TCP/IP - 1 óra alatt

Petrényi József

1.1 verzió, első kiadás

Microsoft Magyarország 2010

TARTALOMJEGYZÉK

1 Bevezetés

Valamikor a Sams kiadónak voltak olyan könyvei, hogy 'Kupáld ki magad ebben, meg abban 24 óra alatt!'[1]. Belegondoltam, hogy ilyet én is tudok.
Mármint nem írni, hanem ekkorát hazudni egy címben.
Sőt.

Amit most vagy a kezedben tartasz, vagy a monitoron nézel, az egy kellően magasról áttekintett leírása a TCP/IP protokollcsaládnak, mely írást 1 óra alatt... el lehet olvasni, ha kellően jártas vagy a gyorsolvasásban.

Hogy mi értelme van ennek a kicsi könyvnek? Nézd, nemrég írtam egy kétkötetes, durván 500 oldalas könyvet ugyanerről a témáról. Ha ezt a mostani könyvet repülőgépről szemlélt tájképhez hasonlítom, akkor az a másik egy vadászvizslával fél centiről végigszaglászott esőerdő.
Csakhogy egyáltalán nem biztos, hogy már az első találkozáskor mindenki rögtön annyi részletre lesz kíváncsi. Lehet, hogy először csak egy átfogó képet szeretne kapni, a többi meg majd jöjjön, ha már kialakult valamiféle kép.

Az ő számukra íródott ez a rövid összefoglaló. Terjedelemben durván egy tizede a részletes könyvnek - ennek ellenére a lényeg benne van. Ha nulláról indulsz, az elv megértését megcélozva, akkor ez a te könyved.

A nagy testvér, a TCP/IP Alapok című kétkötetes könyv elérhetősége:
http://mivanvelem.hu/letoltheto-konyvek/

[1] Sams Teach Yourself Atomphysics in 24 Hours

2 TCP/IP v4

Hatalmas újdonságot fogok közölni: a TCP/IP rétegekből áll.
Leültél? Ja, már eddig is ültél? Hja, ez egy ilyen sokat ücsörgős szakma.

A rétegmodellnek ugyanis rengeteg előnye van. Egyik oldalról elég a réteghatáron egyeztetni az átadott paramétereket - és nem kell foglalkozni azzal, hogy mi is zajlik az egyes rétegeken belül. Másik oldalról rengeteg rétegen belüli eljárást lehet definiálni, arra kell csak figyelni, hogy a réteghatárra mindegyik a megfelelő paramétert adja ki.

Így épül fel a TCP/IP: kismillió szabvány, protokoll, melyek előírják mind a belső, mind a rétegek közötti folyamatokat.

Azt írtam, szabvány. Az informatikában a szabványt RFC-nek hívják. Ez első blikkre fura lehet, mivel az RFC eredetileg azt jelenti, hogy Request For Comment. Ha szószerint fordítjuk, akkor lehetne ajánlásnak, véleménykérésnek is fordítani - de ez a mi szintünkön már kábé annyira véleménykérés csak, mint amikor a feleség megkérdezi a férjét, hogy kövérnek tartja-e? Csak egy válasz létezik.
Rögzített folyamatokban - pl. ITIL - az RFC annyit tesz, hogy szeretnénk formába önteni egy részfolyamatot, egy módszert. Ezt leírjuk és megköpködtetjük az érintettekkel. Ezt hívják véleménykérésnek. Ha mindenki elmondta a véleményét, az eredeti dokumentumokon átvezették a módosításokat, akkor áll össze a szabvány - melyet az IETF-nél ugyanúgy neveznek, mint a véleménykérő iratot, az RFC-t. Csak éppen ekkor már kötelező érvényű. (Vannak kivételek, és igazából ez az egész messze nem ilyen egyszerű, de egyelőre bőven elég ennyi absztrakció.)

> IETF: Internet Engineering Task Force. Ez a társaság gondozza mind az internet, mind az internethez köthető rendszerek szabványait.
> IANA: Internet Assigned Numbers Authority. Ők a felelősek mindenért, ami szám alakú és az internettel kapcsolatos.

