Hálózati Rendszerek és Szolgáltatások Tanszék

IPv6 és IPv6 áttérési technológiák vizsgálata

Felkészülési segédlet

v1.1.0

A segédletet összeállította: Dr. Lencse Gábor, BME HIT, utoljára frissítve: 2019. 08. 16.

Tudnivalók:

- A felkészülési segédlet a tárgy anyagának részét képezi, ZH-ra is tudni kell, és lesz belőle beugró is, amely beleszámít a mérés érdemjegyébe, így a végső érdemjegybe is. A mérésvezetők a felkészületlen hallgatót pótmérésre kötelezhetik. Fő célja mégis az, hogy a hallgatók teljes mértékben megértsék a mérés elméleti hátterét. Az anyag mérés közben már elővehető, szükség esetén referenciaként használható.
- FIGYELEM! A mérésben erősen építünk az első két mérés során megszerzett tudásra és tapasztalatokra. Azok elvégzése nélkül NE kíséreljék meg a harmadik mérés elvégzését!
- Az továbbra is fontos, hogy a hallgatók a mérésre kipihenten és pontosan érkezzenek. Kérjük továbbá, hogy a mérés megkezdése előtt némítsák le a mobiltelefonjukat. ☺
- A mérések célja az, hogy a hallgatók gyakorlati módon ismerjék meg a tananyagot, a harmadik mérésben az *IPv6 és a DNS64+NAT64 IPv6 áttérési technológiák működését*, és a saját szemükkel lássák, hogy az IPv6-ban alkalmazott egyes megoldások mennyire eltérőek az IPv4 megoldásaihoz képest.
- A mérések anyaga publikus, de a feladatok előre történő megoldását nem javasoljuk: a mérés nem vizsga, hanem lehetőség a hatékony tanulásra. Ezt felkészült és segítőkész mérésvezetők is támogatják azzal, hogy bármikor lehet tőlük kérdezni, segítséget kérni, ha valaki elakadna. Kérjük, hogy a mérést komolyan végezzék, mert a jegyzőkönyvet is értékeljük a beugró mellett.
- Ebben a segédletben a következő témaköröket tárgyaljuk: IPv6 (címzés, datagram felépítése, ICMPv6, NDP, MLD, PMTUD) és IPv6 áttérési technológiák (DNS64, NAT64, 464XLAT, DS-Lite, MAP-E/T, Lw4o6, egyebek).
- A segédlet olvasójáról feltételezzük, hogy az első két mérésre rendesen felkészült, és azokat sikeresen elvégezte. Ellenkező esetben a továbbiak elolvasása előtt kérjük a hiányosságot pótolni (mert különben gondok lehetnek az anyag megértésével).
- A segédlet az IPv6 könyvön [1] alapul, amely mindenki számára ingyenesen elérhető és a benne szereplő további témaköröket az IPv6 és/vagy az IPv6 áttérési technológiák iránt érdeklődők figyelmébe ajánljuk. Azonban a könyv megírása óta már 4 év telt el, és ezért ez a segédlet a könyvhöz képest frissebb, az eltérések különösen az IPv4aaS fejezet táján jelentősek.

Tartalom

1. Az IPv6 protokoll

1.1. IPv6 címek

Az IPv6 protokoll egyik legszembetűnőbb eltérése az IPv4-hez képest a címek mérete. Az IPv6 128 bites címeket használ, így még a címtartomány nem hatékony felhasználása esetén is hosszú időre elegendő címet biztosít a várható és az előre nem látható feladatokra.

1.1.1. IPv6 címek írása

Rugalmas formátum

Az IPv6 címek szöveges leírásában az RFC 4291 meglehetősen rugalmas, a felhasználónak számos lehetőséget nyújt az egyszerűsítésre, de ezeket nem teszi kötelezővé. Az IPv6 cím 128 bitjét 16 bites csoportonként négy hexadecimális jeggyel írjuk, a csoportokat kettősponttal választjuk el egymástól. A csoportok elején a nullák mindig elhagyhatók. Így amennyiben egy 16 bites csoportban négy nulla áll egymás után, azt mindig helyettesíthetjük egy nullával. Ha egymás utáni 16 bites csoportok csak nullákat tartalmaznak, az összes ilyen csoportot helyettesíthetjük dupla kettősponttal („::”), de ez csak egyszer alkalmazható, hogy a dekódolás egyértelmű legyen. Lássunk egy példát:

2001:0DB8:0000:00AE:0000:0000:0000:ABCD
↓
2001:DB8:0:AE:0:0:0:ABCD
↓
2001:DB8:0:AE::ABCD

Amennyiben egy IPv6 címet IPv4 címből képeztünk, és az IPv6 cím az IPv4 címet az utolsó 32 bitjén tartalmazza, akkor az IPv4 címet tartalmazó részt írhatjuk az IPv4 kanonikus formátumában is, például a 64:FF9B::C000:20A cím írható úgy is, hogy: 64:FF9B::192.0.2.10.

A hexadecimális számjegyekben szereplő betűkkel kapcsolatban sincs megkötés, hogy kis- vagy nagybetűk legyenek, a példákban az RFC 4291 nagybetűket használ.

Gyakran van szükség egy IPv6 cím első *n* bitjének a megadására, amit *n* méretű *előtag*nak (prefix) nevezünk és az előtag után „/*n*” megadásával jelölünk. Fontos, hogy prefix írásakor az utolsó olyan 16 bites csoportot, ami nem csupa 0-t tartalmaz, mindig végig ki kell írni.

Példa: 2001:DB8:AB00::/40 a helyes írásmód, mert ha 2001:DB8:AB::/40-et írnánk, az azt jelentené, hogy: 2001:0DB8:00AB::/40, ami pedig valójában: 2001:0DB8::/40.

A dupla kettőspont használatával prefixeknél különösen körültekintően kell bánni, például a 2001:DB8:0:A::/64 nem rövidíthető tovább úgy, hogy 2001:DB8::A/64, mert az teljesen mást jelent, konkrétan azt, hogy: 2001:DB8:0:0:0:0:0:A/64.

Az RFC 3986 definiálja az IPv6 címek használatát URL-ekben. Példák:

- http://[2001:DB8:FFFF:FFFF::192.224.128.1]:8080/index.html
- ftp://[2001:DB8:2C01:8000::15]

Kanonikus formátum

Az RFC 4291 rugalmassága nagyon kényelmes, így egy IPv6 címnek sokféle írásmódja lehetséges attól függően, hogy a felhasználó mely lehetőségekkel kíván élni, és melyekkel nem. Azonban az RFC 5952 leírja, hogy ez a szabadság számos esetben problémákat okozhat a gyakorlatban. Például gondoljuk arra, hogy egy naplófájlban a 2001:db8::a:0:0:1 címet szeretnénk megkeresni, miközben az olyan formában szerepel benne, hogy: 2001:0DB8:0:0:A::1. A probléma megoldására bevezetik az IPv6 címek szöveges írásának *kanonikus* formáját azzal, hogy egy program bemenetként minden olyan címet *köteles* (MUST) elfogadni, amit RFC 4291 megenged, de kimenetén *erősen ajánlott* (SHOULD) a kanonikus forma használata. Továbbá azt tanácsolják, hogy az IPv6 címek leírásakor mi emberek is használjuk a kanonikus formát. Ennek szabályai a következők:

- Az egyes 16 bites csoportokban a vezető nullákat el *kell* (MUST) hagyni.
- A dupla kettőspontot a maximális kapacitásáig ki *kell* (MUST) használni.
- A duplakettőspontot *tilos* (MUST NOT) egyetlen 16 bites csoport rövidítésére használni.

- Amennyiben több csupa nullás csoport van, akkor a leghosszabbat, ha pedig a csoportok hossza egyenlő, akkor balról jobbra haladva az elsőt *kell* (MUST) dupla kettősponttal helyettesíteni.
- Hexadecimális számokban kisbetűket *kell* (MUST) használni.

Ebben a segédletben a továbbiakban törekszünk a fentiek betartására, de címekben (vagy definíciós címkékben) szándékosan csupa nagybetűt használunk, és máshonnan átvett anyagokat (pl. ábrák, RFC részletek) sem módosítunk.

1.1.2. Az IPv6 címzési architektúrája

Az IPv6 címtartomány különféle célokra való felosztását prefixek segítségével lehet megadni. A következő főbb kategóriákat definiálták (RFC 4291):

Cím típusa	Bináris prefix	IPv6 jelöléssel
Unspecified	00...0 (128 bit)	::/128
Loopback	00...1 (128 bit)	::1/128
Multicast	11111111	FF00::/8
Link-Local unicast	1111111010	FE80::/10
Global Unicast	(minden más, ami nem speciális)	

A következőkben az egyes kategóriákat részletesen megvizsgáljuk.

Speciális IPv6 címek/prefixek

Az alábbi két címet minden host használja:

::/128 – Unspecified Address

Meghatározatlan cím, akkor használjuk, amikor egy host inicializálja magát.

::1/128 – Loopback Address

A 127.0.0.1 IPv4 loopback címhez hasonlóan használt loopback cím.

FF00::/8 – Multicast Address

Az IPv4-ben broadcastnak és multicastnak nevezett funkciók megvalósítására egyaránt használt címtartomány. Részletesen foglalkozunk vele.

FE80::/10 – Link-Local IPv6 Unicast Addresses

Olyan címek, amik csak egy adott fizikai linken érvényesek, például egy Ethernet szegmensen. Az ilyen forráscímmel rendelkező IPv6 csomagokat az útválasztók (routerek) nem továbbítják. Olyan esetekben használjuk, amikor például egy IPv6 hálózatban nincs router, vagy Neighbour Discovery (lásd később) esetén. Minden hálózati interfészhez kötelezően tartozik ilyen cím. Amikor ilyen címet használunk egy programban forráscímként, akkor kötelezően meg kell adni, hogy melyik interfészen menjen ki a csomag (hiszen a link-lokális címből nem derül ki).

FEC0::/10 – Site-Local IPv6 Unicast Addresses – ÉRVÉNYTELEN!

Ez a tartomány ma már nem használt, érvénytelenített. Eredetileg az *egy adott site körzetén belüli* címzésre definiálták (mint IPv4-nél az RFC 1918 szerinti lokális címeket), de a site fogalmának nehéz meghatározhatósága miatt már nem használatos, lásd: RFC 3879.[1]

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

FC00::/7 – Unique Local IPv6 Unicast Addresses

Az RFC 4193 definiálja. Az *egyedi lokális címek* (a továbbiakban ULA) használatával olyan helyeken is lehet IPv6-ot használni, ahol nincs hivatalosan kiosztott IPv6 prefix, azaz felhasználásuk célja teljesen hasonló az RFC 1918-ban meghatározott privát IPv4 címekhez. Nagy különbség azonban, hogy az ilyen címeknél a *hálózatcím* (Network ID) meghatározásához egy *álvéletlen* (pseudo-random) számot használunk, amit az RFC-ben megadott módon lehet előállítani. Több szervezeti egység is generálhat így magának címet, és a pseudo-random algoritmusnak köszönhetően nagyon kicsi valószínűséggel fognak ezek a címek ütközni. (Segítségükkel tehát sikerült kiküszöbölni az érvénytelenített site-local unicast címek hibáit.)

Felépítésük az alábbi:

```
| 7 bits |1|  40 bits   |  16 bits  |          64 bits           |
+--------+-+------------+-----------+----------------------------+
| Prefix |L| Global ID  | Subnet ID |        Interface ID        |
+--------+-+------------+-----------+----------------------------+
```

Ahol:

- Prefix: fc00::/7
- L: kötelezően 1 értékű
- Global ID: véletlenszerűen választott, igen jó eséllyel globálisan egyedi[2]
- Subnet ID: a hálózatcím használója osztja ki a hálózatainak
- Interface ID: a hálózati interfész 64 bites azonosítója, lásd később

Az *IPv4-ről IPv6-ra való átállás* (IPv6 transition) támogatására szánták az alábbi IPv6 címtípusokat azzal a céllal, hogy IPv6 hálózatokban IPv4 címeket reprezentáljanak:

::/96 – IPv4-Compatible IPv6 Addresses – ÉRVÉNYTELEN!

IPv6 hálózatokban az x.y.z.w IPv4 címet reprezentálta volna a következő alakban: ::x.y.z.w (az elején 96 db 0 értékű bit volt). Eredetileg az IPv4 → IPv6 átmenethez való felhasználásra szánták, de már más megoldást (lásd következő) találtak ki, ezért érvénytelenítették.

::FFFF:0:0/96 – IPv4-Mapped IPv6 Addresses (ez használható!)

IPv6 hálózatokban az x.y.z.w IPv4 címet reprezentálja a következő alakban: ::ffff:x.y.z.w (az elején 80 db 0 értékű bit van). A korábbi ::x.y.z.w helyett azért kellett másik, mert változott a módszer, amit az IPv4 → IPv6 átmenethez szeretnének alkalmazni.

Megjegyzés: a fenti prefix végén szükséges volt kiírni a két darab 16 bites (csupa 0 értékű) csoportot jelölő „:0:0” részt, mert a „::ffff/96” jelölésnél a 16 darab 1 értékű bit a 128 bites IPv6 cím legalsó 16 bitjének a helyére kerül. (Lásd az RFC 6890 hibajegyzékében a 2. bejegyzést.)

Hálózati alkalmazások készítése esetén ennek a címtípusnak a használatával lehetséges az, hogy egyetlen socketet használjunk mind IPv6, mint IPv4 kommunikációra. Ilyenkor elegendő a socket kezelő függvényeknek csak az IPv6-os verzióját használni. Ha IPv4-mapped IPv6 címeket adunk meg, akkor a (Linux) kernel IPv4-es csomagokat küld ki, illetve ha az alkalmazás által figyelt portra IPv4 csomag érkezik, akkor ilyen IPv6 címekkel kapja meg az alkalmazás a csomagot. Ezzel a megoldással igen jelentős mennyiségű programozási munkát takaríthatunk meg.

[2] Megjegyzés: létezett olyan szolgáltatás (https://www.sixxs.net/tools/grh/ula/), ahol generáltak számunkra ULA IPv6 címet, és be is lehetett azt jegyeztetni, de mivel használata nem volt kötelező, ezért semmilyen garanciát sem nyújtott a cím egyediségére. (Ez a szolgáltatás a SixXS project befejeződésével 2017. 06. 06-án megszűnt: https://www.sixxs.net/sunset/.)

2001:DB8::/32 – IPv6 szabványos dokumentációs prefix

Annak érdekében, hogy a dokumentációkban szereplő példák ne okozzanak zavart (a fejekben) vagy működő rendszerekkel való ütközést, lefoglaltak egy globális unicast prefixet kifejezetten dokumentációs célra. Ez a 2001:db8::/32 (RFC 3849). Ezt a prefixet soha senkinek sem fogják kiosztani és nem is routolják. De természetesen ettől még nem számít lokális unicast prefixnek!

(Így ha egy példában ezt a prefixet használják, és valaki a példát szó szerint begépeli, az nem fog kárt okozni, mert a prefixet a valós életben soha semmire nem használjuk. A prefixet helyi hálózatokban is kiszűrhetik; ezért senki ne számítson rá, hogy működni fog, ha használni próbálná!)

Megjegyzés: IPv4-nél is vannak ilyenek, az RFC 5737 szerint: 192.0.2.0/24, 198.51.100.0/24, 203.0.113.0/24.

A speciális célú IPv4 és IPv6 címekről összefoglaló található: RFC 6890

Az IPv6 multicast címzése

Fontos, hogy IPv6-ban az IPv4-gyel ellentétben broadcast cím nincs. Multicast címzésre az ff00::/8 tartomány használható. A multicast címek felépítése az alábbi:

bitek száma	8	4	4	112
mező neve	*prefix*	*flags*	*scope*	*group ID*

Az egyes mezők jelentése:

- **prefix**: a cím csoportcím voltát jelző *előtag*, értéke: ff
- **flags**: ebben a mezőben *jelzőbiteket* definiáltak: 0, R, P, T
- **scope**: a cím érvényességének a *hatókör*ét fejezi ki (0-f)
- **group ID**: az egyes *multicast csoportok azonosítására* használható bitek

A flags mezőben található jelzőbitek értelmezése:

- **0**: fenntartott (reserved)
- **R**: A csoporthoz tartozó *randevú pont* (Rendezvous point) címe a csoportcímbe beágyazott-e (RFC 3956), 1: igen, 0: nem.
- **P**: A csoportcímet az azt létrehozó szervezet *hálózatának előtagja* (Prefix) alapján generálták-e (RFC 3306), 1: igen, 0: nem.
- **T**: A csoportcím dinamikus-e (Transient), 1: igen, 0: nem, azaz az IANA által kiosztott well-known multicast address. Lásd: http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xml

A csoportcímek érvényességi körét kifejező 4 bites scope mező lehetséges értékei:

- **0**: Reserved – fenntartott
- **1**: Interface-local – multicast loopback átvitelére használható
- **2**: Link-Local – hatóköre a fizikai hálózat broadcast domainje
- **4**: Admin-Local – hatóköre a lehető legkisebb értelmes adminisztratív tartomány
- **5**: Site-Local – hatóköre egy fizikai telephely
- **6-7**: Unassigned – a hálózati adminisztrátorok definiálhatják
- **8**: Organization-Local – egy szervezet összes telephelyére kiterjed
- **9-D**: Unassigned – a hálózati adminisztrátorok definiálhatják
- **E**: Global – globális
- **F**: Reserved – fenntartott

Az IANA által kiosztott néhány általános célú csoportazonosító (group ID):

- **FF02::1** – link local all nodes – az összes eszköz az adott fizikai hálózaton (broadcast domainben)
- **FF02::2** – link local all routers – minden útválasztó a fenti körben
- **FF02::5** – link local all OSPF routers – minden OSPF útválasztó a fenti körben
- **FF02::D** – link local all PIM routers – minden PIM útválasztó a fenti körben

- **FF02::16** – link local all MLDv2 routers – minden MLDv2 útválasztó a fenti körben
- **FF05::2** – site local all routers – a telephely összes útválasztója

Vegyük észre, hogy az FF02::1 jelentése nem más, mint IPv4 esetén az aktuális hálózatra vonatkozó broadcast cím! Megjegyzés: IPv4-ben is van ilyen csoportcím: 224.0.0.1

Vannak minden scope esetén érvényes csoportazonosítók is, például:

- **FF0x::C** – SSDP – Az SSDP (Simple Service Discovery Protocol) által használt IPv6 csoportcím. (IPv4 alatt a 239.255.255.250 csoportcímre „szemetel" az SSDP.)

