárón található, akkor ki is próbálható a működés (58. ábra).

```
root@DebianIPv6kliens-2:~# ping6 ripe.net
PING ripe.net(www.ipv6.ripe.net) 56 data bytes
64 bytes from www.ipv6.ripe.net: icmp_seq=1 ttl=44 time=25.8 ms
64 bytes from www.ipv6.ripe.net: icmp_seq=2 ttl=44 time=25.7 ms
^C
--- ripe.net ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1006ms
rtt min/avg/max/mdev = 25.723/25.791/25.859/0.068 ms
root@DebianIPv6kliens-2:~# ping6 2a02:a50::1
PING 2a02:a50::1(2a02:a50::1) 56 data bytes
64 bytes from 2a02:a50::1: icmp_seq=1 ttl=55 time=19.8 ms
64 bytes from 2a02:a50::1: icmp_seq=2 ttl=55 time=19.3 ms
```

58. ábra Natív IPv6-os host kapcsolat saját 6to4 relay segítségével

Ha a hálózatban a 2002::/16-os tartományt is az új 6to4 relayre rányítjuk, akkor a válaszidő még tovább rövidül, azonban az is jól megfigyelhető, hogy ha egy host (jelen esetben a ripe.net) nem ezt az új 6to4 relayt használja a válaszadásra, akkor a válaszidő nem változik (59. ábra). Ennek oka az említett anycast technológia, hiszen egy másik hálózat anycast (BGP) hirdetéssel bármelyik nyilvános 6to4 relayt választhatja, arra befolyással nem bírunk. Ezek alapján a legjobb megoldás az lenne, ha minden natív IPv6-os hálózatot használó ISP működtetne ügyfelei részére 6to4 relayt, amit azonban nem szükséges BGP segítségével meghirdetve, bárki által elérhető nyilvános relayként működtetnie. Így csökkentheti hálózata és az Internet terhelését, növelheti hálózata biztonságát, csökkentheti saját és ügyfelei függését a nyilvános relay kiszolgálóktól.

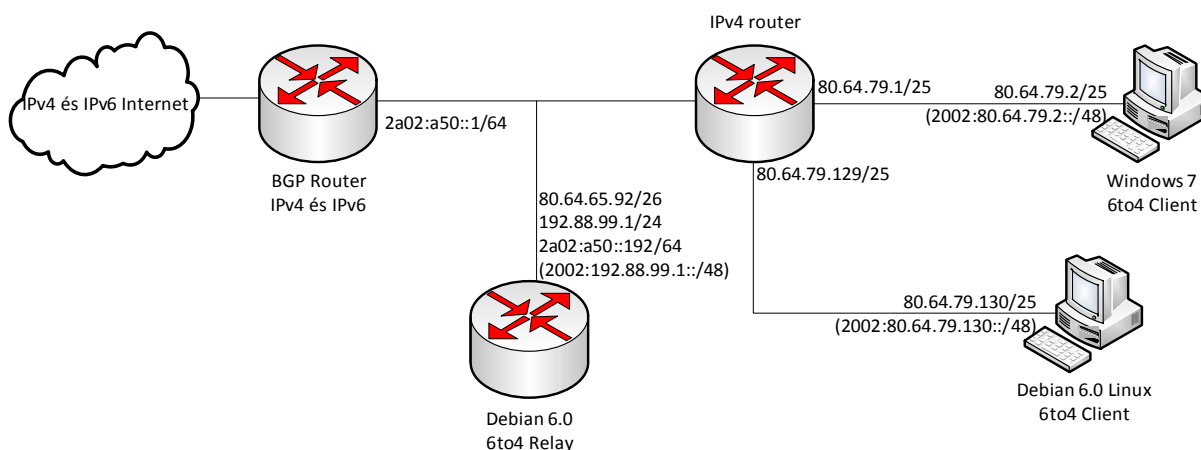
```

root@DebianIPv6kliens-2:~# ping6 ripe.net
PING ripe.net(www.ipv6.ripe.net) 56 data bytes
64 bytes from www.ipv6.ripe.net: icmp_seq=1 ttl=44 time=25.8 ms
64 bytes from www.ipv6.ripe.net: icmp_seq=2 ttl=44 time=25.3 ms
^C
--- ripe.net ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1005ms
rtt min/avg/max/mdev = 25.353/25.588/25.824/0.284 ms
root@DebianIPv6kliens-2:~# ping6 2a02:a50::1
PING 2a02:a50::1(2a02:a50::1) 56 data bytes
64 bytes from 2a02:a50::1: icmp_seq=1 ttl=63 time=1.37 ms
64 bytes from 2a02:a50::1: icmp_seq=2 ttl=63 time=1.08 ms