A könyvben rengeteg RFC-re fogunk hivatkozni. Ez a csontváza a kommunikációnak az informatikában. Minden, hangsúlyozom, minden erre épül - eltekintve az abszolút egyedi, nem szabványos alkalmazásoktól. Az egyes cégek fejlesztői az RFC-k alapján fejlesztenek. (Nem ritka az sem, hogy egy cég a saját módszereiből fogadtat el RFC-t.) Az RFC-k garantálják, hogy különböző cégek fejlesztései képesek legyenek kommunikálni egymással.

Jó hír, hogy az RFC-k publikusak. Még jobb hír, hogy rengeteg helyen megtalálhatók a neten. Kevésbé jó hír, hogy viszonylag kevés bennük a humoros jelenet. Meglehetősen száraz anyagok, na. De ez az egyértelműség miatt muszáj is.

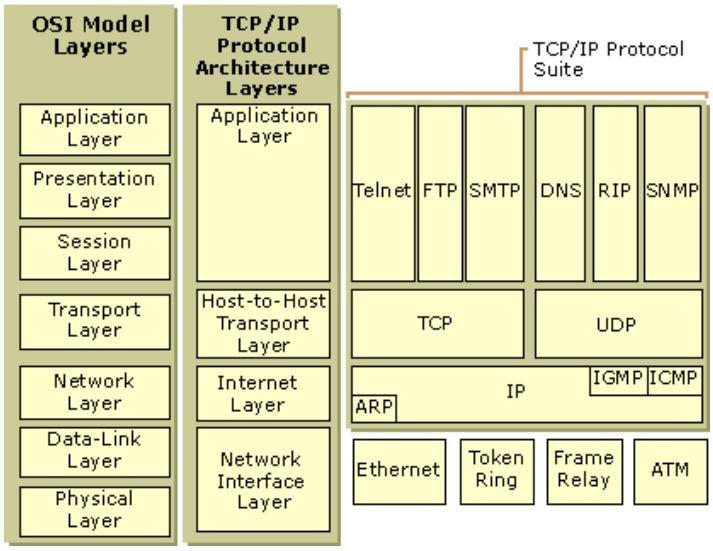
> A teljesség szándéka nélkül néhány link:
> http://en.wikipedia.org/wiki/Request_for_Comments
> http://www.ietf.org/
> http://www.faqs.org/rfcs/
> http://www.rfc-editor.org/rfc.html

Mivel a könyv elsősorban üzemeltetőknek készül, így nem fogunk leásni az RFC-k legmélyére. De nem is hagyhatjuk figyelmen kívül azokat. Több ponton is belefuthatunk olyan szituációkba, amikor jó lenne értenünk, mi is áramlik éppen keresztül a dróton.

- Például nem akarunk engedni minden HTTP forgalmat a tűzfalon. Jó lenne tudni ilyenkor, hogyan, milyen jellegzetességénél lehetne megfogni a forgalmat. Mit jelentenek a grafikus felületeken az egyes lehetőségek?
- De hasonlóan fontos az is, hogy ne idegenkedjünk a hálózati forgalom elemzésétől. Ne csak bottal merjük megpiszkálni a netmont, hanem értsük is, amit mutat.
- Különösen könyvírás közben, amikor az ember próbál elmélyedni a részletekben és több forrásmunkát is átolvas, ilyenkor lehet rádöbbenni, hogy akinek két órája van, az sohasem tudja, mennyi a pontos idő. Nem ritka, hogy az egyes források teljesen ellenkező tényeket állítanak, vagy egyszerűen csak igyekeznek homályosan fogalmazni. Ilyenkor külön öröm az, hogy az ősforrás elérhető a neten - az RFC-knél egyértelműbb megfogalmazást sehol nem találunk.

2.1 Rétegek

In medias res. Azaz bele a rés közepébe.