Meg kell ismernünk még egy fontos csoportcímet, amelyre később fontos szerep hárul majd. Ez a kérdéses *eszköz* (node) IPv6 címének felhasználásával képzett link-lokális hatókörű *solicited-node multicast address*. Előállításához az ff02:0:0:0:0:1:ff00:0/104 prefixhez kell hozzáírni a kérdéses node IPv6 címének legkisebb helyiértékű 24 bitjét. Például a 2001:db8::800:abba:edda IPv6 címhez tartozó ilyen multicast cím: ff02:0:0:0:0:1:ffba:edda. Mivel egy eszköznek csatlakozni kell az összes IPv6 címéhez tartozó solicited-node multicast csoporthoz, a csoportcímnek az IPv6 cím utolsó 24 bitjéből való képzése előnyös lehet, ha az interfész többféle prefix használatával képzett IPv6 címmel is rendelkezik[3], ugyanis ekkor csak egyetlen csoportról van szó (amennyiben az IPv6 címeinek utolsó 64 bitjét a hálózati interfész azonosítójából képezték). Azt pedig a későbbiekben fogjuk látni, hogy egy adott multicast csoportban nem fognak túlságosan sokan tartózkodni, sőt gyakran csak egy hálózati interfész tartozik bele (az összes IPv6 címével).

Itt említjük még meg, hogy az IPv6 multicast Ethernet szinten a csoportcímben a „33:33" prefixet használja, utána pedig az IPv6 csoportcím utolsó 32 bitje következik. (Tehát fentiek alapján a solicited-node multicast address esetén az Ethernet csoportcím prefixe „33:33:ff" lesz.)

Az IPv6 Anycast címzése

Az IPv6 címzési architektúrája háromféle címtípust sorol fel: unicast, multicast és anycast. Az anycast címek szintaktikailag unicast címek, csak a használatuk módja eltérő. Míg unicast esetén minden hálózati interfész címe különböző, addig anycast esetén ugyanazt a címet több különböző számítógép hálózati interfészéhez is hozzárendelik. Egy anycast címre küldött csomagot az adott anycast címmel rendelkező interfészek közül (a hálózatban használt routing protokoll metrikája szerint) a forráshoz legközelebb található interfésznek kézbesítik. (Az anycast címzést IPv4-nél is alkalmazzák.)

Ismereteink rendszerezésére tekintsük most át a címzési módszereket!

- **Unicast**: A csomag pontosan egy általunk kiválasztott állomásnak szól.
- **Broadcast**: A csomag (az adott hálózaton vagy tartományban) az összes állomásnak szól.
- **Multicast**: A csomag egy csoport összes tagjának szól.
- **Anycast**: A csomag egy csoport tagjai közül egynek szól, de nem a feladó, hanem a hálózat dönti el, hogy melyik legyen az. Tipikus választás, hogy legyen a küldőhöz legközelebb eső csoporttag.

Az egyes módszerek működését illusztrálja az 1. ábra.

Az anycast címzés megvalósítása:

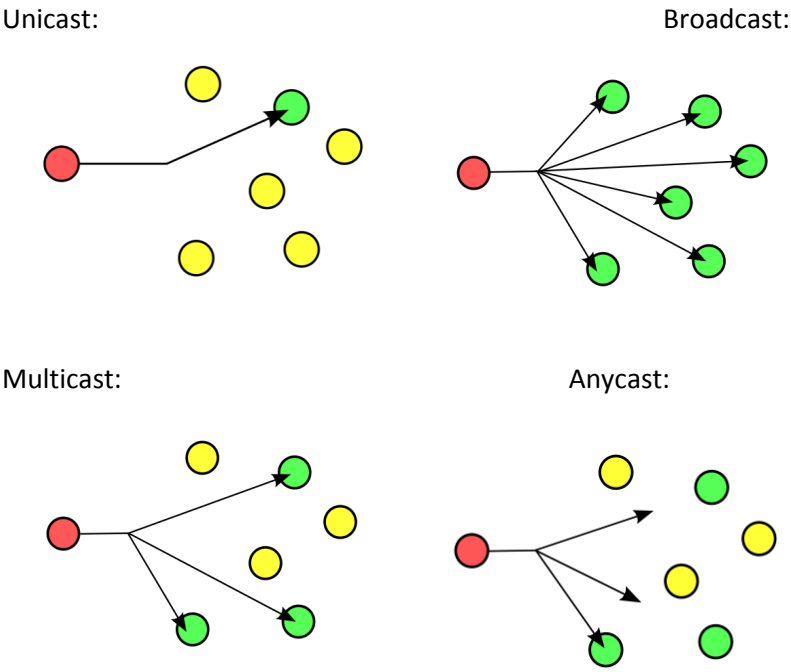
- Ugyanazt az IP-címtartományt (közönséges unicast címek!) több helyen is alkalmazzák és hirdetik az Interneten (BGP-vel). (Természetesen akár egy szolgáltató hálózatán belül is alkalmazható az Anycast címzés.)
- Az IP datagramok a legközelebbi célhoz fognak megérkezni.

Az Anycast címzés használata a gyakorlatban:

- DNS kiszolgálók esetén: ugyanolyan nevű (IP-című) legfelső szintű névkiszolgáló a világ több pontján is létezik. A kliensek a legközelebbit érik el.
- Az *IPv4-ről IPv6-ra való átállás* (IPv6 transition) során használt 6to4 megoldás is ezt használja a legközelebbi 6to4 átjáró elérésére.

[3] Ami meglehetősen tipikus, hiszen egy interfésznek mindenképpen van egy link-lokális IPv6 címe, valamint célszerűen egy globális (vagy legalább egy ULA) IPv6 címe is. Sőt egy interfésznek több globális IPv6 címe is lehet.

- *Tartalomszolgáltató hálózat*ok (CDN: Content Delivery Network) is alkalmazzák abból a célból, hogy a kliens a legközelebbi szerverről tudja letölteni a médiatartalmat.



1. ábra: Címzési módszerek illusztrációja

Globális Unicast címek

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

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

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

Bár jelenleg az IANA még csak a 2000::/3 tartományból delegált a RIR-eknek, ez elvileg semmiben sem különbözik a többi globális unicast IPv6 címtartománytól.

A jelenleg érvényes szabvány az RFC 3587: „IPv6 Global Unicast Address Format”.

Általános felépítésük:

```
|         n bits         |   m bits   |        128-n-m bits         |
+------------------------+------------+-----------------------------+
| global routing prefix | subnet ID |        interface ID         |
+------------------------+------------+-----------------------------+
```

Ahol az egyes mezők:

- global routing prefix: hierarchikus felépítésű, az aggregációról a RIR-ek, LIR-ek és az ISP-k gondoskodnak, a site-ok az ISP-ktől kapják
- subnet ID: hierarchikus felépítésű, az aggregációról a site-ok adminisztrátorai gondoskodnak

- Interface ID: a hálózati interfészt azonosítja

Az IPv6 címzési architektúráját definiáló RFC 4291 szerint a 0000/3 tartomány kivételével az Interface ID mérete 64 bit. Így felépítésük az alábbi:

```
|          n bits         | 64-n bits |       64 bits               |
+-------------------------+-----------+-----------------------------+
| global routing prefix | subnet ID |       interface ID          |
+-------------------------+-----------+-----------------------------+
```

Ha a jelenleg használt 2000::/3 tartományban az egyes szervezetek a már elavult (obsolete) RFC 3177 által ajánlott /48-as tartományokat kaptak, akkor a címeik felépítése a következő:

```
| 3 |     45 bits       |  16 bits  |       64 bits               |
+---+-------------------+-----------+-----------------------------+
|001|global routing pre.| subnet ID |       interface ID          |
+---+-------------------+-----------+-----------------------------+
```

Például a BME tartománya is ilyen méretű, a prefixe: 2001:738:2001::/48

A címek szerkezetét korábban az RFC 2374 írta le „An IPv6 Aggregatable Global Unicast Address Format” címmel. Erről fontos tudni, hogy *elavult* (obsolete), ezért nem használjuk.

A címallokáció megfontolásai

Az *elavult* (obsolete) RFC 3177 szerint a site-ok számára az alapértelmezett allokációs egység mérete a /48 volt. Az új irányelvet az RFC 6177 „IPv6 Address Assignment to End Sites” fogalmazza meg. Ezt röviden az alábbiakban foglaljuk össze:

- Kis felhasználóknál nem szükséges a pazarló /48 méret.
- A /64 viszont nem elég, mert idővel több fizikai hálózatra lehet szükség.
- Szempont a takarékosság is, de az is, hogy később se legyen szükség NAT-ra (a site kapjon elegendő címet).
- A reverse DNS feloldás szempontjából megfontolandó a 4 bites határra való illeszkedés.
- Megemlíti az egyes RIR-ek gyakorlatát (/56-os méret), de alapelv, hogy nincs új alapértelmezett méret.
- Bár a /128 bizonyos esetekben elfogadott, site esetén semmiképpen sem javasolt.

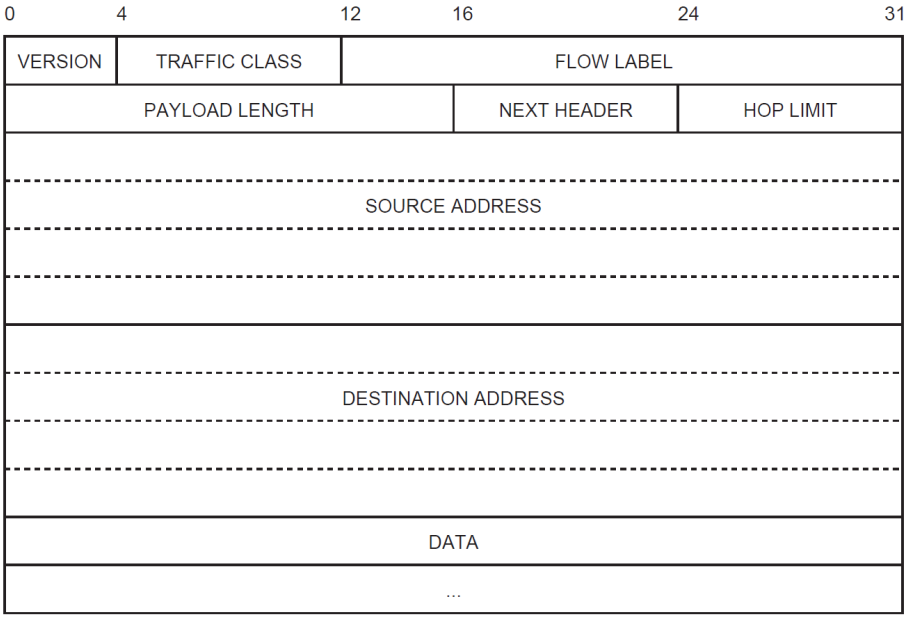
1.2. Az IPv6 datagram felépítése

Az IP 6-os verziójában a címek méretének növekedése mellett a másik jelentős változás a 4-es verzióhoz képest az, hogy jó néhány mező hiányzik. Így például a kötelező, fix fejrészből teljesen hiányoznak a tördelést szolgáló mezők, a tördelés lehetőségét a hálózati átvitel közben teljesen meg is szüntették (csak a forrás tördelhet, lásd a fejrész kiterjesztéseknél). Ugyancsak hiányzik az ellenőrző összeg, erre a mai átviteli megoldások mellett nincs igazán szükség, és így nem lassítja feleslegesen a routereket. A kötelező fejrész hossza fix, így annak hosszát sem kell megadni. Opciók ettől még lehetségesek, ezt ügyes trükkel oldották meg (lásd: next header). Ennyi bevezető után nézzük meg egy, csak a kötelező fejrészt tartalmazó IPv6 datagram felépítését (2. ábra).

Az IPv6 datagram mezőinek jelentése:

- **Version** (4 bit) – verziószám – Értéke természetesen: 6, bár érdemben nem használjuk, lásd később.
- **Traffic Class** (8 bit) – forgalmi osztály – Az IPv4 *Type of Service* mező utódja. Az RFC 2474 szerinti *Differentiated Services* megoldást használjuk.
- **Flow Label** (20 bit) – folyamcímke – Adott forrástól adott cél IP-cím (utóbbi multicast vagy anycast is lehet) felé irányuló forgalmon belül *folyam*ok (flow) elkülönítését és kezelését hivatott támogatni. Jelenleg érvényes specifikációja: RFC 6437.

- **Payload Length** (16 bit) – adatmező hossza – Mivel 16 bit hosszú, így értéke legfeljebb 65535 lehet. Az IPv4-gyel ellentétben itt a fejrész 40 oktettje nem számít bele, ezt fejezi ki a neve is: „hasznos teher hossza”.
- **Next Header** (8 bit) – következő fejrész – Megmutatja, hogy mit hordoz az IP címek utáni rész. Ez a mező egyrészt tartalmazhatja az IPv6 fölötti protokoll típusának megadását (ilyen értelemben az IPv4 *Protocol* mezőjének utódja), másrészt ezzel ki lehet terjeszteni a fejrészt, ahova további mezők kerülhetnek (ilyen értelemben az IPv4 *Options* mezőjét is helyettesíti). Ez utóbbi esetben a *fő fejrész* (a címekkel befejeződő 40 oktett) után jön egy *következő fejrész*.
- **Hop Limit** (8 bit) – átugrás korlát – Funkciója megfelel az IPv4 *Time To Live* mezőjének, de az elnevezése jobban kifejezi a funkcióját, és a mértékegysége sem másodperc (hanem darab).
- **Source Address** (128 bit) – forrás IPv6-os címe – Az IPv6 címekkel már részletesen foglalkoztunk.
- **Destination Address** (128 bit) – cél IPv6-os címe



2. ábra: IPv6 datagram felépítése

Megjegyzések:

1. Elvileg a *verziószám*ból derülne ki, hogy az utána következő mezőket nem IPv4, hanem IPv6 mezőknek kell tekinteni. Gyakorlatilag nem így történik, mert már a *hordozóhálózat*ban (link layer) az IPv4-től (0x0800) eltérő EtherType protokollazonosítót használnak az IPv6-ra (0x86DD).
2. Egy folyam klasszikus definíciója például a következő öt számmal adható meg: forrás és cél IP-címek, az IP fölötti protokoll száma (TCP vagy UDP), valamint a forrás és cél portszámok. A folyamcímke lehetőséget ad arra, hogy a folyamot sokkal egyszerűbben, tisztán az IPv6 protokoll mezői alapján azonosítsuk. (Esetünkben a két IP-cím és a folyamcímke egyértelműen azonosítják a folyamot.)
3. Az adatmező hosszát alapesetben a 16 bites érték erősen korlátozza, de lehetőségünk van úgynevezett jumbogram kialakítására is a Jumbo Payload option segítségével (RFC 2675). Ez 64kB-nál jóval nagyobb csomag is lehet, így a routereknek nem kell 64kB adatonként a fejrészt feldolgozniuk, ezáltal jelentősen gyorsulhat az adatátvitel. Ugyanakkor a

nagyméretű csomag sokáig foglalja a linket, ami szolgáltatásminőségi (QoS) problémákat vethet fel.

1.2.1. Az IPv6 fejrész kiterjesztése

A fejrész kiterjesztése nagyon szellemes megoldás[4]: mivel a next header mező mérete 8 bit, és az IP fölötti protokollok száma messze elmarad a 8 biten kifejezhető 256-tól, ezért jó néhány értéket más célra használhatunk: ezekkel különféle fejrész kiterjesztéseket adhatunk meg. Közülük néhányat már az RFC 2460-ban definiáltak:

- **Hop-by-Hop Options** (0) – Ha van ilyen, akkor ennek kell a legelsőnek lennie. Amint a nevéből is kiderül, ezt minden egyes routernek (hop) meg kell vizsgálnia (míg az egyebeket általában csak a célnál).
- **Routing** (43) – Eredetileg a 0-s típusú változatát definiálták (Type mező értéke 0, röviden RH0-nak is nevezik), ami az IPv4 *Loose Source and Record Route* opciójának a megfelelője volt, segítségével olyan csomópontok IP-címeit lehetett megadni, amin a csomagnak keresztül kell haladnia. Mivel ez kiváló lehetőséget nyújtott hálózati torlódást előidéző, tulajdonképpen *szolgáltatásmegtagadás* (Denial of Service, DoS) típusú támadásra, ezért érvénytelenítették (RFC 5095). Mobil IPv6-hoz a 2-es típusa használható.
- **Fragment** (44) – Csomagok tördelésére lehet használni, de erre IPv6-ban kizárólag a forrásnál van lehetőség. A tördelés megvalósítására használt mezők teljesen hasonlóak, mint IPv4 esetén, mélyebben nem foglalkozunk vele.
- **Destination Options** (60) – Az általa szállított információt csak a cél csomópontban kell megvizsgálni. (Multicast esetén több cél is lehet.)