```

59. ábra Natív IPv6-os host kapcsolat saját 6to4 relay segítségével

9.4. 6to4 hálózat kialakítása Linux 6to4 Windows 7 klienssel



60. ábra Windows 7 kliens 6to4 hálózat

Windows 7 kliens alkalmazása esetén nincs szükség semmilyen konfiguráció módosítására a számítógépen (60. ábra). A Windows 7 automatikusan megtalálja a 6to4 relayt és elvégzi a szükséges beállításokat (61. ábra).

```

Tunnel adapter 6T04 Adapter:

Connection-specific DNS Suffix . : 
Description . . . . . : Microsoft 6to4 Adapter
Physical Address. . . . . : 00-00-00-00-00-00-E0
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . : Yes
IPv6 Address. . . . . : 2002:5040:4f02::5040:4f02(Preferred)
Default Gateway . . . . . : 2002:c058:6301::c058:6301
DNS Servers . . . . . : 80.64.65.69
NetBIOS over Tcpip. . . . . : Disabled

```

61. ábra 6to4 interface "ipconfig /all" kimenete

Ha nem az RFC 3068-ban meghatározott 6to4 relay unicast címet (192.88.99.1) szeretnénk használni, akkor a netsh paranccsal megadhatjuk az új címet (62. ábra).

```

C:\Windows\system32>netsh interface 6to4 set relay 1.2.3.4
Ok.

C:\Windows\system32>netsh interface 6to4 show relay
Relay Name       : 1.2.3.4
Use Relay        : default
Resolution Interval : default

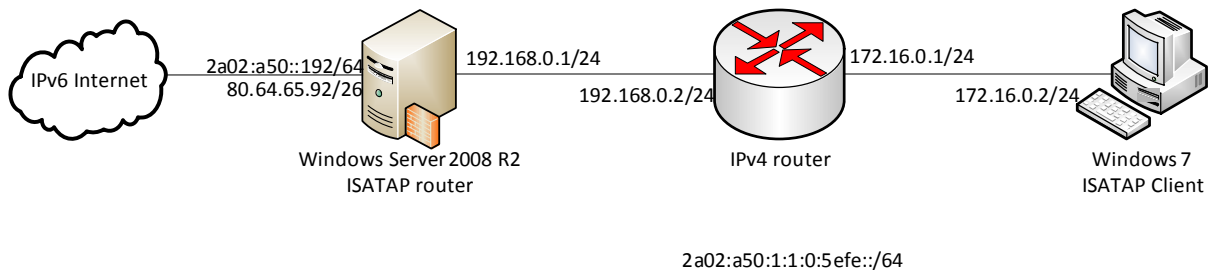
C:\Windows\system32>

```

62. ábra 6to4 relay címének beállítása Windows 7 számítógépen

9.5. ISATAP hálózat létrehozása Windows Server 2008 R2 ISATAP routerrel

A 63. ábraán megadott hálózatot alakítottam ki a teszt elvégzéséhez. Az IPv4-es router kizárólag az IPv4-et ismeri, nem támogatja az IPv6 protokollt. A Windows Serveren található DNS kiszolgáló és Routing and Remote Access Services szerver az IPv4-es címfordítás elvégzéséhez, melyek segítségével a jelenlegi Internetelés meg van oldva.



63. ábra Hálózat Windows Server 2008 R2 ISATAP routerrel

Elsőként a DNS beállítására van szükség. A Windows Server 2008 esetében biztonsági okokból az ISATAP névre nincs feloldás, csak ha azt külön engedélyezzük. Ennek lekérdezése látható az ábrán (64. ábra).

```

C:\Users\Administrator>dnscmd /info /globalqueryblocklist
Query result:
String: wpad
String: isatap

Command completed successfully.

C:\Users\Administrator>_

```

64. ábra DNS biztonsági beállítás lekérdezése

A listából úgy törölhetünk, hogy megadjuk a lista új tagjait. Az ábrán jól látható, hogy sikerült az ISATAP hostot törölni (65. ábra).

```

C:\Users\Administrator>dnscmd /config /globalqueryblocklist wpad
Registry property globalqueryblocklist successfully reset.
Command completed successfully.

C:\Users\Administrator>dnscmd /info /globalqueryblocklist
Query result:
String: wpad
Command completed successfully.

C:\Users\Administrator>

```

65. ábra Az ISATAP host törlése, és ellenőrzése

Ezek után létre kell hozni a zónában az ISATAP A rekordot az ISATAP router belső, privát IPv4-es címével (66. ábra).