2.1. ábra Rétegek a Microsoft TCP/IP megvalósításában

A bal oldali oszlop az ún OSI modell rétegeit mutatja. Ezzel most túl sok dolgunk nem lesz. Koncentráljunk inkább a mellette lévő TCP/IP architektúrára. Szépen láthatjuk az egymásra épülő négy réteget, illetve jobbra mellettük azt, hogy az egyes rétegekben például milyen protokollok dolgoznak. (A választék messze nem teljes.)

Nevezzük nevükön is az egyes rétegeket:

Network Interface réteg: Ez van a legközelebb a kóbor elektronokhoz. Ebben a rétegben történik meg az egyes csomagok tényleges elszállítása a feladótól a címzettekig. Az ábrán is látható, hogy itt elsősorban a hálózat hardveres megvalósításához kötődő szabványokat találjuk.

Internet, vagy IP réteg: A címzés a borítékon. Ebben a rétegben kap címet mind a feladó, mind a címzett. Ez a réteg felelős azért, hogy a csomag tudja, merre kell mennie, amíg célba nem jut.

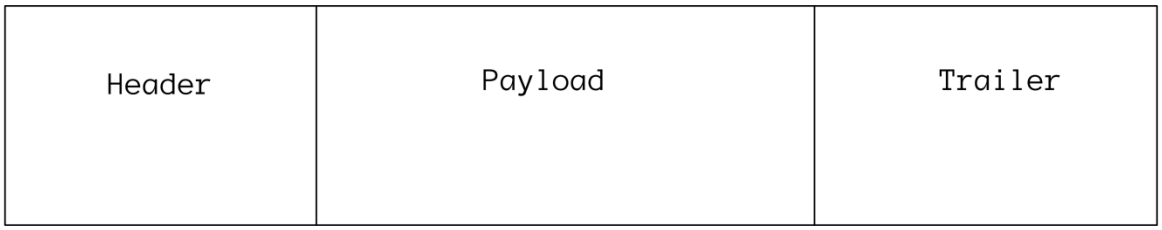
Transport réteg: Ez a réteg kapja meg az elszállítandó adatokat és állítja össze belőlük a csomagokat.

Alkalmazás réteg: Ez egy meglehetősen bonyolult réteg. Itt mindazon szabványokat találjuk meg, melyek az egyes konkrét alkalmazásokból illetve alkalmazásokba érkező adatfolyamok előfeldolgozásához szükségesek.

Most egy kicsit zavaros rész következik, de igyekszem érthető lenni.

Doboz.

Az információk a hálózaton dobozszerűségekben utaznak. Értem ezalatt azt, hogy az egységeknek általában van alja, teteje - és van tartalma.