A fentieken túl még két fontos fejrészkiterjesztést említünk meg:

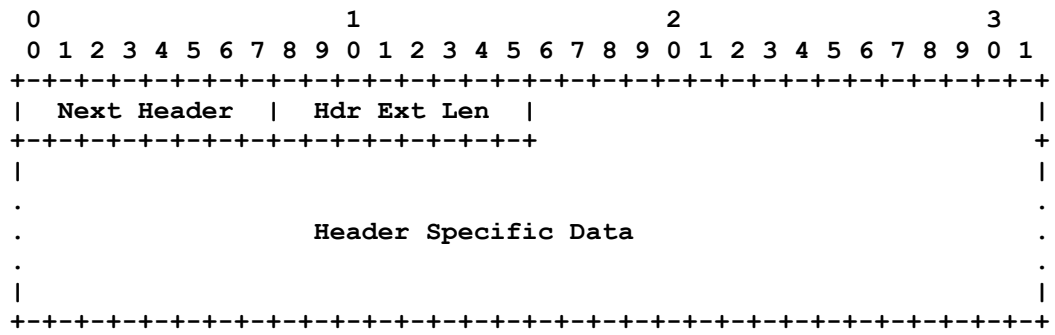
- **Authentication** (51) – Az adatok *eredet*ét (origin) és *változatlanság*át (integrity) igazolja és *visszajátszás* (replay) ellen is véd (RFC 4302).
- **Encapsulating Security Payload** (50) – Az adatok *titkosság*ának (confidentality), *változatlanság*ának (integrity) és (áttételesen) *eredet*ének (origin) védelmére használható (RFC 4303).

Megjegyezzük, hogy az utolsó kettőt használó IPsec (RFC 4301) eredetileg minden teljes IPv6 implementáció *kötelező* (MUST) része volt, de az RFC 6434 *erősen ajánlott*á (SHOULD) minősítette át azzal az indoklással, hogy vannak más megoldások is (például TLS, SSH), és az IPsec nem minden esetben az ideális megoldás.

Az egyes opciók szerkezetét, működését mélyebben nem vizsgáljuk, annyit jegyzünk meg róluk, hogy sorrendjük (közelítőleg) kötött, méretük oktettben kifejezve mindig 8-cal osztható, és az első oktettjük mindig a következő kiterjesztés vagy az IP fölötti protokoll azonosítója. (Vegyük észre, hogy IPv4 esetén 32-bites, IPv6 esetén pedig 64-bites egységekben gondolkozunk.)

Később újabb kiterjesztéseket is definiáltak, sőt a fejrész kiterjesztés általános formátumát is definiálták az RFC 6564-ben az alábbiak szerint (de ez csak az későbbiekre kötelező):

[4] Ugyanakkor számos kockázattal is jár. Növeli az útválasztók terhelését, problémát okozhat a tűzfalaknál, különféle támadásokra ad lehetőséget, például az RFC 7113 2.1. pontjában is található ilyen.



A fejrész kiterjesztés mezőinek jelentése:

- **Next header** (8 bit) – következő fejrész – A közvetlenül utána következő fejrészt azonosítja. Ez lehet további fejrész kiterjesztés vagy valamely az IP fölötti protokoll azonosítója.
- **Hdr Ext Len** (8 bit) – fejrész kiterjesztés hossza – Ez a 8 bites előjel nélküli egész szám az adott fejrész kiterjesztés méretét adja meg, 8 oktettes egységekben mérve úgy, hogy az első 8 oktettet nem számítjuk bele.
- **Header Specific Data** (változó hosszban) – kiterjesztés specifikus adatok – Kiterjesztéstől függően más-más adatok.

1.3. Internet Control Message Protocol version 6 (ICMPv6)

Az ICMP IPv6-os implementációja. Üzenetei az IPv6 felett utaznak (next header=0x3A, decimálisan 58), de minden IPv6 implementáció kötelező része! Aktuális dokumentáció: RFC 4443.

Üzenetei két típusba sorolhatók: hibaüzenetek és információs üzenetek.

Az ICMPv6 üzenetek formátuma egyedi. Ami közös bennük, az az ICMPv6 fejrész első 32 bitjén található három adatmező:

- **Type** (8 bit) – típus – 0-127: hibaüzenetek, 128-255: információs üzenetek
- **Code** (8 bit) – altípus – a típuson belül további fajtát jelölhet, ha egyáltalán van ilyen
- **Checksum** (16 bit) – ellenőrző összeg

A további rész kiosztása függ az üzenet típusától.

Az alábbiakban felsoroljuk a fontosabb ICMPv6 üzeneteket, az első szám a típus mező értéke.

- **1 – Destination Unreachable** – cél nem elérhető – Mint az azonos nevű ICMP üzenet.
- **2 – Packet Too Big** – a csomag mérete túl nagy – Az IPv6 útközben nem tördel. Ha egy csomag nem fér bele az MTU-ba, akkor a router eldobja, és visszajelzést küld.
- **3 – Time Exceeded** – időtúllépés – Mint az azonos nevű ICMP üzenet.
- **4 – Parameter Problem** – paraméter értelmezési hiba – A hiba okát a Code mezőben jelzi:
  - 0: Hibás IPv6 fejrész mező,
  - 1: Ismeretlen Next Header típus,
  - 2: Ismeretlen IPv6 opció.
- **128 – Echo Request** – visszhang kérés – Mint az azonos nevű ICMP üzenet.
- **129 – Echo Reply** – visszhang válasz – Mint az azonos nevű ICMP üzenet.
- **130 – Multicast Listener Query** – csoporttagok lekérdezése – Multicast cím megadása nélkül: mely multicast csoportoknak vannak tagjai az adott hálózaton? Multicast cím megadásával: a megadott című multicast csoportnak vannak-e tagjai az adott hálózaton? MLDv2 esetén pedig még forrás specifikus lekérdezésre is lehetőséget nyújt.
- **131 – Multicast Listener Report** (MLDv1) – csoporttagság jelzése – A csoporttagok ezzel az üzenettel jelzik igényüket a multicast forgalomra. (MLDv2-ben a 143-as típusú Multicast Listener Report vette át a funckióját!)
- **132 – Multicast Listener Done** (MLDv1) – multicast csoportból kilépés – A csoporttagok ezzel az üzenettel jelzik, hogy már nem tartanak igényt az adott multicast csoport

forgalmára. (MLDv2-ben a 143-as kódú Multicast Listener Report vette át a funkcióját, ami így kétféle funkciót is megvalósít.)

- **133 – Router Solicitation** – router információ kérése – Az NDP része, részletesen tárgyaljuk majd.
- **134 – Router Advertisement** – router információ hirdetése – Az NDP része, lehet kérésre válasz, de a routerek kérés nélkül is hirdetik. A kéretlen hirdetés szándékosan nem pontosan periodikus.
- **135 – Neighbor Solicitation** – MAC-cím kérése – Az NDP része, az ARP megfelelője (pl. ARP Request és Probe funkciók).
- **136 – Neighbor Advertisement** – MAC-cím hirdetése – Lehet kérésre válasz, ekkor Solicited Neighbor Advertisement. Kérés nélkül is küldhető (pl. IP-cím változásakor), ez az Unsolicited Neighbor Advertisement.
- **137 – Redirect** – jobb útvonal közlése – Mint az azonos nevű ICMP üzenet.
- **143 – Multicast Listener Report** (MLDv2) – csoporttagság jelzése – Az MLDv2 protokoll esetén az MLDv1 131-es és 132-es típusú üzenetének funkcióját is ellátja.

1.4. A Neighbor Discovery Protocol (NDP)

Az NDP (Neighbor Discovery Protocol, RFC 4861) a TCP/IP referenciamodell internet rétegében működik. Működéséhez ICMPv6 protokoll üzeneteket használ, azaz az IPv6 fejrészben a *következő fejrész* (next header) mező értéke 58, és aztán az ICMPv6 fejrész *típus* (type) mezője azonosítja a konkrét üzentet (például: Neigbor Solicitation, Neighbor Advertisement).

Egy host a működéséhez szükséges azonosítókhoz, paraméterekhez különféle módon juthat hozzá: beállíthatók statikusan, megszerezhetők az állapotmentes automatikus címkonfiguráció (SLAAC, lásd lent), valamint a DHCPv6 segítségével.

Az NDP többek között képes:

- Állapotmentes automatikus címkonfigurációra
- Címfeloldásra (IPv6 címből MAC cím)
- Routerek címének megállapítására
- DNS szerverek címének megállapítására
- Címduplikáció ellenőrzésére, azaz annak vizsgálatára, hogy egy IPv6 címet már használnak-e (DAD: Duplicate Address Detection)
- Az adott linken érvényes prefixek és MTU kiderítésére

Közülük néhányat a továbbiakban részletesen bemutatunk.

1.4.1. Állapotmentes automatikus címkonfiguráció

Az állapotmentes automatikus címkonfiguráció (SLAAC: Stateless Address Autoconfiguration, RFC 4862) lehetővé teszi, hogy a számítógépek (host) emberi beavatkozás nélkül kapjanak IPv6 címet állapot alapú kiszolgáló (DHCPv6) nélkül is. A folyamat több lépésből áll, a host először link-lokális IPv6 címet hoz létre, majd annak segítségével szerez egy routertől prefixet a globálisan egyedi IPv6 címhez. Mind a link-lokális, mind a globális IPv6 cím esetén az IPv6 cím utolsó 64 bitjében található Interface ID-t a hálózati interfész MAC-címéből állítja elő a *módosított EUI-64 cím*et előállító algoritmussal (lásd lent). Használat előtt mindkét IPv6 cím egyediségét ellenőrzi a DAD (Duplicate Address Detection) algoritmussal (külön tárgyaljuk).

Az SLAAC lépései:

- Link-lokális IPv6 cím generálása (fe80::/64 + módosított EUI-64 azonosító)
- Link-lokális IPv6 cím egyediségének ellenőrzése (DAD)
- Hálózati prefix kérése (ICMPv6 Router Solicitation)
    - Az üzenet küldéséhez forráscímként már használható a link-lokális IPv6-cím
    - A célcím pedig a link-lokális hatókörű (link-local scope) all routers IPv6 multicast cím (ff02::2)

- Hálózati prefix információ vétele (ICMPv6 Router Advertisement)
  - A router ezt a saját link-lokális IPv6 címéről általában a link-local scope all nodes IPv6 multicast címre (ff02::1) küldi. (De az RFC 4861 6.2.6. pontja szerint küldheti a host unicast címére is, ha az nem az unspecified (::) IPv6 cím.)
- Globálisan egyedi IPv6 cím (global unicast IPv6 address) előállítása (a kapott prefix + módosított EUI-64 azonosító). (Ehhez /64 hosszúságú kapott prefix szükséges!)
- Globálisan egyedi IPv6 cím egyediségének ellenőrzése (DAD)

Figyeljük meg, hogy a link-lokális és a globális unicast címek utolsó 64 bitje azonos lesz, hiszen mindegyikben a módosított EUI-64 azonosító szerepel.

Az SLAAC biztonsági kockázattal jár: hamis Router Advertisement üzenettel a kliens megtéveszthető, lásd:

- SLAAC Attack http://resources.infosecinstitute.com/slaac-attack/
- RFC 6104: „Rogue IPv6 Router Advertisement Problem Statement”

A módosított EUI-64 címet előállító algoritmus

Célja a hálózati interfész 48 bites MAC címéből 64 bites EUI (Extended Unique Identifier) azonosító létrehozása. Az algoritmus lépései:

- A 48 bites címet középen kettévágva a két fél közé beszúrjuk az FFFE 16 bites értéket.
- A MAC cím első bájtjának második legkisebb (azaz 2-es) helyiértékű bitjét 1-re állítjuk. (Ez a MAC cím OUI részének U/L bitje, tehát azt jelezzük vele, hogy nem univerzálisan egyedi, hanem lokálisan adminisztrált címről van szó.[5])

Az algoritmus működését egy példával illusztráljuk:

- Az eredeti MAC cím: 00:C1:C0:0B:0C:0D
- Az első lépés eredménye: 00:C1:C0:**FF:FE**:0B:0C:0D
- A második lépés eredménye: 0**2**:C1:C0:FF:FE:0B:0C:0D (kész)
- Az IPv6-nál használt alakban: 2C1:C0FF:FE0B:C0D

A kapott EUI-64 azonosítót az IPv6-nál használt alakban egy 64 bites prefix után írva megkapjuk az IPv6 címet.

Adatvédelmi megfontolás

Amennyiben egy állomás az IPv6 címét az SLAAC algoritmus segítségével, valamely hálózati interfész MAC címének felhasználásával állítja elő, akkor az állomás tevékenysége nyomon követhetővé válik. Ezt a problémát hivatott orvosolni az RFC 4941. A megoldás lényege, hogy a MAC-címen alapuló globális IPv6 címen kívül létrehoznak egy másik, időben változó globális IPv6 címet is. A Windows 7 operációs rendszer ezt „Temporary IPv6 Address”-nek nevezi.

1.4.2. IPv6-cím egyediségének ellenőrzése

Az IPv6 *cím egyediségének ellenőrzése* (DAD: Duplicate Address Detection) elvi szinten hasonlóan történik, mint IPv4-nél az ARP Probe üzenettel. IPv6 esetén az ICMPv6 Neighbor Solicitation (NS) üzenetet használják, és ha nem érkezik rá válasz, akkor a címet még senki sem használja.

A megvalósítás viszont bonyolultabb. Ugyanis míg az NS üzenetben a forráscím (az ARP-hez hasonlóan) érvénytelen (::/128), addig a célcím a *vizsgált IPv6 címhez* (tentative address) tartozó *solicited-node multicast address*. Ennek haszna nyilvánvaló: míg IPv4 esetén az ARP *üzenetszórás*t (broadcast) használ, és így az adott *üzenetszórási tartomány* (broadcast domain) összes gépét terheli, addig az ND multicast használatával megkíméli a gépek döntő többségét. Az adott multicast csoportban elvileg többen is tartózkodhatnak, de SLAAC esetén ezek száma várhatóan alacsony. (Mivel a módosított EUI-64 azonosító utolsó 24 bitje alapján képezik a csoportcímet, ezért nem zárható ki, hogy egy adott hálózaton különböző gyártóktól származó hálózati interfészek esetén ez a 24 bit azonos legyen, de várhatóan nem lesz belőlük túlságosan sok.)

[5] Fontos, hogy ez a kitétel csak a MAC címre vonatkozik, ettől még a generált IPv6 cím lehet link lokális és globális is.

Megjegyzés: Azt már láttuk, hogy az Ethernet szinten a solicited node multicasthoz használt csoportcím prefixe: „33:33:ff”. Amennyiben az IPv6 címet SLAAC használatával hoztuk létre, akkor az utolsó 24 bitje pedig a módosított EUI-64 azonosító utolsó 24 bitje, ami viszont nem más, mint a hálózati interfész MAC címének az utolsó 24 bitje.

A DAD során a vizsgált címet használni kívánó interfészének még a fenti NS üzenet kiküldése előtt csatlakoznia kell két multicast csoporthoz, ezek az all-nodes multicast (azért, hogy ha valaki már használja a vizsgált címet, akkor megkapja a Neighbor Advertisement üzenetet) és a vizsgált címhez tartozó solicited-node multicast (azért, hogy ha valaki más is szeretné használni a vizsgált címet, akkor hallják egymást). Az RFC még számos egyéb feltételt és részletet leír (például a különböző véletlenszerűen megválasztott késleltetésekkel kapcsolatban), de ilyen szintű részletekbe most nem megyünk bele (a részletek megtalálhatók az RFC 4862 5.4. részében).

Végül, ha válaszként Neighbor Advertisement üzenet érkezik, akkor a vizsgált cím nem egyedi, tehát tilos (MUST NOT) az interfészhez rendelni (használni)[6] és erősen ajánlott (SHOULD) az eseményt naplózni.

1.4.3. Címfeloldás

Bár a számítógépek IP-címmel azonosítják egymást, egy csomag tényleges (fizikai) elküldéséhez szükség van annak az interfésznek az adatkapcsolati szintű (link layer) címére (MAC-címnek is szoktuk nevezni) amely számára közvetlenül (link szinten) küldeni szeretnénk a csomagot szállító adatkapcsolati szintű adategységet, pl. Ethernet keretet. Az IP 4-es verziója erre a célra a jól ismert ARP-t alkalmazza. IPv6 esetén pedig a Neighbor Solicitation (NS) és Neighbor Advertisement (NA) ICMPv6 üzenteteket használjuk. (Az IPv6 címfeloldás részletes leírása az RFC 4861 7.2. részében található.)

Amikor egy (multicast képes) hálózati interfészt engedélyeznek, az köteles (MUST) csatlakozni az all-nodes multicast csoporthoz, valamint az összes IPv6 címéhez tartozó solicited-nodes multicast address által meghatározott csoporthoz is. Ha az IPv6 címe megváltozik, akkor köteles (MUST) tagja lenni az új IPv6 címéhez tartozó solicited-nodes multicast address cím által meghatározott csoportnak, illetve elhagyni az olyan csoportot, amiben már nem illetékes. Megjegyzés: amint már korábban említettük, lehetséges, hogy több különböző IPv6 címhez ugyanaz a csoportcím tartozik. (Például tipikusan ez a helyzet, ha egy fizikai interfészen többféle prefix alapján képzett interfésze is van, és ezek utolsó 64 bitjét a hálózati interfész azonosítójából képezték.) Így nem minden esetben történik ténylegesen csoportba való belépés illetve onnan való kilépés. Amennyiben ilyen történik akkor erre az MLD (Multicast Listener Discovery) protokoll 1-es vagy 2-es verzióját használják.

Az RFC számos lehetőséget leír, hogy NS/NA üzenetváltáskor milyen címeket használhatnak, itt most egy tipikus példát mutatunk be. A kérdező gép az NS üzenetet a saját link-lokális IPv6 címéről a kérdéses IPv6 címhez tartozó solicited-node multicast címre küldi, és az üzenet *Target Address* mezőjében benne van a kérdéses IPv6 cím, valamint a kérdező a *Source link-layer address* opcióval megadja a saját MAC címét is. Az IPv6 cím gazdája egy NA üzenettel válaszol, amit a saját link-lokális IPv6 címéről a kérdező link-lokális IPv6 címére küld, benne van a *Target Address* mezőben a kérdéses IPv6 cím, és a hozzá tartozó MAC cím pedig a *Target link-layer address* opcióban található.

1.5. Multicast Listener Discovery (MLD)

Amint a címzésnél már láttuk, az IPv6 kifinomultan támogatja a *többesküldés* (multicast) megvalósítását, és amint az ND-nél láttuk, az IPv6 alaposan ki is használja a multicast lehetőségeit, kiváltva vele a broadcastot.