66. ábra ISATAP A rekord létrehozása

Ezután újra kell indítani a szerveret, hogy megpróbálja feloldani az ISATAP hostnevet. Ha ez sikerül neki, akkor az ISATAP Tunnel adapternek kioszt egy IPv6-os link local típusú címet. Ezt az `ipconfig` paranccsal tudjuk ellenőrizni (67. ábra).

```

C:\Users\Administrator>ipconfig

Windows IP Configuration

Ethernet adapter Internal:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::9e5:42aa:a4a:48fb%14
    IPv4 Address. . . . . : 192.168.0.1
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 

Ethernet adapter Internet:

    Connection-specific DNS Suffix  . : 
    IPv6 Address. . . . . : 2a02:a50::192
    IPv6 Address. . . . . : 2a02:a50::792a:adcc:6540:65ee
    Link-local IPv6 Address . . . . . : fe80::792a:adcc:6540:65ee%10
    IPv4 Address. . . . . : 80.64.65.92
    Subnet Mask . . . . . : 255.255.255.192
    Default Gateway . . . . . : 2a02:a50::1
                                fe80::20c:42ff:fe9a:4b84%10
                                80.64.65.65

Tunnel adapter isatap.{8F6A7444-50CD-4031-A94F-08F76CEDB1F5}:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::200:5efe:80.64.65.92%11
    Default Gateway . . . . . : 

Tunnel adapter Teredo Tunneling Pseudo-Interface:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

Tunnel adapter isatap.{F8063A54-A8AA-403E-B58C-ABB10429BB6C}:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::5efe:192.168.0.1%15
    Default Gateway . . . . . : 

C:\Users\Administrator>_

```

67. ábra IPv6-os cím a szerver ISATAP interfészen

Ezután a `netsh interface ipv6 show interface` paranccsal le kell kérdeznünk a belső hálózat ISATAP Tunnel interface ID-jét, ami jelen esetben a 15-ös (68. ábra).

```

C:\Users\Administrator>netsh interface ipv6 show interface

```

Idx	Met	MTU	State	Name
1	50	4294967295	connected	Loopback Pseudo-Interface 1
11	5	1280	connected	isatap.{8F6A7444-50CD-4031-A94F-08F76CEDB1F5}
10	5	1500	connected	Internet
12	50	1280	disconnected	Teredo Tunneling Pseudo-Interface
15	5	1280	connected	isatap.{F8063A54-A8AA-403E-B58C-ABB10429BB6C}
14	5	1500	connected	Internal

68. ábra Interface ID-k lekérdezése

Ekkor már be tudjuk kapcsolni a szerveren, hogy magát ISATAP routerként hirdesse az ISATAP kliensek felé. Ehhez a `netsh interface ipv6 set interface 15 advertise=enabled` parancsot kell kiadni (69. ábra).

```

C:\Users\Administrator>netsh interface ipv6 set interface 15 advertise=enabled
Ok.

C:\Users\Administrator>