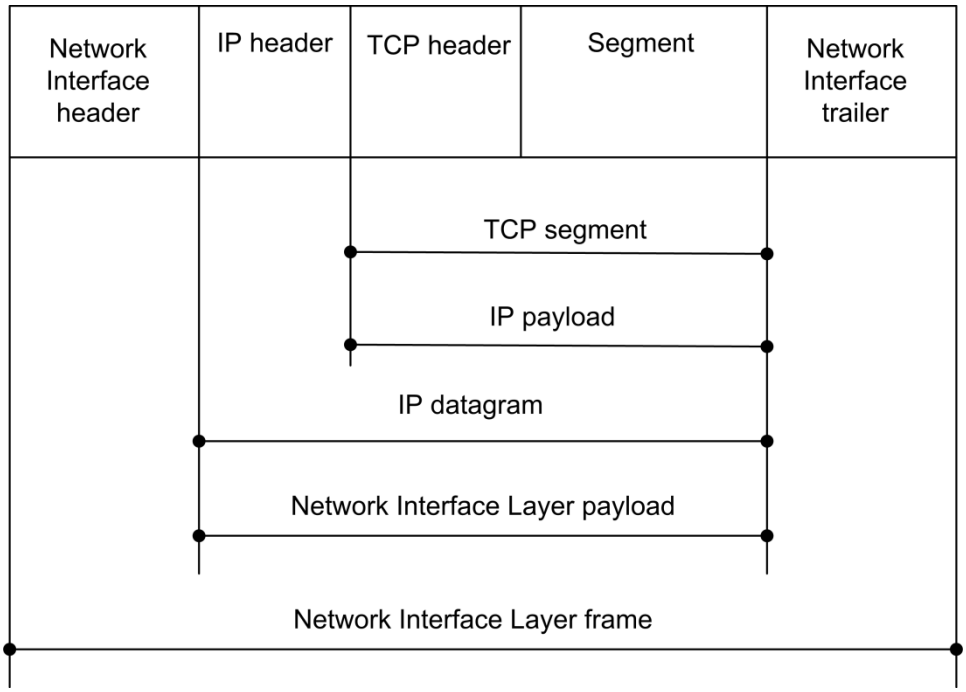
2.2. ÁBRA ÁLTALÁNOS CSOMAGOLÁSI SÉMA

A header az a bitkupac, ahol a csomagra vonatkozó információk utaznak.
A payload az a szállított tartalom.
A trailer pedig a csomag végét jelző bitsorozat.

- FRAME, AVAGY KERET: A Network Interface rétegre jellemző. Azért hívják keretnek, mert mindkét oldalról le van határolva, azaz ténylegesen is van neki fejléce és lezárása.
- DATAGRAM, AVAGY DOBOZ: Az Internet rétegre jellemző, de előfordul a Transport rétegben is. Nincs lezárása, ehelyett a fejléc tartalmazza a csomag hosszát. Fontos jellegzetesség, hogy kompakt információt szállít, azaz a tartalom ömagában is értelmezhető.
- SEGMENT, AVAGY SZEGMENS: A Transport rétegre jellemző. Formailag ugyanaz, mint a datagram, de az általa szállított információ nem kompakt. Képzeljünk el egy hosszú adatfolyamot, melyet felszeleteltünk és a darabokat raktuk bele egy-egy csomagba. A csomag tartalma darabonként nem értelmezhető. csak ha újból összerakjuk a teljes folyamot.

Látható, hogy markáns különbségek vannak az egyes tipusok között. Csakhogy ez nem mindig lényeges. Van amikor csak egy entitásról beszélünk, melyet továbbítani kell - és ilyenkor a továbbítás módja a lényeges, nem az entitás felépítése. Ekkor egyszerűen csak csomagnak (packet) nevezzük ezt a bigyót.

Nagyon fontos megérteni, hogy a csomagok egymásba vannak csomagolva.



2.3. ÁBRA CSOMAGOK HÁTÁN CSOMAGOK

Ezen menjünk most végig. A Transport rétegben keletkezik egy csomag, egy TCP szegmens. Ennek van fejléce (TCP header) és tartalma (segment, azaz adatfolyam-darab). Ez a csomag kerül át az Internet rétegbe, ahol ő, azaz a teljes Transport csomag lesz az IP datagram tartalma. Az Internet réteg hozzáteszi a saját fejlécét, majd az így keletkező datagramot passzolja le a Network Interface rétegnek. A séma ugyanaz, a Network Interface keret tartalma a teljes IP datagram lesz, ennek az elejére jön a Network Interface fejléc, illetve a keretet fizikailag is lezáró trailer.
Amikor ez az egész megérkezik a címzetthez, ő az egyes szintek fejlécei alapján kódolja vissza az üzenetet és értelmezi a legvégül kapott adatszegmenst.

> A kompaktság értelemszerűen csak rétegszinten értendő. Azaz ha nézek egy TCP szegmenst, akkor az abban lévő payload az egy adatfolyam része, önmagában nem értelmezhető, nem kompakt. Ellenben ha megnézem azt az IP datagramot, amelyikbe az előző TCP szegmenst csomagoltam, akkor ez már kompakt lesz, hiszen egy jól megfogható dolog van benne, a TCP szegmens. Nem megyünk bele, hogy a TCP szegmens odabent nem kompakt.