[6] A 3. mérés 2015. évi tapasztalatai alapján ezt a szabályt a laborban használt Windows 7 operációs rendszer nem tartja be. Ezért különösen fontos, hogy a hallgatók a mérést figyelmesen végezzék. (Több ízben előfordult, hogy valamely hallgató figyelmetlenségből a **`komhal.kl.bme.hu`** szerver IPv6 címét állította be a saját gépe IPv6 címeként. Sajnos a beállítás sikeres volt, és utána a szerver az összes hallgató számára elérhetetlenné vált. A hibára első esetben a szerver naplójának vizsgálatából jöttünk rá.)

IPv6-ban az egyes állomások az MLD (Multicast Listener Discovery, MLDv1: RFC 2710, MLDv2: RFC 3810) protokoll segítségével tudják kifejezni igényüket valamely multicast csoport forgalmára. (Az MLDv1 az IGMPv2-nek, az MLDv2 pedig az IGMPv3-nak az IPv6-os megfelelője. Az újabb verzió célja mindkét esetben az volt, hogy a vevők kifejezhessék azon igényüket, hogy az adott multicast csoport forgalmából csak egy adott forrás adására tartanak / nem tartanak igényt.)

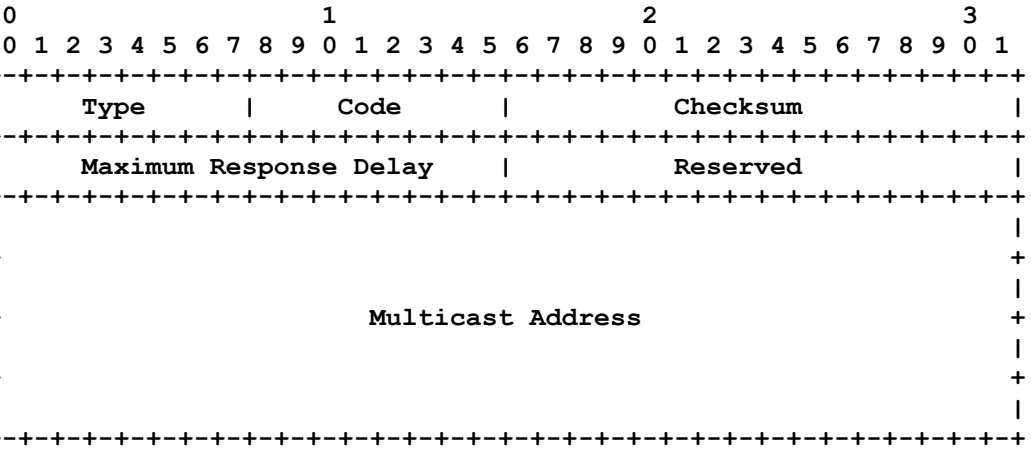
Az MLD ICMPv6 üzeneteket használ, azaz az IPv6 fejrészben a *következő fejrész* (next header) mező értéke 58, és aztán az ICMPv6 fejrész *típus* (type) mezője azonosítja a konkrét üzentet. Ezek a v2-ben a v1-hez képest változtak, az 1. táblázatban röviden összefoglaljuk, hogy melyik verzió milyen célra milyen üzeneteket használ. Az egyes üzenetek után megadjuk az ICMPv6 típus mező decimális értékét is, amire a különböző verziószámú, ezért eltérő Multicast Listener Report üzenetek azonosításához van szükség.

verzió	MLDv1	MLDv2
csoporttagság lekérdezése	Multicast Listener Query (130)	Multicast Listener Query (130)
csoporttagság jelzése	Multicast Listener Report (131)	Multicast Listener Report (143)
csoportból kilépés	Multicast Listener Done (132)	Multicast Listener Report (143)

1. táblázat: Az MLDv1 és az MLDv2 által használt ICMPv6 üzentek típusa

Tehát a 2. verzióban a csoporttagság jelentésére és a csoportból való kilépés (kérdés nélküli) bejelentésére ugyanazt az üzenetet használják, amely azonban eltér az 1. verzióban a csoporttagság jelzésére használt üzenettől. A táblázatból nem látszik, de a 2. verzióban a lekérdezésre használt üzenet formátuma is változott (bővült).

Az MLDv1-ben mindhárom üzenet formátuma azonos:



Ahol az egyes mezők jelentése:

- **Type** (8 bit) – típus – Lehetséges értékei: 130, 131, 132. Az üzenet típusát határozza meg.
- **Code** (8 bit) – nem használják – Csak az ICMPv6 fejrész szerkezete miatt szerepel. A küldő az értékét 0-ra állítja, a vevő pedig nem veszi figyelembe.
- **Checksum** (16 bit) – ellenőrző összeg – A szabványos ICMPv6 ellenőrző összeg.
- **Maximum Response Delay** (16 bit) – maximális megengedett késleltetés – Csak a kérdésben (query) használják, értéke ms-ban adja meg, hogy mennyi időn belül kell a választ küldeni. A többi üzenetben az értékét 0-ra állítja a küldő és a vevő nem veszi figyelembe.
- **Reserved** (16 bit) – további fejlesztésre fenntartva – Értékét a küldő 0-ra állítja és a vevő nem veszi figyelembe.
- **Multicast Address** (128 bit) – multicast cím – Lekérdezés (query) esetén két lehetőség van. Ha az értéke 0, akkor *általános lekérdezés*ről (General Query) van szó, a kérdező arra kíváncsi, hogy mely csoportoknak vannak tagjai. Ha értéke 0-tól különböző, akkor

*csoportcím specifikus lekérdezés*ről (Multicast-Address-Specific Query) van szó, a kérdező azt szeretné megtudni, hogy az adott című IPv6 multicast csoportnak van-e tagja. Válasz (Report) vagy kilépés (Done) esetén pedig azt a csoportot adja meg, amelyik forgalmára a válasz küldője az igényét bejelenti (Report) vagy éppen amelyiket el kívánja hagyni (Done).

A 2. verzió az 1. verzióval képes együttműködni (interoperable), a bővítés célja a *forrásspecifikus multicast* (SSM: Source-Specific Multicast, RFC 3569) támogatása. A lekérdezés (Query) üzenet típusa (type) nem változik, de számos további mezővel bővül, valamint a benne található Maximum Response Delay mező neve *Maximum Respons Code*ra változik, miközben értelmezése 0-32767 értékek esetén változatlan, fölötte viszont lebegőpontos számként kell értelmezni (lásd: RFC 3810 5.1.3 rész). A válaszüzenet típuskódja és formátuma is teljesen megváltozik. Ezeket a továbbiakban nem részletezzük. (A gyakorlatban találkozhatunk velük, a Wireshark képes dekódolni.)

1.6. Path MTU discovery (PMTUD)

A hálózati átvitel hatékonysága szempontjából kívánatos, hogy a lehető legnagyobb csomagméretet használjuk. Egy adott *útvonal legnagyobb megengedett csomagméretének kiderítése* (PMTUD: Path MTU Discovery) mind IPv4 (RFC 1191), mind IPv6 (RFC 1981) esetén felmerülő kérdés. Míg IPv4 esetén egy adott útvonalon a közbülső routerek tördelhetnek, ha a DF (Don't Fragment) bit nincs beállítva, így IPv4-nél túl nagy csomagméret esetén „csak" a hatékonyság csökken (esetleg a csomagok összerakásánál lehet probléma, ha pl. egy tűzfal eldobja a töredékeket), addig IPv6 esetén kizárólag a forrás tördelhet, tehát a csomagméret megválasztása itt még fontosabb kérdés. Amennyiben IPv6 esetén egy csomag mérete túl nagy, és ezért egy közbülső router eldobja, akkor az adott router ezt a problémát a Packet Too Big (PTB) ICMPv6 üzenettel *köteles* (MUST) jelezni a forrás számára (lásd RFC 1885, 3.2). A PMTUD (első körben) ezen üzenetek használatán alapul.

Mivel az útvonalak időközben megváltozhatnak, a PMTUD-t időről időre újra el kell végezni. Amennyiben egy host PTB üzenetet kap, akkor *köteles* (MUST) az adott útvonalra küldött üzenetek méretét csökkenteni, és a közeljövőben (10 perc a szokásos idő, 5 percnél semmiképpen sem lehet kisebb) nem is küldhet ilyen méretű csomagot. (Az RFC a csökkentés konkrét mértékét nem határozza meg.) Egy adott útvonal MTU értékének csökkenésére tehát azonnal kell reagálni, az esetleges növekedést pedig 10 percenként ajánlott megvizsgálni.

Azonban előfordulhat, hogy valamely útvonalon egy tűzfal nem engedi át a PTB üzeneteket. Ez a gyakorlatban például azt jelentheti, hogy egy TCP kapcsolat felépül ugyan, de aztán a forgalom nem halad át rajta. ICMPv6 üzenetek hiányában történő PMTUD-re ad megoldást az RFC 4821. A benne leírt PLPMTUD (Packet Layer PMDUD) alapötlete (TCP-ben gondolkozva) az, hogy a TCP szegmens méretet kis értékről kezdve növeli, és az első izolált csomagvesztést nem torlódásnak tekinti (nem csökkenti a congestion window értékét), hanem azt feltételezi, hogy a szegmens mérete túl nagy volt, ezért eldobásra került, de a PTB üzenet elveszett. A megoldással bővebben nem tudunk foglalkozni, csak jeleztük a probléma és a megoldás létét.

Mivel az 1280-as MTU-t minden IPv6 node-nak kötelezően támogatnia kell, és Ethernet esetén 1500 bájt az általános MTU, ezért a mindennapi életben az esetek többségében az 1280-1500 az MTU lehetséges intervalluma. Speciális esetben lehet komoly jelentősége a nagyobb MTU használatának.

2. IPv6 áttérési technológiák összehasonlító elemzése

2.1. Általános áttekintés

2.1.1. Az IPv4-ről IPv6-ra való áttérés időbeli lezajlása

Az Internet történelmében egyszer sikerült a pillanatszerű protokollváltás: az ARPANET-en 1983. 01. 01-én volt az áttérés NCP-ről (Network Control Program) TCP/IP-re (RFC 801). Ma ez lehetetlen feladat. Több milliárd csomópont van, lehetetlen őket egyszerre „átkapcsolni". Sőt jelenleg rengeteg hardver és szoftver eleve alkalmatlan IPv6-ra.

Bár az átmenet régen megkezdődött, először nagyon lassan haladt, és csak az IPv4 címek kifogyása adott neki lendületet. A két protokoll tartósan egymás mellett fog élni a fentieken túl azért is, mert egyrészt bizonyos hardver és szoftver szállítók nem fogják megoldani az IPv6 kompatibilitást, másrészt a felhasználók ragaszkodnak a régi eszközeikhez. Tehát meg kell oldani az IPv4 és az IPv6 alapú rendszerek együttműködését!

2.1.2. Az IPv4 és IPv6 alapú rendszerek együttműködésének fontosabb esetei

A legtöbb alkalmazásunk kliens-szerver konfigurációban működik. Egyik szempont, hogy mely protokollokra képes a kliens és a szerver. A másik szempont, hogy a hálózat mely protokollokra képes a klienstől a szerverig terjedő úton. Ezen szempontok szerint vizsgálva a következő eseteket és megoldásokat tartjuk említésre érdemesnek:

Az egyszerű eset: dual stack használata

Ha a kliens és a szerver közül bármelyik is képes mindkét protokoll használatára (*dual stack*), akkor a „közös nyelv" használatával a kommunikáció megoldott – feltéve, hogy a köztük levő hálózat is támogatja azt. A probléma az, hogy már nincs elég IPv4 cím!

IPv6 képes kliens IPv4-only környezetben és IPv6 szerver

A kliens képes IPv6-ra, de az internetszolgáltató (ISP) csak IPv4-címet ad a kliensnek, a szerver pedig kizárólag IPv6-ra képes. Egy régi és viszonylag működőképes, de számos problémával járó megoldás: *6to4* használata. Ezt a megoldást idén még mélyebben is bemutatjuk, de néhány év múlva már nem. A 6to4-hez hasonló megoldás még a *teredo* és a *6rd*, röviden ezekkel is foglalkozunk.

IPv6 kliens és IPv4 szerver

Csak IPv6-ra képes a kliens (már csak IPv6 cím jutott neki) és csak IPv4-re képes a szerver (régi, az IPv6-ot nem támogatja). Egy jó megoldás: *DNS64* szolgáltatás + *NAT64* átjáró használata. Mindkét részével mélyebben foglalkozunk.

Az ISP a hálózatában csak IPv6-ot szeretne használni, de bizonyos alkalmazások csak IPv4-re képesek

Egy szolgáltatónak jelentős többletfeladatot (és így többletköltséget) okoz, ha mind az IPv4-et, mind az IPv6-ot támogatja a hálózatában (dual stack). Egyre több olyan szolgáltató van, amely már csak IPv6-ot szeretne használni. Viszont vannak olyan alkalmazások, amelyek csak IPv4-et tudnak használni. A problémára számos megoldás született, közülük az 5 legfontosabbat megismerjük.

IPv4 kliens és IPv6 szerver

Csak IPv4-re képes a kliens (régi hardver és/vagy szoftver) és csak IPv6-ra képes a szerver (ilyenek is vannak, és számuk várhatóan nőni fog). Egy megoldás lehetne (vagy inkább lehetett volna): *NAT46+DNS46*, de évek óta nem lett belőle szabvány, bár léteznek rá implementációk. Röviden ismertetjük a megoldás alapötletét.

IPv6 kliens és IPv6 szerver DE útközben egy szakaszon csak IPv4 van

Tipikus eset, hogy az új IPv6 „szigeteket" össze kell kötni. A megoldás az IPv6 datagramok szállítása IPv4 fölötti „alagútban" (6in4 tunnel).

IPv4 kliens és IPv4 szerver DE útközben egy szakaszon csak IPv6 van

Ma még nálunk nem igazán jellemző, de majd lehet. A megoldás az IPv4 datagramok szállítása IPv6 fölötti „alagútban" (4in6 tunnel).

2.2. Emlékeztető az IP-címek megosztásáról

Ennek a témakörnek (legalább részleges) ismeretét ugyan feltételezzük az Olvasóról, de a továbbiak megértéséhez szükséges néhány fontos részlet bemutatása és az egységes terminológia használata érdekében egy rövid összefoglalót adunk róla.

2.2.1. A címfordítást használó megoldás

A hálózati címfordítás (NAT/NAPT) működési elve

A TCP/IP protokollcsaládot eredetileg végponttól végpontig való kommunikációra tervezték. Az alkalmazások arra számítanak, hogy a címek és portszámok a hálózati átvitel során változatlanok.

Az IPv4-címek szűkössége miatt terjedt el az a megoldás, hogy egy hálózatban, ahol privát IP-címeket használnak, de szükség van külső kommunikációra is, a problémát címfordítással oldják meg.

Legyen a feladat először az, hogy a *privát IP-címmel rendelkező (kliens) gépek elérhessenek külső, publikus IP-címmel rendelkező (szerver) gépeket*. Ehhez rendelkezésünkre áll egy router, aminek van publikus IP-címe.

A megoldás alapötlete a következő:

- A privát IP-címmel rendelkező gépről a célcím alapján küldjük el a csomagot az Internet felé. (Ekkor a csomag elvileg ugyan odaérne, de a válasz nem találna vissza. Egyébként pedig tiltott a privát IP-címeknek a publikus környezetben történő használata.)
- A kimenő router cserélje ki a forrás privát IP-címét a saját publikus IP-címére. Így már a válasz visszaér a címcserét végrehajtó routerhez.
- A router továbbítsa a választ a megfelelő privát IP-című gépnek. – De hogyan?

Ahhoz, hogy a router a választ az eredeti feladónak vissza tudja küldeni, nyilván kell tartania, és egy válaszcsomag érkezésekor tudnia kell, hogy ki volt a küldője annak a csomagnak, amire a válasz érkezett. Ennek érdekében a nyilvántartáshoz az IP-címen túl mást is felhasznál. TCP és UDP esetén ezek a forrás portszámok. De a routernek nem elegendő ezeket megjegyeznie, hiszen a forrás portszámok csak *gépenként egyediek*, a forrás IP-címet viszont a sajátjára cseréli. Tehát a megoldás (az ún. *traditional NAT* esetén, de lásd majd később: *extended NAT*) a következő:

- A router a kimenő csomagokban a forrás IP-cím cseréjekor a forrás portszámokat is kicseréli a routeren egyedi portszámokra: az IP-címek és a célportszám mellett ezekkel már egyértelműen azonosítani tudja a kapcsolatokat. Kapcsolatonként nyilvántartja, hogy mit mire cserélt ki.
- A bejövő csomagoknál az IP-címen kívül a portszámot is vissza kell cserélnie. (Itt most az irány változása miatt a cél IP-cím és a célport az, amit átír.)

Megjegyzés: terminológia nem egységes, de eredetileg a NAT csak az IP-címek cseréjét jelentette, ezt hívják ma *basic NAT*-nak, vagy *one-to-one NAT*-nak. A fenti megoldás precíz neve a *NAPT* (Network Address and Port Translation). Nevezik *many-to-one NAT*-nak is.

A NAT célja és működése szempontjából a fent ismertetett megoldást *Source NAT*-nak (SNAT) hívjuk akkor, ha a router publikus IP-címe fix, és *Masquerade*-nek, ha DHCP-vel kapta az interfésze az IP-címet (ilyenkor nem egy fixen megadható IP-címre, hanem az interfész aktuális IP-címére kell a forrás IP-címet kicserélni).

A másik irányú feladat az, hogy *privát IP-címmel rendelkező gépeket elérhetővé tegyünk az Internet felől*. Erre a megoldás a *Destination NAT* (DNAT) vagy más néven *port forwarding*, ahol a router az adott portjára érkező csomagokat egy meghatározott privát IP-című gépnek továbbítja úgy, hogy a célcímet kicseréli a csomagban. Például a 80-as portra érkező csomagokat a 10.1.1.2 webszerver, a 25-ösre érkezőket pedig a 10.1.1.3 SMTP szerver felé továbbítja.