```

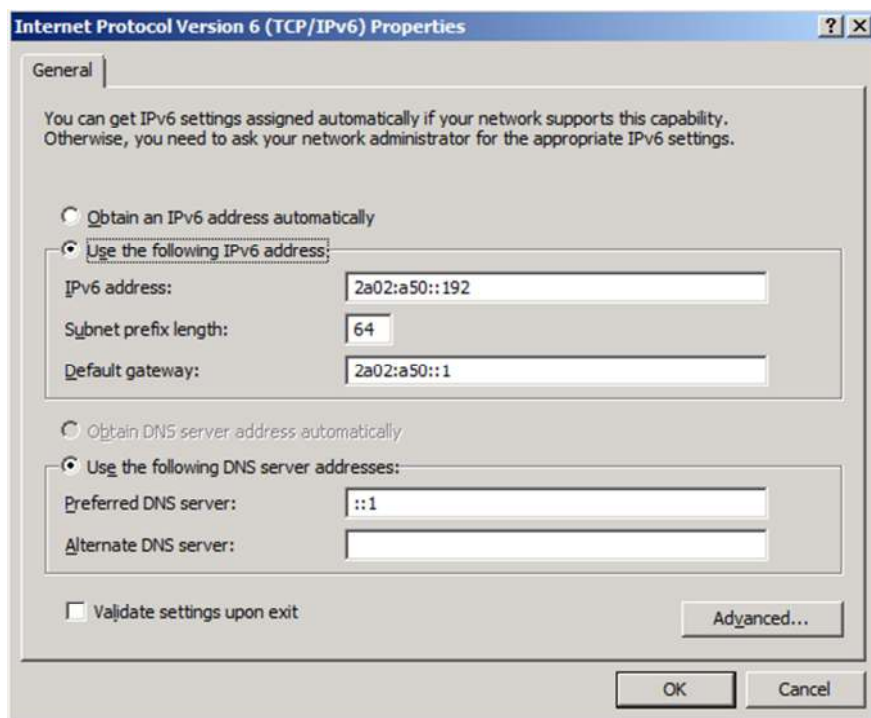
69. ábra ISATAP router beállítása

Ezután be kell állítani a szolgáltatótól kapott IPv6 prefixet, hogy a kliensek ezt használják IPv6-os címük előállításához. Ez a prefix a példában a `2a02:a50:1:1::/64`. Így a beállításhoz szükséges parancs `netsh interface ipv6 add route 2a02:a50:1:1::/64 15 publish=yes` (70. ábra).

```
C:\Users\Administrator>netsh interface ipv6 add route 2a02:a50:1:1::/64 15 publi
sh=yes
Ok.
C:\Users\Administrator>
```

70. ábra ISATAP szolgáltatáshoz használt prefix beállítása

Végül érdemes default routként hirdetni az ISATAP kliensek felé az újonnan beállított ISATAP routerünket. Figyelni kell arra, hogy ha meg van adva az IPv6 defult route a Windows Server 2008-on, akkor előbb ezt távolítsuk el, ahogy az ábrán is látszik (71. ábra). A default route hirdetéssel együtt a `netsh interface ipv6 add route ::/0 InternetInterfaceIndex nexthop=Internetszolgáltatóiátjáró` paranccsal állítható be (72. ábra) [43].



71. ábra Internet interface IPv6 beállítások default gateway nélkül

```
C:\Users\Administrator>netsh interface ipv6 add route ::/0 10 nexthop=2a02:a50::
1 publish=yes
Ok.
C:\Users\Administrator>
```

72. ábra IPv6 default route beállítása parancssorból

Ha ezek után újraindítjuk az Windows 7-es kliensünket, az ISATAP interface az ábrán látható Media disconnected állapotból átkerül a használható állapotba (73. ábra és 74. ábra).

```

C:\Users\repa>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::9940:ce35:8471:e00f%11
    IPv4 Address. . . . . : 172.16.0.2
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 172.16.0.1

Tunnel adapter isatap.{9326A7AD-0DF3-459E-BD37-0731D40301DA}:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

```

73. ábra Disconnected ISATAP interface

```

C:\Users\repa>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::9940:ce35:8471:e00f%11
    IPv4 Address. . . . . : 172.16.0.2
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 172.16.0.1

Tunnel adapter isatap.{9326A7AD-0DF3-459E-BD37-0731D40301DA}:

    Connection-specific DNS Suffix  . : 
    IPv6 Address. . . . . : 2a02:a50:1:1:0:5efe:172.16.0.2
    Link-local IPv6 Address . . . . . : fe80::5efe:172.16.0.2%12
    Default Gateway . . . . . : fe80::5efe:192.168.0.1%12

Tunnel adapter Teredo Tunneling Pseudo-Interface:

    Connection-specific DNS Suffix  . : 
    IPv6 Address. . . . . : 2001:0:5ef5:79fb:3858:1003:afbf:bea3
    Link-local IPv6 Address . . . . . : fe80::3858:1003:afbf:bea3%13
    Default Gateway . . . . . : 

C:\Users\repa>