ICMP esetén nincs portszám, de segíthet az, hogy egy hibaüzenetben benne van az azt kiváltó TCP vagy UDP adategység első 64 bitje a portszámokkal. Amennyiben nem hibaüzenetről van szó, akkor is van megoldás: az ICMP üzenet valamilyen azonosító jellegű mezőjét használják fel.

A NAT-ról bővebben az RFC 3022-ben olvashatunk, az IP, TCP, UDP, ICMP fejrészek mezőinek módosításával a 4.1. rész, az ellenőrző összeg hatékony újraszámításával a 4.2. rész foglalkozik.

Lényegében arról van szó, hogy a régi portszámot "kivonjuk", az újat pedig "hozzáadjuk" és nem kell a számítást a teljes adategységre elvégezni.

A NAPT kritikája

A megoldás súlyos problémája, hogy sérti az IP-címek és portszámok végponttól végpontig történő változatlanságának elvét, ami számos következménnyel jár. Vizsgáljunk meg közülük néhányat:

- Koncepcionálisan az összes olyan alkalmazási rétegbeli protokollal probléma lehet, ahol az alkalmazási rétegbeli PDU-ban megjelenik valamilyen kommunikációazonosító (pl. IP-cím, portszám). Például egy FTP kliens aktív módban a vezérlő kapcsolaton keresztül megadja a szervernek, hogy mely portján várja, hogy a szerver felépítse az adatkapcsolatot. Privát IP címmel várhatja – hacsak nem segít valaki: *protocol helper*. (Az adott esetben persze tökéletes megoldás az FTP passzív módjának alkalmazása, amikor a vezérlő kapcsolaton keresztül a szerver küldi el a kliensnek az IP-cím + portszám párost, ahova a kliens gond nélkül fel tudja építeni az adatkapcsolatot. Ez a megoldás a kliens oldali tűzfal problémáját is megoldja, ami kifele engedni fogja a kapcsolat felépítését, míg befele nem engedné.) Hasonlóképpen egy IP-telefonrendszernél is gond lesz az IP-címek átküldésével (arra is van megoldás, létezik például SIP NAT helper is).
- A port forwarding ugyan alkalmas arra, hogy egyetlen publikus IP-cím használatával több szervert tegyünk elérhetővé, de ettől még a privát IP-címmel rendelkező gépek használhatósága nem teljes értékű, hiszen elérhetőségük nem automatikus, a NAPT eszközön beállításra van szükség, sőt egy adott portot csak egy eszköz fele lehet továbbítani. (Tehát ha két privát IP-címmel rendelkező, azonos publikus IP-cím mögé helyezett előfizető szeretne azonos porton szolgáltatni – például mindkettő web szervert szeretne üzemeltetni –, az már problémát okoz.)
- A közös publikus IP-címen való osztozás további problémája, hogy a kommunikációban részt vevő felek nem tudják egymást kölcsönösen azonosítani. Például egy web szerver nem a NAPT mögötti kliensek IP-címét látja (privát IP címük egyébként teljesen haszontalan is lenne), hanem az NAPT eszköz külső interfészének publikus IP-címét. Visszaélések megállapítása (lásd következő pont) vagy például szavazások (egy IP-címről naponta egy szavazat) esetén ez komoly gondot okoz.
- Internetszolgáltatók esetén fontos jogszabályi előírás, hogy képeseknek kell lenniük utólag adatot szolgáltatni arról, hogy adott időpontban egy adott IP-címet mely előfizetőjük használt. Többletköltség árán műszakilag természetesen megoldható, hogy az IP-címeket a portszámokkal együtt naplózzák, de még ekkor is problémát okozhat, ha olyan megkeresést kapnak, hogy csak időpont és IP-cím alapján kell azonosítaniuk az előfizetőt (mert a támadást vagy visszaélést bejelentő fél nem naplózta a kliens portszámát, csak IP-címét).

A fenti problémák majd NAT64-nél is előjönnek, annak az egyes hálózati alkalmazásokkal való kompatibilitásáról az IPv6 könyvben [1] további információ található.

NAPT mélyebben: hagyományos és kiterjesztett típusok

A fent bemutatott NAPT megoldás előnye az egyszerűség, aminek ára a portszámok pazarló használata. Már 2007-ben javasoltak olyan megoldást, ami a portszámokkal sokkal takarékosabban bánik [2]. Nézzünk bele most egy kicsit mélyebben mindkét megoldás működésébe!

Két számítógép között egyszerre több kommunikáció is folyamatban lehet, akár TCP akár UDP használatával. Bár az UDP kapcsolatmentes protokoll, nevezzük most ezeket mégis *kapcsolat*nak (connection vagy session) mindkét esetben. Egy kapcsolatot a következő 5 szám azonosít egyértelműen: forrás IP-cím, forrás portszám, cél IP-cím, cél portszám, protokoll (TCP vagy UDP).

Egy *hagyományos* (traditional) típusú NAPT implementáció a *kapcsolatok követésé*re (connection tracking) csak a 2. táblázat szerinti azonosítókat használja, tehát kihagyja a (privát IP címmel rendelkező kliens szempontjából) cél IP-címet és cél portszámot. Minden forrás IP-cím + forrás port páros esetén egyedire cseréli a kimenő csomagban a forrás portszámot. Mivel a portszámok 16 bitesek, és erre a célra csak az 1024-től 65535-ig terjedő tartományt használja, ezért összesen 63k darab TCP és ugyanennyi UDP kapcsolatot képes kezelni a kimenő interfészére felhúzott minden

egyes publikus IP-cím mellett. Ez jelentős korlátozás, ezért erre a problémára már 2007-ben javasolták a következő, ma elterjedten alkalmazott megoldást.

Egy *kiterjesztett* (extended) típusú NAPT implementáció a kapcsolatok követésére a 3. táblázat szerinti azonosítókat használja, tehát a cél IP-címet és cél portszámot is. Csak akkor cseréli ki a csomagban a forrás portszámot, ha az feltétlenül szükséges, azaz a csere nélkül nem lenne a kapcsolat egyértelműen azonosítható a visszaérkező csomagban szereplő értékek alapján. A 3. táblázatban több olyan esetet is bemutatunk, amikor a forrás portszámot nem kell cserélni, a csere csak az utolsó sorban volt szükséges. Ezzel a megoldással lényegesen sikerült kibővíteni a kimenő interfészre felhúzott publikus IP-címenként kezelhető kapcsolatok számát.

Source IP Address	Source Port Number	External IP Address	Temp. Port Number	Transport Protocol
10.1.2.2	5001	192.0.2.1	10001	TCP
10.1.2.3	5001	192.0.2.1	10002	TCP
10.1.3.5	5002	192.0.2.1.	10003	TCP

2. táblázat: Hagyományos NAPT kapcsolat követési táblázat (translation table) [3]

Source IP Address	Source Port Number	External IP Address	Temp. Port Number	Destination IP Address	Dest. Port Number	Transport Protocol
10.1.2.2	5001	192.0.2.1	5001	198.51.100.2	80	TCP
10.1.2.3	5001	192.0.2.1	5001	203.0.113.3	80	TCP
10.1.3.5	5001	192.0.2.1.	5001	198.51.100.2	443	TCP
10.1.3.6	5001	192.0.2.1.	10001	198.51.100.2	80	TCP

3. táblázat: Kiterjesztett NAPT kapcsolat követési táblázat (translation table) [3]

Megjegyezzük azonban, hogy a kezelhető kapcsolatok száma így sem végtelen.

A cél IP-cím és cél portszám bevonásának természetesen ára van, a kapcsolattáblában lényegesen nagyobb méretű kulcs alapján kell keresni! (A gyakorlatban hash függvény segítségével oldják meg a gyors keresést.)

Az extended megoldást használja például a Linux *Netfilter* keretrendszere [4] is, amit a felhasználói interfészének neve után *iptables*nek is szoktak nevezni.

2.2.2. Az Address plus Port (A+P) megoldás

Az A+P működési elve

Ez a megoldás is az IPv4 címek szűkösségének problémáját hivatott enyhíteni. Alapötlete, hogy az útválasztásba vonjuk be az IP-címeken túl a portokat is. Egy szolgáltató ugyanazt az IP-címet több előfizetőnek is kiosztja, és ezzel együtt az azonos IP-címmel rendelkező előfizetőknek diszjunkt és megfelelő méretű portszám tartományokat is kioszt. Ezenkívül mindegyik előfizetőnél olyan eszközt helyez el, ami az adott előfizető forrás portszám használatát a neki kiosztott portszám tartományra korlátozza. A szolgáltató ezután saját hálózatában az útválasztási algoritmust úgy módosítja, hogy az A+P megoldásba bevont címek esetén az adott címre érkező datagramok a cél portszám alapján a megfelelő előfizetőhöz érkezzenek meg. Nézzük, meg, hogyan működik ez a megoldás!

Az előfizető kliense természetesen nem foglalkozik azzal, hogy A+P mögött van, tehát tetszőleges forrás portot választ. A szolgáltató által kihelyezett eszköz NAPT segítségével átírja a forrás portot az előfizető számára kiosztott tartományból valamelyik szabad portra. A datagram gond nélkül eljut a címzetthez, majd a válasz visszaér a szolgáltatóhoz. A szolgáltató speciális útválasztása a cél portszám alapján az azonos IP-című előfizetők közül a megfelelőhöz továbbítja a datagramot, majd a kihelyezett NAPT eszköz átírja a portszámot és így a datagram eljut az illetékes alkalmazáshoz.

A megoldásról bővebben a (kísérletinek nevezett, 2011-ben kiadott) RFC 6346-ban olvashatunk.

Az A+P kritikája

A megváltoztatott útválasztást támogató eszköz még költséghatékonyan elhelyezhető a szolgáltatónál, de a minden előfizetőhöz kihelyezendő eszköz már érdemben drágítja a szolgáltatást. Az előfizetőknek kiosztandó diszjunkt portszám tartományok követelménye még súlyosabb probléma, ugyanis ha túl kicsire választják, akkor az korlátozza az előfizetőt[7], így csökkenti a szolgáltatás értékét, ha pedig túl nagyra választják, az viszont tönkreteszi a megoldás hatékonyságát. Például előfizetőnként 1024 portot számítva 64 előfizető[8] osztozhat egy publikus IP-címen. Ezért a megoldás jóval kevésbé hatékony, mint a NAPT, ahol nem kell maximumra méretezve fixen kiosztani az egyes előfizetőknek a portszám tartományokat, hanem a traditional esetben is statisztikai multiplexelés van, az extended hatékonysága pedig még annál is sokkal jobb. Az A+P előnye viszont a speciális útválasztást végző eszköz állapotmentessége: vele szemben egy extended NAPT eszköznek igen nagyszámú kapcsolat kezelésére kell képesnek lennie!

A NAPT-nál említett további problémák természetesen itt is fennállnak.

2.3. A DNS64 + NAT64 megoldás

2.3.1. A megoldás működése

Az IPv4 címek kifogyása miatt az IP 4-es és 6-os verziójának együttműködésében várhatóan az lesz az első tipikus megoldandó probléma, hogy az internetszolgáltatók csak IPv6 címet tudnak adni az új ügyfeleiknek, és a *csak IPv6 címmel rendelkező* (IPv6-only) klienseknek el kell érniük a *csak IPv4 címmel rendelkező* (IPv4-only) szervereket is.

A DNS64+NAT64 megoldás használhatóságának egyik szükséges feltétele, hogy a csatlakozni kívánó fél (kliens) a DNS rendszert használja az elérni kívánt fél (szerver) IP-címének kiderítésére, és a kliensben névkiszolgálóként egy DNS64 szerver legyen beállítva.

A *DNS64* szerver (RFC 6147) egy *caching-only* névkiszolgálóhoz hasonlóan *recursive query*kre válaszol. Ennek során, egy adott *szimbolikus név*hez (domain name):

- ha van IPv6 cím, akkor továbbítja a válaszában a kliensnek
- ha nincs IPv6 cím, de IPv4 cím van, akkor az IPv4 cím alapján generál egy speciális IPv6 címet, és azt adja vissza a kliensnek.

A speciális IPv6 cím egy *IPv4 címet beágyazó IPv6 cím* (IPv4-Embedded IPv6 Address) ami a legegyszerűbb esetben, az ún. *NAT64 Well-Known Prefix* használata esetén, a 64:ff9b::/96 prefix + az utolsó 32 biten a szimbolikus névhez tartozó IPv4 cím.

A megoldás működéséhez továbbá szükséges, hogy az útválasztási táblázatok szerint a NAT64 Well-Known Prefix (mint hálózat) felé az út egy *NAT64* átjárón (RFC 6146) keresztül vezessen (anycast címzés használható).

A megoldás működését a 3. ábra példáján mutatjuk be. Kövessük végig a példát. Az IPv6-only kliens csatlakozni szeretne az IPv4-only **`www.hit.bme.hu`** nevű webszerverhez, ennek érdekében:

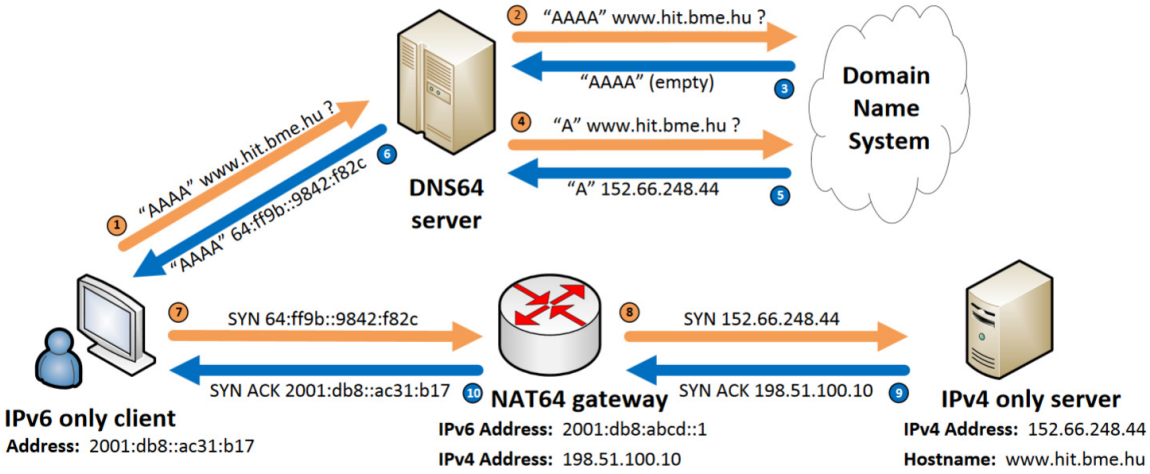
1. A kliens lekéri a **`www.hit.bme.hu`** szerver IPv6 címét („AAAA” rekord) a szimbolikus neve alapján.
2. A DNS64 szerver a DNS rendszertől megkérdezi a **`www.hit.bme.hu`** névhez tartozó IPv6-címet.
3. A kérdésre üres választ kap.
4. A DNS64 szerver a DNS rendszertől megkérdezi a **`www.hit.bme.hu`** névhez tartozó IPv4-címet („A” rekord).[9]
5. Válaszként a 152.66.248.44 IPv4-címet kapja.

[7] Lásd az IPv6 könyvben [1] az egyes alkalmazások portszám fogyasztását. Illetve egy előfizető (és családtagjai) egyidejűleg több eszköz használatára is igényt tarthatnak.

[8] Vagy 63, ha a 0-1023 tartományt nem osztjuk ki.

[9] Az RFC 6417 5.1.8. pontja azt is megengedi, hogy az „AAAA” és „A” rekord kérést konkurens módon küldje el (a második küldése előtt nem várja meg az első válaszát).

6. A DNS64 szerver a benne beállított NAT64 Well-Known Prefix (64:ff9b::/96) használatával előállítja a szükséges IPv4 címet beágyazó IPv6 címet (64:ff9b::9842:f82c, amit természetesen úgy is írhatnánk, hogy: 64:ff9b::152.66.248.44) és ezt elküldi a kliensnek.
7. A kliens TCP SYN szegmenst tartalmazó IPv6 csomagot küld a kapott IPv4 címet beágyazó IPv6 címre (64:ff9b::9842:f82c), amely a routing beállítása miatt a NAT64 átjáróhoz érkezik meg.
8. A NAT64 átjáró az IPv6 csomag alapján egy IPv4 csomagot készít; benne a célcím a speciális IPv6 célcím utolsó 32 bitje (152.66.248.44), a forráscím pedig a NAT64 átjáró IPv4 címe lesz. Közben a NAT64 átjáró eltárolja a kapcsolattáblájába a szükséges információkat, és ha korszerű az implementáció, akkor a forrás portszámot csak szükség esetén cseréli. Az átjáró az elkészült IPv4 csomagot elküldi a címzettnek, ami az IPv4-only szerver.
9. Az IPv4-only szerver megkapja a TCP SYN szegmenst tartalmazó IPv4 csomagot és a megszokott módon válaszol (SYN+ACK). Mivel a kapott IPv4 csomagban a forráscím a NAT64 átjáró IPv4 címe volt, a válasz címzettje is a NAT64 átjáró IPv4 interfésze lesz, a forráscím pedig a saját IPv4 címe. A válasz megérkezik az NAT64 átjáróhoz.
10. A válasz alapján a NAT64 átjáró elkészíti a neki megfelelő IPv6 csomagot, melybe forráscímként ugyanaz a speciális IPv6 cím kerül, amit a DNS64 szerver generált, célcímként az IPv6-only kliens IPv6 címe kerül; mindez a kapcsolattábla alapján történik. Az IPv6 csomagot a NAT64 átjáró elküldi az IPv6-only kliensnek.



3. ábra: DNS64+NAT64 megoldás elvi működése [5]

Amikor az IPv6-only kliens megkapja a csomagot, úgy folytatja a kommunikációt, hogy közben mit sem sejt arról, hogy a szervernek IPv4 címe van. A klienshez hasonlóan a szerver sem észlel semmi különöset, hiszen látszólag a NAT64 átjáró IPv4 interfészével kommunikál. Ennek persze az a következménye, hogy a szerver az IPv6-only kliens valódi IPv6 címe helyett a NAT64 átjáró IPv4 címét fogja naplózni. Ez egyben a megoldás egyik gyengesége is, de természetesen nem csupán a NAT64-é, hanem a privát IP-címek használatakor alkalmazott NAT-é is, mint azt már említettük. (Számos esetben okoz súlyos problémát, például naplózásnál, IP-cím alapján végzett szavazásnál, vagy kitiltásnál, stb.)

Megjegyzések:

1. Itt állapottartó NAT64-ről (Stateful NAT64) van szó, amely több IPv6 címről is ugyanarra az IPv4 címre fordít. Ezzel szemben az állapotmentes NAT64 csak 1-1 összerendelésre képes az IPv6 és IPv4 címek között.
2. A NAT64-től való megkülönböztetésül a NAT-ot NAT44-nek is hívják, amikor mind a lecserélt, mind az új IP-cím verziószáma 4-es.
3. A példában szereplő **`www.hit.bme.hu`** névhez ma már létezik „AAAA” rekord is.

A fentiekben bemutatott megoldás az RFC 6052 szerinti 64:ff9b::/96 előre lefoglalt, ún. NAT64 Well-Known Prefixet használja. A prefix használatának számos korlátja van, lásd RFC 6052 3. rész. A NAT64 átjáró megvalósításakor a gyakorlatban sok esetben az egyes IPv6 hálózatokból szoktak erre a célra lefoglalni egy részt, ezt hálózat-specifikus prefixnek (Network-Specific Prefix) nevezik.

2.3.2. Az IPv4-címeket beágyazó IPv6-címekről

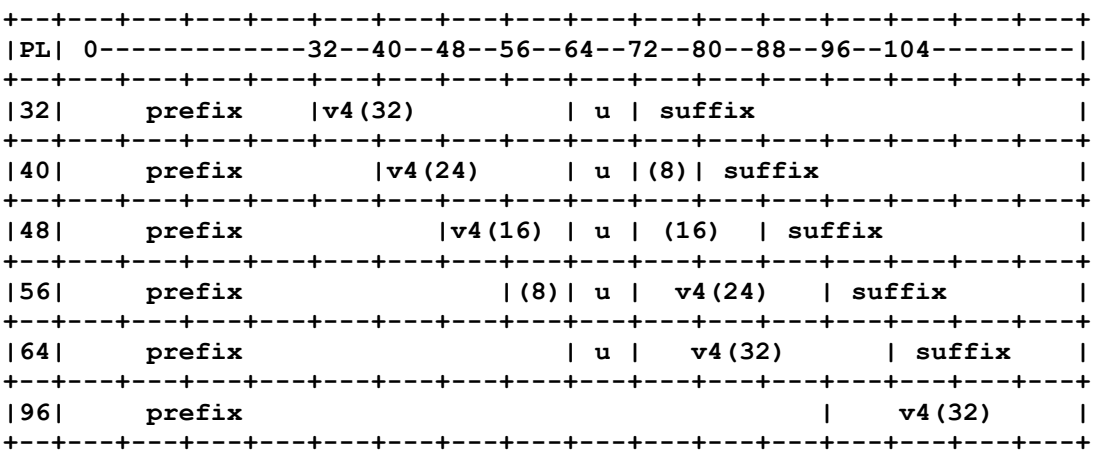
Az *IPv4-címeket beágyazó IPv6-cím*eket (IPv4-Embedded IPv6 Address) még két további névvel illetik a felhasználásuk céljától függően, bár szerkezetük és előállításuk azonos.

- Azokat az IPv6 címeket, amiket arra használunk, hogy IPv4 állomásokat képviseljenek IPv6 hálózatokban, úgy nevezzük, hogy *IPv4-ből konvertált IPv6 cím*ek (IPv4-Converted IPv6 Address). A fenti példában pontosan erről volt szó.
- Az *IPv4-re lefordítható IPv6 cím* (IPv4-Translatable IPv6 Address) megnevezést pedig akkor használjuk, ha a cím egy IPv6 állomáshoz tartozik, és a fordítás célja, hogy a csak IPv4-re képes eszközök is el tudják érni az IPv6 állomást. (Ezzel az esettel most nem foglalkozunk.)

Az RFC 6052 definiálja, hogy hogyan kell az IPv4-címeket beágyazó IPv6-címeket képezni. Hálózat-specifikus prefixnél a hálózat adminisztrátora dönti el, hogy a rendelkezésére álló címtartományból mekkora tartományt szeretne erre a célra szánni. Ennek megválasztása során tekintettel kell lennie az alábbi szabályokra is:

- A prefix mérete szigorúan csak 32, 40, 48, 56, 64 vagy 96 lehet.
- Az IPv6 címben a 64-71. sorszámú biteknek 0 értékűeknek kell lenniük.
- Az IPv4 cím 32 bitjét a választott méretű prefix után írjuk, de a fenti követelmény kielégítése érdekében a szigorúan 0 értékű bitek helyét „átugorjuk”.
- A cím végét szükség esetén ugyancsak 0 értékű bitekkel töltjük ki.

A címek lehetséges formátuma:



Ha például egy /48-as címtartományunk van, akkor a hálózat-specifikus prefix mérete /56, /64 vagy /96 lehet. A választás szempontjairól az RFC 6052 3.3. és 3.4. részében olvashatunk.

2.3.3. A megoldás élettartama

IPv6 áttérési megoldásról lévén szó, az nyilvánvaló, hogy akkor már nem lesz szükség rá, amikor minden eszköz képes lesz az IPv6-ra. Azonban ehhez várhatóan még akár egy vagy több évtizedre is szükség lehet. A megvalósításhoz használt eszközök forgalma kezdetben várhatóan növekedni fog a csak IPv6-ra képes kliensek számának növekedésével, aztán várhatóan csökkenésnek indul, amint egyre nagyobb arányban képesek lesznek a szerverek IPv6-ra. És a forgalom hosszú idő alatt válik elenyészővé. Olyan technológiának ítéljük meg, amit érdemes mind megismerni, mind kutatni, mert legalább egy évtizedig szükség lesz rá.

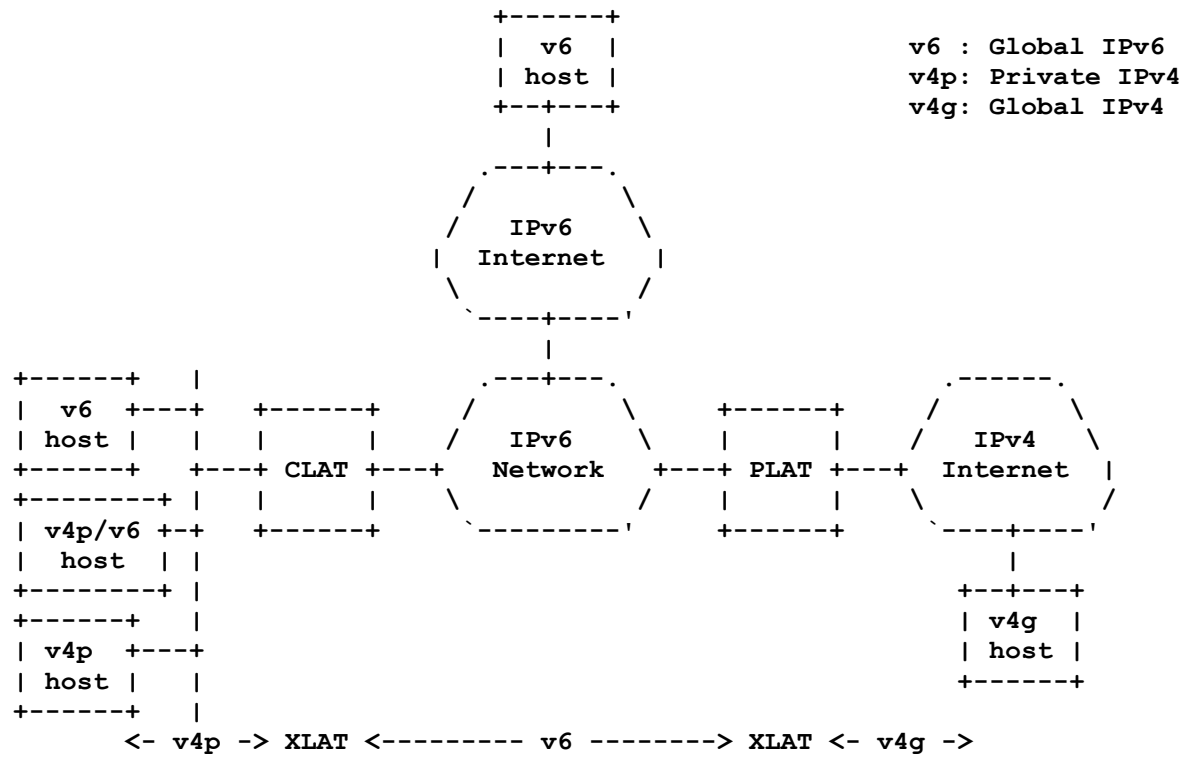
2.4. Az IPv4 mint szolgáltatás (IPv4aaS) probléma megoldásai

Egyre több olyan internetszolgáltató létezik, amely nem kíván IPv4 és IPv6 hálózatot is üzemeltetni, hanem csak IPv6-ot használ a hálózatában. Vannak viszont olyan alkalmazások (pl. Skype, Spotify, stb.), amelyek nem képesek IPv6-ot használni. Mivel a csak IPv4-re képes alkalmazásokhoz számos felhasználó ragaszkodik, ezért a szolgáltatóknak az előfizetőik felé támogatniuk kell az IPv4-et. A probléma megoldására számos protokoll született, ezeknek különféle előnyei és hátrányai vannak, melyek adott esetben döntőek lehetnek abból a szempontból, hogy egy szolgáltatónak melyiket célszerű választania. Az egyes megoldások összehasonlításán az IETF IPv6 Operations munkacsoportjában egy Internet Draft [6] keretében jelenleg is dolgozunk, amelyet az alábbiakban is felhasználtam.

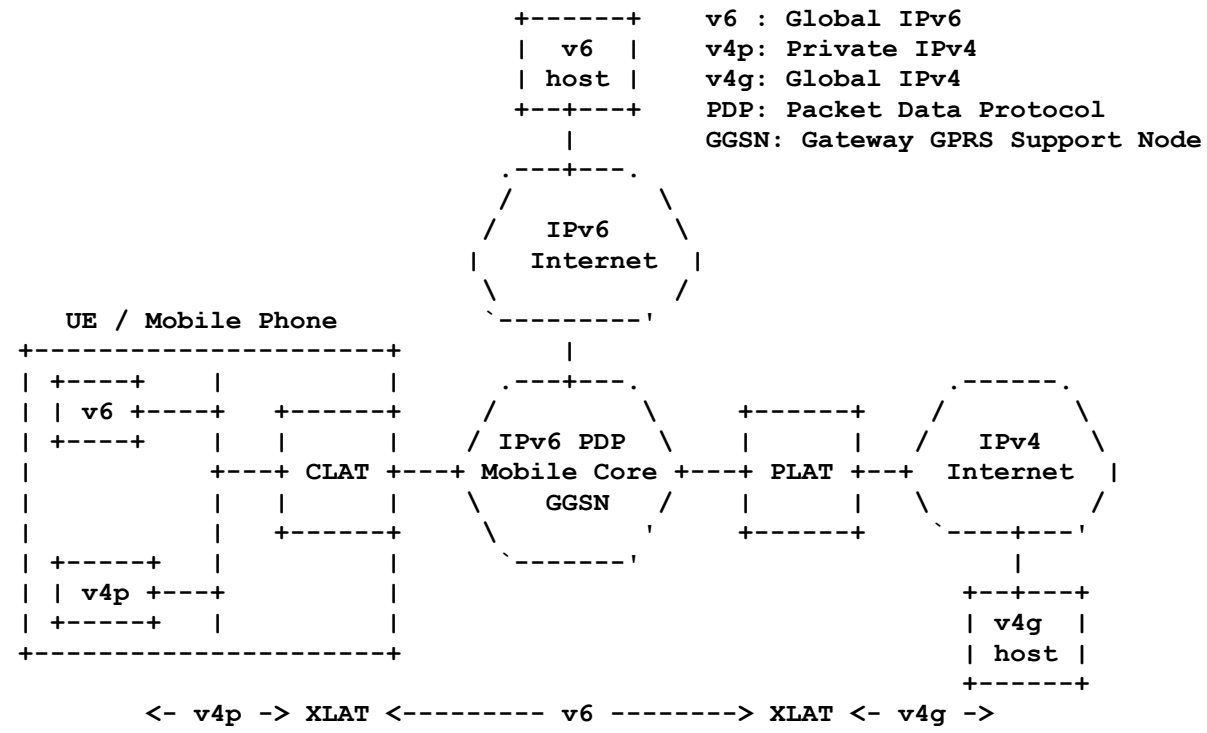
2.4.1. A 464XLAT megoldás

A 464XLAT (RFC 6877) megoldás a csak IPv4-re képes alkalmazások (kliensek) számára privát IPv4 címeket nyújt. Az IPv4 csomagokat a kliens oldalon a CLAT eszköz IPv6-ra fordítja (állapotmentes NAT46: minden privát IPv4 címhez eltérő IPv6 cím tartozik), és természetesen a másik irányú forgalom csomagjaira is elvégzi az IPv6-ról IPv4-re való fordítást. A szolgáltató hálózatában így csak IPv6 forgalom jelenik meg. (Az IPv6 csomagok szempontjából a CLAT routerként viselkedik.) Természetesen a szolgáltató eszközei a prefix alapján felismerik azokat az IPv6 címeket, amelyek IPv4-címek megfelelői, és ezeket a külvilág felé egy PLAT eszközre irányítják, amely nem más, mint a már megismert állapottartó NAT64 átjáró.

Vezetékes és mobil hálózatoknál némileg eltér a CLAT funkció megvalósításának elhelyezése. Míg vezetékes hálózatoknál a CLAT egység tipikusan egy külön eszköz, addig mobil hálózatok esetén a CLAT a felhasználó eszközén futó szoftver. Vezetékes hálózatoknál a rendszer felépítése az 4. ábra szerint alakul. Az ábrán jól látható, hogy a felhasználónak lehetnek csak IPv4-re, csak IPv6-ra és mindkettőre képes eszközei is.



4. ábra: 464XLAT rendszer felépítése vezetékes hálózat esetén (RFC 6877)



5. ábra: 464XLAT rendszer felépítése mobil hálózat esetén (RFC 6877)

Ezzel szemben egy mobil hálózat logikai felépítését az 5. ábra mutatja be. Ebben az esetben tehát nem külön eszköz, hanem a mobil eszközön futó szoftver valósítja meg a CLAT funkciót.

A 464XLAT célszerűen együtt használható a DNS64+NAT64 megoldással. Ilyen esetben háromféle forgalmat különböztethetünk meg:

1. IPv6-ra képes kliens és IPv6-ra képes szerver között: natív IPv6 forgalom.
2. Csak IPv6-ra képes (vagy valamiért azt preferáló) kliens forgalma csak IPv4-re képes szerverrel: a DNS64+NAT64 megoldást használja.
3. Csak IPv4-re képes (vagy valamiért azt preferáló) alkalmazás (vagy kliens) forgalma (az IPv4 Internet felé): 464XLAT segítségével halad át a szolgáltató hálózatán.

Vegyük észre, hogy:

- A fenti három forgalom közül az 1. és a 2. esetén a CLAT eszköz csak routerként viselkedik.
- Az állapottartó NAT64 átjárón mind a 2. mind a 3. fajta forgalom áthalad.

Arra is lehetőség van, hogy a 464XLAT megoldást a DNS64+NAT64 megoldás nélkül használjuk. Ilyen esetben csak kétféle forgalom lesz a rendszerben:

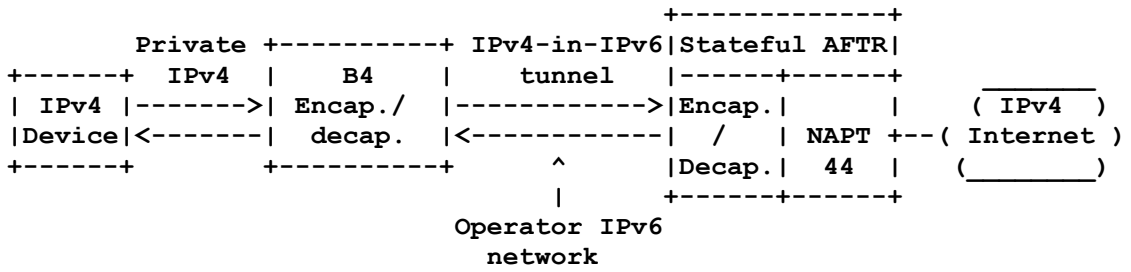
1. IPv6-ra képes kliens és IPv6-ra képes szerver között: natív IPv6 forgalom.
2. Minden más esetben a kliens és a szerver IPv4-et használ, és forgalmuk kétszeres fordításon megy át.

Megjegyzés: Bár DNS forgalom is haladhat a fenti szabályok szerint, a CLAT eszközön DNS proxyt is megvalósítanak, így az azt használó kliensek DNS forgalma nem fog kétszeres fordításon átmenni.