```

74. ábra Felkonfigurált, működő ISATAP interface

Jól megfigyelhető, hogy megjelent a default gateway is az IPv6 routing táblában. (75. ábra)

```

C:\Users\repa>route -6 print

=====
Interface List
11...00 15 5d 20 42 7a .....Microsoft Virtual Machine Bus Network Adapter
1.....Software Loopback Interface 1
12...00 00 00 00 00 00 00 e0 Microsoft ISATAP Adapter
13...00 00 00 00 00 00 e0 Teredo Tunneling Pseudo-Interface
=====

IPv6 Route Table
=====
Active Routes:
If Metric Network Destination Gateway
12 261 ::/0 fe80::5efe:192.168.0.1
1 306 ::1/128 On-link
13 58 2001::/32 On-link
13 306 2001:0:5ef5:79fb:3858:1003:afbf:bea3/128 On-link
12 13 2a02:a50:1:1::/64 On-link
12 261 2a02:a50:1:1:0:5efe:172.16.0.2/128 On-link
11 261 fe80::/64 On-link
13 306 fe80::/64 On-link
12 261 fe80::5efe:172.16.0.2/128 On-link
13 306 fe80::3858:1003:afbf:bea3/128 On-link
11 261 fe80::9940:ce35:8471:e00f/128 On-link
1 306 ff00::/8 On-link
13 306 ff00::/8 On-link
11 261 ff00::/8 On-link
=====
Persistent Routes:
None

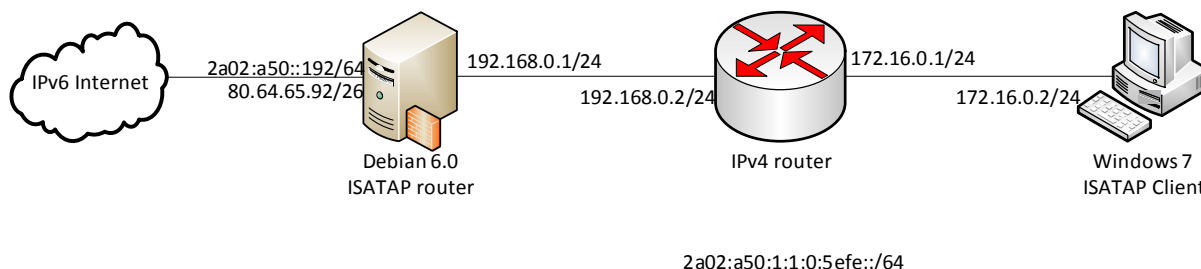
C:\Users\repa>

```

75. ábra IPv6 kliens routing tábla

9.6. ISATAP hálózat létrehozása Linux ISATAP routerrel

Itt is egy már működő hálózatról indultam ki. A konfiguráció megegyezik az előzővel, azzal a különbséggel, hogy itt az ISATAP router szerepét egy Debian 6.0 Linux radvd-val tölti be (76. ábra).



76. ábra Hálózat Linux alapú ISATAP routerrel

Elsőként a DNS szerveren kell beállítani az ISATAP A rekordot a belső hálózati címre a zónafájlból (77. ábra).

```
$ttl 38400
@      IN      SOA      DebianIPv6.isataptest.local. root.isataptest.local. (
                                2012012402
                                10800
                                3600
                                604800
                                38400 )
@      IN      NS       DebianIPv6.isataptest.local.
DebianIPv6  IN      A       192.168.0.1
isatap      IN      A       192.168.0.1
```

77. ábra ISATAP A rekord a zónában

Ezután a radvd telepítése az `apt-get install radvd` paranccsal megoldható. A `/etc/radvd.conf` konfigurációs állományban be kell állítanunk az ISATAP tunnelhez a prefix és a routing hirdetések (78. ábra).

```
interface is0
{
    AdvSendAdvert on;
    UnicastOnly on;
    AdvHomeAgentFlag off;

    prefix 2a02:a50:1:1::/64
    {
        AdvOnLink on;
        AdvAutonomous on;
        AdvRouterAddr off;
    };
};
```

78. ábra radvd beállítása a `/etc/radvd.conf` állományban

Engedélyezni kell az IPv6-os routings a kernelben. Ez a legegyszerűbben a `/etc/sysctl.conf` – ba írt `net.ipv6.conf.all.forwarding=1` sorral oldható meg (79. ábra).

```
# Uncomment the next line to enable packet forwarding for IPv6
# Enabling this option disables Stateless Address Autoconfiguration
# based on Router Advertisements for this host
net.ipv6.conf.all.forwarding=1
```