A 464XLAT különösen mobil szolgáltatóknál terjed gyorsan. Néhány évvel ezelőtt a három hazai mobilszolgáltató közül kettőnek a munkatársaival való beszélgetésből is kiderült a számomra, hogy ismerik és fontosnak tartják. ☺

2.4.2. A DS-Lite megoldás

A Dual Stack Lite (RFC 6333) történetileg az első olyan megoldás, ami IPv6 szolgáltatói hálózaton IPv4 szolgáltatást nyújt a felhasználóknak. A felhasználó számára privát IPv4 címet ad, és a felhasználó IPv4 csomagjait a nála elhelyezett B4 (Basic BroadBand Bridging) elem IPv6 csomagokba



6. ábra: A DS-Lite működésének vázlata [6]

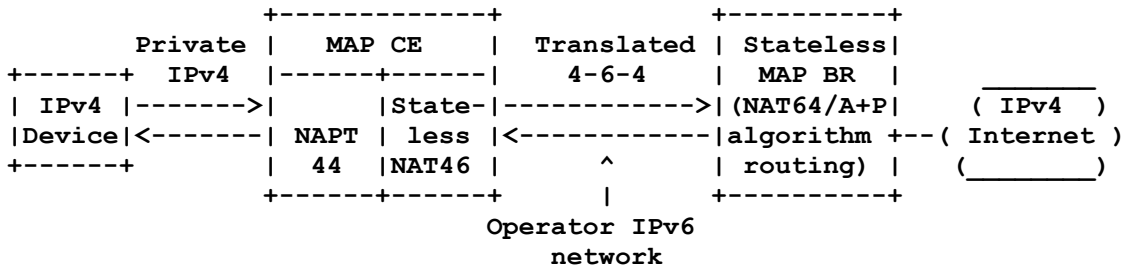
ágyazza (és természetesen a visszafele irányuló csomagokat kibontja) és IPv6 csomagként szállítja a szolgáltató hálózatában a B4 és az AFTR (Address Family Transition Router) elem között. Az AFTR valósítja meg a kicsomagolást (és a visszafele irányuló forgalom becsomagolását), valamit az állapottartó NAPT44 funkciót. A DS-Lite működését a 6. ábrán mutatjuk be.

A központosított NAPT44-et tekintve a DS-Lite hasonlít a 464XLAT-ra, a beágyazást tekintve pedig éppen annak az ellenpárja. A beágyazás kompatibilitási szempontból előnyösebb lehet a kétszeres fordításnál, viszont a plusz fejrész kis csomagok esetén jelentős *overhead*et okozhat. (IPv4-ről IPv6-ra történő fordítás esetén a fejrész mérete 20 oktettel nő, míg az IPv4 datagaramoknak IPv6 datagramokba való beágyazása 40 oktettel növeli meg azok méretét.)

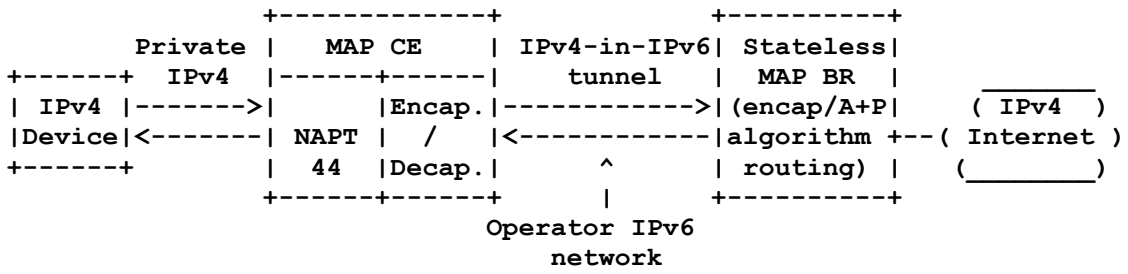
2.4.3. A MAP-T és a MAP-E megoldások

A MAP-T (RFC 7599) legfelső szinten (az 464XLAT-hoz hasonlóan) kétszeres címfordítást használ: először IPv4-ről IPv6-ra majd IPv6-ról IPv4-re fordít, viszont itt a fordítás állapotmentes, ami a megoldás egyik előnye (skálázhatóság). A MAP-E (RFC 7597) pedig az IPv4 csomagot IPv6 csomagba ágyazza, majd kibontja. A két megoldás között csak ez a különbség, egyébként nagyon hasonlítanak egymásra. A MAP megoldás rokonságban van az A+P módszerrel, ezt tükrözi a két változatának teljes neve is: *Mapping of Address and Port using Translation* (MAP-T), illetve *Mapping of Address and Port with Encapsulation* (MAP-E). Viszont a MAP ügyes trükkel a portszámot „az IPv6 címben tárolja” (ez így nem egészen pontos, valójában MAP szabályokat használ, de most nem megyünk mélyebbre), ezért az IPv6 hálózatban szabályos, IPv6 cím alapú útválasztást végezhet, és a MAP-T forgalom szabályos IPv6 forgalom (pl. Wiresharkkal dekódolható).

A MAP-T működésének lépéseit a 7. ábrán követhetjük. A MAP CE (Customer Edge) eszköz először egy állapottartó NAPT44-et hajt végre, amivel a felhasználó számára kiosztott IPv4-cím és portszám tartományra fordít, majd egy állapotmentes NAT46 segítségével IPv4-ről IPv6-ra fordít. A felhasználó számára kiosztott IPv4 cím és portszámt tartományt úgy tervezték meg, hogy az egyes felhasználók tartományai diszjunktak, így a központi egységnek már csak állapotmentes fordítást kell végrehajtania, ezért elméletileg jobban skálázódik, mint azok a megoldások, amelyek esetén a hálózat központjában állapottartó NAT-ot használnak (464XLAT, DS-Lite). Megjegyzés: az előfizetőnél történő állapottartó megoldás használatát a skálázhatóság (az előfizetők számának növelése hogyan



7. ábra: A MAP-T működésének vázlata [6]



8. ábra: A MAP-E működésének vázlata [6]

hat a rendszer teljesítményére) szempontjából nem tartják problémának.

A MAP-E megoldás működése teljesen hasonló, a 8. ábrán követhető. Itt kétszeres fordítás helyett beágyazás történik.

A MAP megoldás egyik hátrányát is az A+P megoldásból „örökli". Bár az egyes felhasználók számára kiosztott portszám tartomány mérete változtatható, az előre történő kiosztás miatt a portszám felhasználás hatékonysága akkor sem versenyképes a 464XLAT és DS-Lite esetén használt központosított állapottartó NAPT portszám felhasználásának (és így publikus IPv4 címek felhasználásának) hatékonyságával. Egy másik hátránya a bonyolultság. Ráadásul a hosszabb kódban több a hibalehetőség, és várhatóan a biztonsági rések száma is nagyobb lehet.

A MAP megoldások előnye viszont, hogy képesek publikus IPv4 címek használatára is, ami azt jelenti, hogy például IPv4 címre hallgató szervert is üzemeltethet egy olyan felhasználó, amelynek az internetszolgáltatója a saját hálózatában már kizárólag IPv6-ot használ.

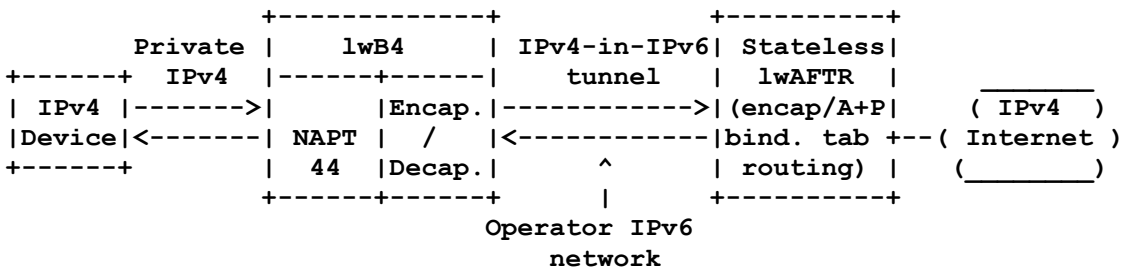
Összefoglalva megállapítjuk, hogy egyik megoldást sem tekintjük jobbnak a másiknál, hanem eltérő esetekben eltérő megoldás választása lehet célszerű.

2.4.4. A Lightweight 4over6 megoldás

Egy másik lehetőség a Lightweight 4over6 (lw4o6, RFC 7596), amely a DS-Lite egy kiterjesztése (más szempontból pedig a MAP-E egy speciális esete). A DS-Lite-hoz képest a fő különbség az, hogy az állapottartó NAPT44-et a központi elemből a B4 elembe helyezték, amit itt lwB4-nek hívnak. Így a központi elem (lwAFTR) csak A+P szerinti csomagtovábbítást és 6in4 beágyazást/kibontást végez. A megoldás működését a 9. ábrán követhetjük, ami nem meglepő módon nagyon hasonlít a 8. ábrára.

2.4.5. Összehasonlítás helyett

A megismert öt IPv4aaS megoldás előnyeit és hátrányait a korábban említett Internet Draft [6] készítése keretében vizsgáljuk, és még számos nyitott kérdés van, ilyennek számít az egyes megoldások teljesítménye, különös tekintettel annak skálázódására. Aszimptotikus értelemben (a kliensek száma tart a végtelenbe) az állapotmentes megoldások (ahol a rendszer központi eleme állapotmentes) nyilván jobban skálázódnak, mint az állapottal rendelkezők, azonban véges, bár



9. ábra: Az lw4o6 működésének vázlata [6]

nagyszámú (pl. néhány millió) kliens esetén az állapottartó megoldások versenyképesek lehetnek, amennyiben az állapottáblában való keresést (pl. például hash függvény segítségével) képesek hatékonyan elvégezni. (A 464XLAT működése lényegesen egyszerűbb, mint a MAP-T/E megoldásoké.)

Összefoglalásul az egyes megoldásokat kétféle szempont szerint rendszereztük a 4. táblázatban:

- az ISP IPv6 hálózatán való áthaladáshoz használt technológia, ami lehet kétszeres fordítás vagy beágyazás/kibontás
- állapot megléte vagy annak hiánya a szolgáltató hálózatában.

Érdeklődő hallgatók számára lehetőség van az egyes technológiák összehasonlításában való részvételre TDK munka, témalabor, önálló labor, szakdolgozat keretében.

	Service provider network traversal technology	
	single/double translation	encapsulation/ de-encapsulation
State in the operator network: present	464XLAT	DS-Lite
State in the operator network: absent	MAP-T	MAP-E, Lw4o6

4. táblázat: Az ötféle IPv4aaS technológia összefoglaló rendszerezése [6]

2.5. A DNS46+NAT46 megoldás

A megoldandó probléma éppen a fordítottja, mint a DNS64+NAT64 esetén: itt a csak IPv4 címmel rendelkező kliensek számára szeretnénk biztosítani a csak IPv6 címmel rendelkező szerverek elérését. Sajnos esetünkben a fordított probléma megoldása nem használható egy az egyben. Míg a teljes IPv4 címtartomány könnyen leképezhető az IPv6 címtartomány egy kis részére (egy IPv6 címben bőségesen elfér egy teljes IPv4 cím), a fordítottja nem igaz. Ezért dinamikus összerendelést (mapping) hoznak létre a két címtartomány egyes elemei között. Mivel a kliens kizárólag IPv4 címeket képes kezelni, a DNS46 megoldás az elérni kívánt szerver szimbolikus neve alapján kiderített IPv6 címhez egy IPv4 címet rendel, és azt adja vissza a kliensnek. A kliens az így kapott IPv4 címet célcímként felhasználva kapcsolódik a szerverhez. Úgy van beállítva a routing, hogy a DNS46 szerver által visszaadott *IPv6 címeket reprezentáló IPv4 címek* tartománya felé az átjáró egy NAT46 eszköz, ami ismeri a DNS46 szerver aktuális összerendeléseit (mapping), és elvégzi a címek cseréjét, valamint a protokoll konverziót. Ezt a megoldást leíró RFC sajnos nem került elfogadásra, az utolsó draft már 2010-ben lejárt [7]. Bár a megoldást nem szabványosították, mégis több implementációja létezik. Szabad szoftverként például létezik modul az OpenWRT-hez [8], és egyes nagy hálózati eszközgyártók termékei is támogatják, például a Cisco ASA 5510 tűzfal [9] vagy a Brocade ServerIron ADX [10].

Mivel a megoldás nem szabványos, mélyebben nem foglalkozunk vele, viszont kíváncsian várjuk, hogy az IETF milyen szabványos megoldást kínál a problémára.

2.6. A 6to4 megoldás

Ezt a megoldást akkor használjuk, ha egy IPv6 képes eszköz IPv4-only környezetben van, és IPv6 protokollal egy másik IPv6-os eszközt szeretne elérni (akár az is lehet IPv4-only környezetben).

A 6to4 megoldás (RFC 3056) egy „automatikus” tunnel, ami az IPv6 datagramokat IPv4 datagramokba csomagolja be (és természetesen ki is). A 6in4-hez hasonlóan a 41-es protokollazonosítót használja (lásd később).

2.6.1. A 6to4 címzése

Ahhoz, hogy egy eszköz IPv6 protokollt tudjon használni, szüksége van IPv6 címre. Mivel a 6to4 megoldást natív IPv6 Internet elérés hiányában használjuk, az IPv6 címet az IPv4 címből állítjuk elő, és a natív IPv6 címektől való megkülönböztetés érdekében *6to4 IPv6 cím*nek hívjuk.

A 6to4 címzéshez a 2002::/16 prefixet foglalták le. A 6to4 IPv6 címek képzése az alábbi módon történik.

- Hálózatcím: 2002::/16 prefix + publikus IPv4 cím 32 bitje + 16 bit subnet ID
  - Egyetlen host (lásd később) esetén a subnet ID egy generált véletlenszám.
  - Ha a 6to4 mechanizmust router használja, akkor akár több IPv6 hálózat is lehet mögötte, ekkor hasznos a subnet ID.
- Gépcím: A szabványos módosított EUI-64 azonosító

Ilyen módon a 6to4 megoldás minden publikus IPv4 címhez egy 2002::/16 kezdetű, /48 méretű IPv6 címtartományt rendel. Mindegyik IPv4 cím „mögött” elérhető lehet egy ilyen méretű IPv6 hálózat.

2.6.2. A 6to4 megoldás működése

A kommunikáció során a 6to4 IPv6 címmel rendelkező állomások datagramjainak IPv4 datagramokba való be- és kicsomagolását végző *6to4 pszedo-interfész* (6to4 pseudo-interface) elhelyezése szempontjából kétféle konfiguráció lehetséges:

- lehet egyetlen host, amin az IPv6 kliens fut, és a becsomagolást is a host végzi – a 6to4 pseudo-interface a hoston van (*6to4 host*)
- lehet több IPv6-os gép egy IPv6 hálózaton, aminek a routere végzi a becsomagolást – a *6to4 (border) router* rendelkezik 6to4 pseudo-interface-szel.

A működés során ez a két eset nem különbözik lényegesen, az viszont igen (és ezért külön tárgyalást igényel), hogy az IPv4 környezetben levő IPv6 eszköz – nevezzük a továbbiakban *6to4 IPv6 állomás*nak – egy az IPv6 Internetre kapcsolódó *natív IPv6 állomás*sal vagy egy másik 6to4 IPv6 állomással kommunikál-e.

Tekintsük először egy 6to4 IPv6 állomás és egy natív IPv6 állomás kommunikációját. Ebben az esetben az IPv4 hálózat és a natív IPv6 Internet között egy további eszközre van szükség: ez a *6to4 relay*. A 6to4 relay feladata a 6to4 IPv6 állomástól érkező, IPv4 datagramba ágyazott IPv6 datagram kicsomagolása és továbbítása az IPv6 Interneten át a natív IPv6 állomás felé. A másik irányú kommunikáció során szintén a 6to4 relay feladata a natív IPv6 állomástól származó datagram IPv4 datagramba való becsomagolása és az IPv4 hálózaton keresztül való továbbítása. Ilyen eszközből több is lehet, az IPv4 irányából a legközelebbi a 192.88.99.1 anycast címen érhető el.

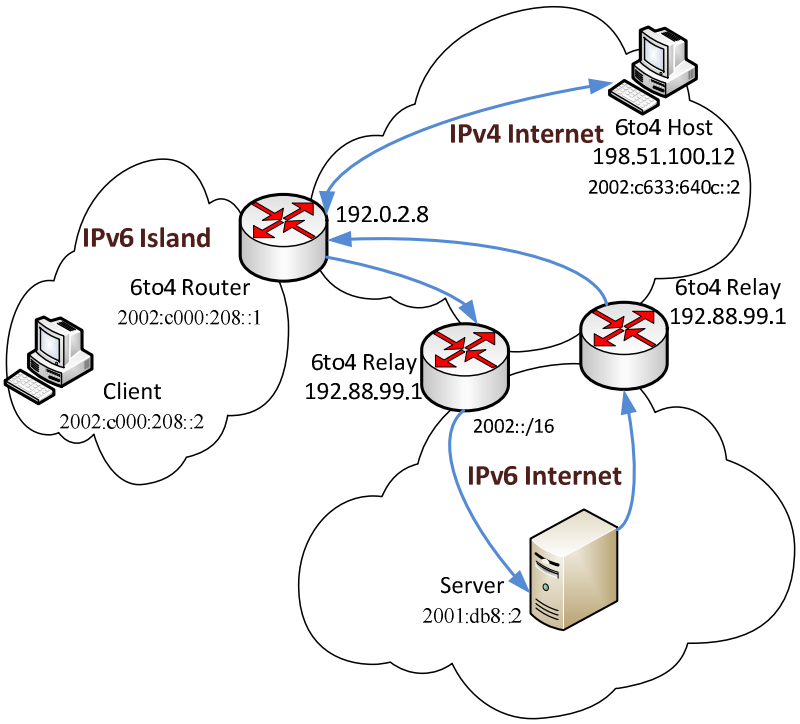
A 6to4 megoldás működését a 10. ábrán látható hálózat példáján mutatjuk be, ahol egy 6to4 IPv6 címmel rendelkező kliens kommunikál egy natív IPv6 szerverrel. A kommunikáció lépései:

- A kliens a szerver felé IPv6 csomagot küld (forráscím: 2002:c000:208::2, célcím: 2001:db8::2).
- A becsomagolás elvégzője (itt most a 2002:c000:208::1 címen elért 6to4 router, de lehetne maga a host is) az IPv6 csomagot IPv4 csomagba ágyazza, és az IPv4 segítségével elküldi egy 6to4 relaynek (forráscím: 192.0.2.8, célcím: 192.88.99.1).
- A beágyazást elvégző 6to4 routerhez az IPv4 hálózat metrikája szerint legközelebbi (192.88.991 anycast címre hallgató) 6to4 relay kapja meg a csomagot.
- A 6to4 relay kicsomagolja az IPv4-be ágyazott IPv6 csomagot, majd továbbítja a natív IPv6 hálózatba (forráscím: 2002:c000:208::2, célcím: 2001:db8::2).
- A natív IPv6 hálózatban a csomag megérkezik a címzetthez.
- A címzett válaszol (forráscím: 2001:db8::2, célcím: 2002:c000:208::2).
- A válasz a natív IPv6 állomáshoz az IPv6 Internet metrikája szerint legközelebbi, az IPv6 hálózatba 2002::/16 prefixet hirdető 6to4 relayhez érkezik meg. Ez lehet az előbbivel azonos vagy attól eltérő 6to4 relay (jelen példánkban eltérő). Erre a kommunikáló feleknek nincs befolyása!