79. ábra IPv6 routing engedélyezése Linux kernelben

Végül a tunnelt is létre kell hozni, amit legegyszerűbben a `/etc/network/interfaces` állományban lehet beállítani. A belső interfészen elhelyezett három darab `post-up` sor létrehozza az ISATAP tunnelt (`is0` néven), majd engedélyezi azt, végül pedig beállítja az IPv6-os címét (80. ábra) [44] [45].

```
iface eth1 inet static
    address 192.168.0.1
    netmask 255.255.255.0
    post-up ip tunnel add is0 mode isatap local 192.168.0.1 ttl 64
    post-up ip link set is0 up
    post-up ip addr add 2a02:a50:1:1:0:5efe:192.168.0.1/64 dev is0
    pre-down ip link set is0 down
    pre-down ip tunnel del is0
```

80. ábra ISATAP tunnel létrehozása Linuxon

Ha ezután újraindítjuk a Linuxot, a Windows 7-es kliens ugyanúgy működik, mintha Windows Server 2008 R2 lenne az ISATAP router. A tesztek során semmilyen működési problémát nem tapasztaltam, így elmondható, hogy a két rendszer ISATAP megoldása kompatibilis (81. ábra).

```
C:\Users\repa>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix . : 
    Link-local IPv6 Address . . . . . : fe80::9940:ce35:8471:e00f%11
    IPv4 Address. . . . . : 172.16.0.2
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 172.16.0.1

Tunnel adapter isatap.{9326A7AD-0DF3-459E-BD37-0731D40301DA}:

    Connection-specific DNS Suffix . : 
    IPv6 Address. . . . . : 2a02:a50:1:1:0:5efe:172.16.0.2
    Link-local IPv6 Address . . . . . : fe80::5efe:172.16.0.2%12
    Default Gateway . . . . . : fe80::5efe:192.168.0.1%12

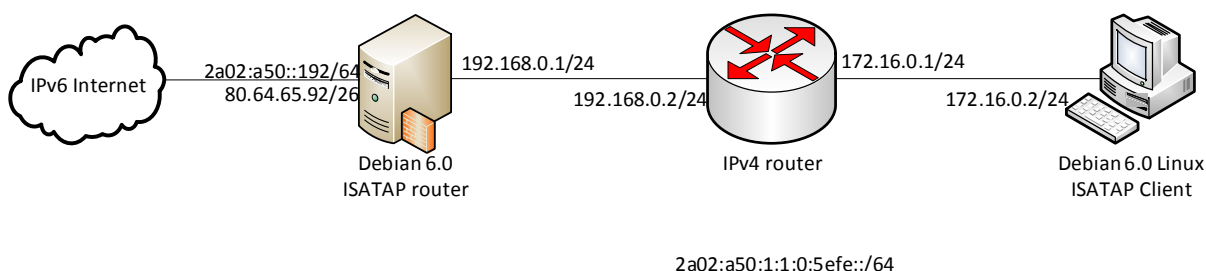
Tunnel adapter Teredo Tunneling Pseudo-Interface:

    Connection-specific DNS Suffix . : 
    IPv6 Address. . . . . : 2001:0:5ef5:79fd:3417:3a58:53ef:fffd
    Link-local IPv6 Address . . . . . : fe80::3417:3a58:53ef:fffd%13
    Default Gateway . . . . . : 

C:\Users\repa>
```

81. ábra Windows 7 kliens Linux ISATAP routerrel

9.7. Linux ISATAP kliens beállítása



82. ábra Linux ISATAP hálózat

A Debian csomagmenedzserével telepíteni kell az isatap daemont, a következők szerint: `apt-get install isatapd`. Az ISATAP kliens Linuxos megvalósítása is képes magát az isatap hosztnév rezolválásával automatikusan konfigurálni, de akár a `/etc/default/isatapd` állományban is megadhatjuk az ISATAP router IP címét. (83. ábra) Az ISATAP routeren futó DNS-nek és RADVD-nek köszönhetően minden konfigurálás nélkül megkapta a kliens a szükséges beállításokat, mint amilyen az IPv6-os cím, átjáró, ISATAP router IPv4-es címe, stb [46].

```
# Set RUN=yes to start isatapd at boot time. You may need to set the
# ISATAP_ROUTER parameter below
RUN="yes"

# A space seperated list of one or more hostnames/IPv4 addresses to use as
# potential routers.
# The default is the unqualified hostname 'isatap'
ISATAP_ROUTERS="192.168.0.1"
```

83. ábra ISATAPD konfigurálása