- Visszafele a 6to4 relay az IPv6 csomagot IPv4-be csomagolva küldi a kliensnek, pontosabban a korábban a kliens IPv6 csomagját IPv4-be csomagoló eszköznek, ami példánkban a 192.0.2.8 IPv4 címre hallgató 6to4 router (forráscím: 192.88.99.1, célcím: 192.0.2.8). Megjegyzés: A 192.0.2.8 célcímet a 6to4 relay a 2002:c000:208::2 IPv6 célcímből tudta megállapítani.
- A válasz megérkezik a 192.0.2.8 IP-című 6to4 routerhez.
- A 6to4 router kicsomagolja az IPv4 csomagból az IPv6 csomagot és elküldi a kliensnek (forráscím: 2001:db8::2, célcím: 2002:c000:208::2).
- Végül a kliens megkapja a szerver válaszát.

A fenti példában a 6to4 állomás volt a kliens a natív IPv6 állomás pedig a szerver, de a fordított esetnek sincs akadálya, a kommunikáció a natív IPv6 irányából is kezdeményezhető.

A megoldás működéséhez szükséges feltétel, hogy a becsomagolást végző eszköznek (6to4 router vagy 6to4 host) legyen publikus IPv4-címe, különben nem tudna érvényes 6to4 IPv6 címet nyújtani a mögötte található eszközöknek. Ha a becsomagolást végző eszköznek nincs publikus IPv4 címe, akkor



10. ábra: A 6to4 megoldás lehetőségei [11] (módosítva)

a 6to4 helyett *Teredo*t lehet használni (RFC 4380).

A másik eset, amikor két 6to4 állomás kommunikál egymással (két IPv6 szigetet kötünk össze a 6to4 segítségével az IPv4 Internet felhasználásával). Ebben az esetben a becsomagolást végző eszköz a datagram cél IPv6-címéből (2002::/16 prefix) tudja meg, hogy nem egy 6to4 relaynek, hanem a megfelelő IPv4 című pszeudo-interfésszel rendelkező eszköznek (6to4 host vagy 6to4 router) kell IPv4 szinten címeznie és küldenie a csomagot. A 10. ábra példája segítségével egy ilyen esetet is nyomon követhetünk. Kommunikáljon most az előbbi példában szereplő kliens az ábra jobb felső sarkában található 6to4 állomással:

- A kliens a 6to4 állomás felé IPv6 csomagot küld (forráscím: 2002:c000:208::2, célcím: 2002:c633:640c::2).
- A becsomagolás elvégzője (a 2002:c000:208::1 című 6to4 router) az IPv6 csomagot IPv4 csomagba ágyazza. Ennek során az IPv6 csomagban szereplő célcím 2002::/16 prefixe alapján tudja, hogy a datagramot nem egy 6to4 relaynek kell címeznie, hanem a cél IPv4 cím a cél

IPv6 címből az első 16 bitet követő 32 bit, azaz 198.51.100.12 lesz (forráscím: 192.0.2.8, célcím: 198.51.100.12).

- A csomag megérkezik a 198.51.100.12 IP-című 6to4 hosthoz.
- A 6to4 host pszeudo-interfésze elvégzi a kicsomagolást és továbbítja az IPv6 datagramot az IPv6 interfésznek (forráscím: 2002:c000:208::2, célcím: 2002:c633:640c::2).
- Az IPv6 címen elérhető alkalmazás válaszol (forráscím: 2002:c633:640c::2, célcím: 2002:c000:208::2).
- A választ a 6to4 host pszeudo-interfésze IPv4 csomagba ágyazza. A cél IPv4 címet szintén a cél IPv6 címből állapítja meg (forráscím: 198.51.100.12, célcím: 192.0.2.8).
- A válasz megérkezik a 192.0.2.8 IP-című 6to4 routerhez.
- A 6to4 router kicsomagolja az IPv4 csomagból az IPv6 csomagot és elküldi a kliensnek (forráscím: 2002:c633:640c::2, célcím: 2002:c000:208::2).
- Végül a kliens megkapja a 6to4 host válaszát.

Figyeljük meg, hogy ebben az esetben a kommunikációban mindkét irányban ugyanazok a 6to4 interfészek vettek részt. (Ez nem véletlen, hiszen a kommunikáló eszközök 6to4 IPv6 címében az IPv4 címek egyértelműen rögzítettek voltak.)

2.6.3. A 6to4 megoldás élettartama

A megoldás már régóta üzemel. A továbbiakban már csak viszonylag rövid ideig van/lesz rá szükség; addig, amíg lesz olyan terület, ahol az internetszolgáltatók nem nyújtanak IPv6 elérést. Bár az Amerikai Egyesült Államokban némelyek már (2011 óta) temetni szeretnék a megoldást[10], Magyarországon Budapest kivételével még nagyon is valóságos probléma, hogy valaki nem tud IPv6 internet előfizetést vásárolni. De reméljük, hogy a helyzet megváltozik, és akkor 6to4 jelentéktelenné válik, és kikerül a tananyagból.

2.6.4. A 6to4 megoldás problémái

Szolgáltatásminőség

A 6to4 megoldás által nyújtott szolgáltatás minőségére súlyos következményekkel jár az, hogy amennyiben egy 6to4 IPv6-ot használó állomás egy a natív IPv6-ot használó állomással kommunikál, akkor – amint láttuk – a két irányban nem feltétlenül azonos útvonalon (és 6to4 relayen) halad a kommunikáció, és ezen ráadásul a 6to4 állomás internetszolgáltatója sem tud segíteni. Miért? Vizsgáljuk meg ezt a kérdést! Amennyiben a 6to4 állomás internetszolgáltatója egyáltalán nem foglalkozik azzal, hogy ügyfelei részére IPv6 elérést biztosítson, akkor a forgalom mindkét irányban valamilyen más szervezet által üzemeltetett 6to4 relayen halad keresztül. (A 6to4 router pedig az ügyfélnél található.) Ha az internetszolgáltató rendelkezik natív IPv6 Internet eléréssel, akkor üzemeltethet egy saját 6to4 relayt, amit az ügyfeleinek a 192.88.99.1 anycast címen meghirdetve elérheti, hogy azok kimenő 6to4 IPv6 forgalma rajta, mint számukra legközelebbi 6to4 relayen keresztül menjen át az IPv6 Internetre. Arra azonban nincs befolyása, hogy a visszafele irányuló forgalom melyik 6to4 relayen menjen keresztül; egy adott ügyfél esetén pontosan azon a relayen fog átmenni, amely az IPv6 Interneten a legközebb van ahhoz a natív IPv6 állomáshoz, amellyel az ügyfél számítógépe éppen kommunikál. Ha az adott relay túlterhelt, akkor az negatívan befolyásolja a felhasználó által tapasztalt szolgáltatásminőséget. A probléma orvosolható a 6to4 megoldás egy „továbbfejlesztett” változata a 6rd (RFC 5969) használatával. (Ami azonban NEM helyettesíti a 6to4 megoldást!)

[10] Az erre irányuló draft RFC nyomon követhető: https://tools.ietf.org/html/draft-ietf-v6ops-6to4-to-historic. A 6-os verzióban még a teljes 6to4 megoldásról (RFC 3056) szó volt, 8-as verziótól már egyre inkább csak az RFC 3068 szerinti 192.88.99.0/24 anycast prefixről. Ezt végül 2015. májusában elfogadták, mint RFC 7526.

Biztonság

A 6to4 megoldás biztonsági kérdéseivel külön RFC foglalkozik (RFC 3964). A 6to4 megoldás természetéből adódóan érzékeny a szolgáltatásmegtagadás (DoS: Denial of Service) és a címhamisítás (address spoofing) típusú támadásokra. A legtöbb biztonsági problémát a 6to4 következő két jellemzője okozza:

1. Egy 6to4 routernek el kell fogadnia és ki kell bontania bármely más 6to4 routertől (6to4 hostot is beleértve) vagy relaytől származó forgalmat.
2. Egy 6to4 relaynek el kell fogadnia bármely natív IPv6 node forgalmát.

Az RFC-ben meghatároztak olyan szabályokat, hogy egy 6to4 routernek, illetve relaynek milyen ellenőrzéseket kellene végrehajtania. Például egy 6to4 router (a 6to4 hostot is beleértve) ne engedjen meg olyan forgalmat, aminek például:

1. IPv4 címe privát, broadcast vagy valamilyen lefoglalt tartományba tartozik
2. IPv4 forráscíme nem egyezik meg azzal, ami a 6to4 prefixében van
3. IPv6 címe nem globális IPv6 cím (hanem például link-lokális, stb.)
4. 6to4 prefixe nem egyezik meg a sajátjával.

Hasonlóan 6to4 relayre is vannak ilyenek. Például a fentiek közül az első három arra is vonatkozik. Kifejezetten 6to4 relayre vonatkozó tiltás, hogy ne fogadjon el 6to4 routertől olyan forgalmat, aminek az IPv6 célcímében 6to4 prefix szerepel (és nem natív IPv6 cím).

Sajnos ezeknek a szabályoknak a betartását a 6to4 implementációk nem mindig ellenőrzik (kényszerítik ki).

Az RFC számos támadási lehetőséget leír, amelyeket nem részletezünk, csak megjegyezzük, hogy a 6to4 sajnos sok lehetőséget ad a támadásra, ezért aki használni szeretné, annak mérlegelnie kell a kockázatokat. Az OpenBSD operációs rendszerben például biztonsági megfontolásból nem implementálták a 6to4 megoldást: http://www.securityfocus.com/columnists/459.

2.6.5. A 6to4 „továbbfejlesztései”

Teredo

A 6to4 megoldás használatát korlátozza, hogy használatához a 6to4 pseudo-interface számára publikus IPv4 cím szükséges, de ma rengeteg felhasználó privát IP-című gépet és valamilyen NAT megoldást használ. A *Teredo*t (RFC 4380) kifejezetten NAT-on keresztüli működésre tervezték. Ezt úgy éri el, hogy a Teredo kliensek a jól ismert (kliensbe beépített) címeken (3544-es UDP port) elérhető Teredo szerverektől megszerzett információ alapján IPv4 fölött UDP használatával építenek ki alagutat a megfelelő *Teredo relay* felé, amelynek natív IPv6 kapcsolata van. A Teredo a 2001::/32 prefixű IP címeket használja, ahol a következő 32 biten a Teredo szerver IPv4 címe található, majd a kommunikáció egyéb paraméterei, így a NAT típusa, és speciális kódolással a NAT eszköz publikus IPv4 címe, valamint az azon használt UDP portszám is.

Mélyebben nem foglalkozunk vele, mivel egyrészt csak *végső megoldás*nak (last resort) szánják arra az esetre, ha sem natív IPv6, sem 6to4 nem használható, másrészt pedig már tervezik lekapcsolni a Teredo relayeket[11].

6rd

Szintén a 6to4 negatív tulajdonságainak kiküszöbölésére született a 6rd (RFC 5969) megoldás. A 6rd fő előnyének szánták, hogy csak az adott szolgáltató hálózatán belül és kontrollja alatt működik, és itt szállítja IPv4 fölött az előfizetők IPv6 forgalmát. Ennek megfelelően nem a 2002::/16 prefixet, hanem a szolgáltató saját IPv6 prefixét használja. Ez azért előnyös, mert míg a 6to4 esetén kiszámíthatatlan, hogy melyik 6to4 relayen keresztül megy át egy csomag a natív IPv6 hálózatból az IPv4 hálózatba, itt a szolgáltató prefixe ezt egyértelműen meghatározza. Mindez garantált elérhetőséget és szolgáltatás minőséget biztosít az előfizetőknek. Amennyiben tehát egy szolgáltató szeretne a segítségével „majdnem igazi” IPv6 internet előfizetést nyújtani, akkor arra a célra kiválóan alkalmazható. Viszont a

[11] A javaslat a következő dokumentum 8. oldalán: http://www.ietf.org/proceedings/88/slides/slides-88-v6ops-0.pdf

6rd-t csak a szolgáltató tudja a hálózatában bevezetni, a szolgáltató nélkül a megoldás használhatatlan, tehát a 6rd NEM képes a 6to4 megoldást helyettesíteni. Megjegyezzük, hogy ez a megoldás is egyre inkább elavultnak számít, ma már egyértelműen natív IPv6-ot szoktak bevezetni.

2.7. A 6in4 megoldás

Ezt a megoldást akkor használjuk, ha IPv6 szigetek csak IPv4 hálózaton keresztül tudnak egymással kommunikálni (IPv6-over-IPv4, lásd: RFC 4213).

Ekkor az IPv6 csomagokat IPv4 csomagokba ágyazva visszük át az IPv4 hálózaton. Az IPv4-ben a 41-es protokollazonosítót használjuk az IPv6 csomagok azonosítására. (Amint az IP fölött a Protocol mezőben a TCP protokollt a 6, az UDP-t a 17, az ICMP-t pedig az 1 érték azonosítja.) A be- és kicsomagolást az IPv6 szigetek határán levő átjárók végzik.

A megoldás nagyon hasonlít a 6to4 technikára, azzal a különbséggel, hogy az alkalmazott tunnelek kizárólag manuális módon hozhatóak létre. Ennek a módszernek legnagyobb előnye a biztonság, hiszen csak előre beállított tunneleket alkalmaz, amelyek konfigurálását mindkét végponton el kell végezni. Ez ugyanakkor hátránya is, hiszen együttműködést és munkát igényel mindkét féltől. Problémát okoz még, ha nincs mindkét oldalnak állandó publikus IPv4 címe, hiszen ekkor minden címváltozáskor módosítani kell a tunnel másik végpontján lévő eszköz beállításait. A módszer ennek ellenére elterjedt; a tunnel brókerek is ezt a megoldást alkalmazzák, és általában valamilyen kiegészítő szoftver segítségével oldják meg a beállítások (tunnel végpont címek) automatikus módosítását. (És természetesen más megoldások részét is képezi, lásd például: DS-Lite.)

Ajánlott irodalom

Az ajánlott irodalom célja az, hogy az egyes témakörök iránt mélyebben érdeklődő hallgatóknak legyen hol tájékozódniuk. Elolvasásuk a mérésre való felkészüléshez nem szükséges.

[1] Lencse Gábor, Répás Sándor, Arató András: IPv6 és bevezetését támogató technológiák, 1. kiadás, HunNet-Média Kft., Budapest, 2015. ISBN: 978-963-12-3272-1, DOI: 10.18660/ipv6-b1.2015.9.1, http://ipv6ready.hu/konyv/

[2] S. Ferdous, F. Chowdhury and J. C. Acharjee, "An extended algorithm to enhance the performance of the current NAPT", in *Proc. International Conference on Information and Communication Technology 2007* (ICICT'07), Dhaka, Bangladesh, March 7-9, 2007, pp. 315-318, DOI: 10.1109/ICICT.2007.37540

[3] G. Lencse, "Estimation of the port number consumption of web browsing", *IEICE Transactions on Communications,* vol. E98-B, no. 8, (August, 2015) pp. 1580-1588. DOI: 10.1587/transcom.E98.B.1580

[4] M. Boye, "Netfilter connection tracking and NAT implementation", in *Proceedings of Seminar on Network Protocols in Operating Systems*, Department of Communications and Networking, Aalto University, 2013, pp. 34-39. http://urn.fi/URN:ISBN:978-952-60-4997-7

[5] G. Lencse and A. G. Soós, Design, "Implementation and Testing of a Tiny Multi-Threaded DNS64 Server", *International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems*, vol. 5, no. 2, (March 2016), pp. 68-78, DOI: 10.11601/ijates.v5i2.129

[6] G. Lencse, J. Palet Martinez, L. Howard, R. Patterson, "Pros and Cons of IPv6 Transition Technologies for IPv4aaS", Internet Draft, October 6, 2018, https://tools.ietf.org/html/draft-lmhp-v6ops-transition-comparison-03

[7] D. Liu, H. Deng, "NAT46 considerations draft-liu-behave-nat46-02", [Online]. Available: https://tools.ietf.org/html/draft-liu-behave-nat46-02

[8] A. Yourchenko, "OpenWRT feed with stateless NAT46 kernel module", [Online]. Available: https://github.com/ayourtch/nat46

[9] Bob Weeink, "Cisco ASA NAT46 (IPv4-to-IPv6)", [Online]. Available: https://supportforums.cisco.com/discussion/11809096/cisco-asa-nat46-ipv4-ipv6

[10] Brocade, *Brocade ServerIron ADX NAT64 Configuration Guide: Supporting Brocade ServerIron ADX version 12.5.02*, Part Number: 53-1003448-01 [Online]. Available: http://www.brocade.com/downloads/documents/html_product_manuals/SI_12502_NAT64/

[11] G. Lencse and S. Répás, "Performance analysis and comparison of 6to4 relay implementations", *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 9. (September, 2013), pp. 13-21. DOI: 10.14569/IJACSA.2013.040903