2.2 Hálózati eszköz (Network Interface) réteg

Mint írtam, ebben a rétegben elsősorban hardver megoldásokkal találkozunk. Teljesen más elven szállítja a csomagokat például egy Ethernet vagy egy Token-Ring, neadjisten FDDI hálózat. Érthetően teljesen mások a hálózatok karakterisztikái is.

Ez egy szép nagy dzsungel. Jelen bevezetőnek nem is célja ezt feltérképezni. Foglalkozzunk most csak az Ethernet hálózatokkal.

Ha azt hiszed, ezzel egyből ki is jöttünk a dzsungelből, tévedsz. Ethernetből is van sokfajta keret.

Vágjunk rendet.

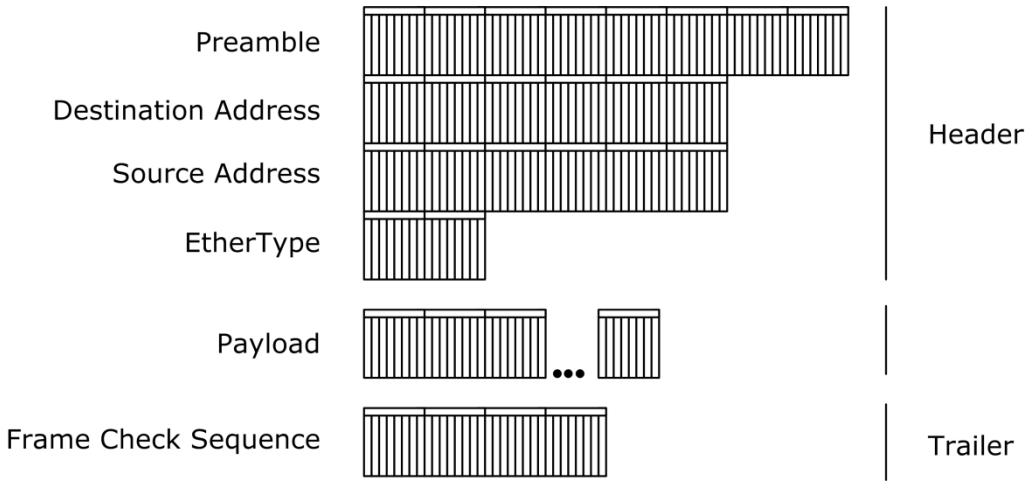
Az Ethernet egy csomagtovábbítási technika, mely az ún. Carrier Sense Multiple Access with Collision Detection, azaz CSMA/CD elven alapul.
Elmagyarázom. A fizikai hordozó közegen (csavart érpár, üveg) egyszerre csak egy csomag tekerhet. Ha a feladó küldeni akar, akkor belekiabál a csőbe, hogy üres--e? Ha igen, akkor küld. Ha nem, akkor vár. Ha ketten kiabálnak bele egyszerre, akkor mindketten várnak.

Maga a küldött csomag - illetve nevezzük most már nevén - az Ethernet keret többféle lehet. Ez elsősorban ott jelenik meg, hogy pontosan mi is kerül az egyes fejlécekbe, mi kerül a payloadba, illetve a trailerbe - azaz milyen struktúrába csomagoljuk az információt.

Megint nem fogok elveszni a részletekben, itt példaként csak a legelterjedtebbel, az Ethernet II szabvánnyal fogunk foglalkozni. (Futottak még az Ethernet 802.3, illetve Ethernet SNAP.) Annyit azért jegyezzünk meg, hogy ezeket nem lehet keverni. Az Ethernet II hálózati kártya nem fog tudni mit kezdeni az Ethernet 802.3 kerettel - hiszen ő csak egy struktúrát ismer, az érkező keret meg másikat használ.

2.2.1 Ethernet II frame.

RFC 894



2.4. ábra Ethernet II keret

Preamble: Egy bitsorozat, mely jelzi, hogy itt indul a keret. A network monitorozó eszközök nem mutatják.

Destination Address: A címzett címe.

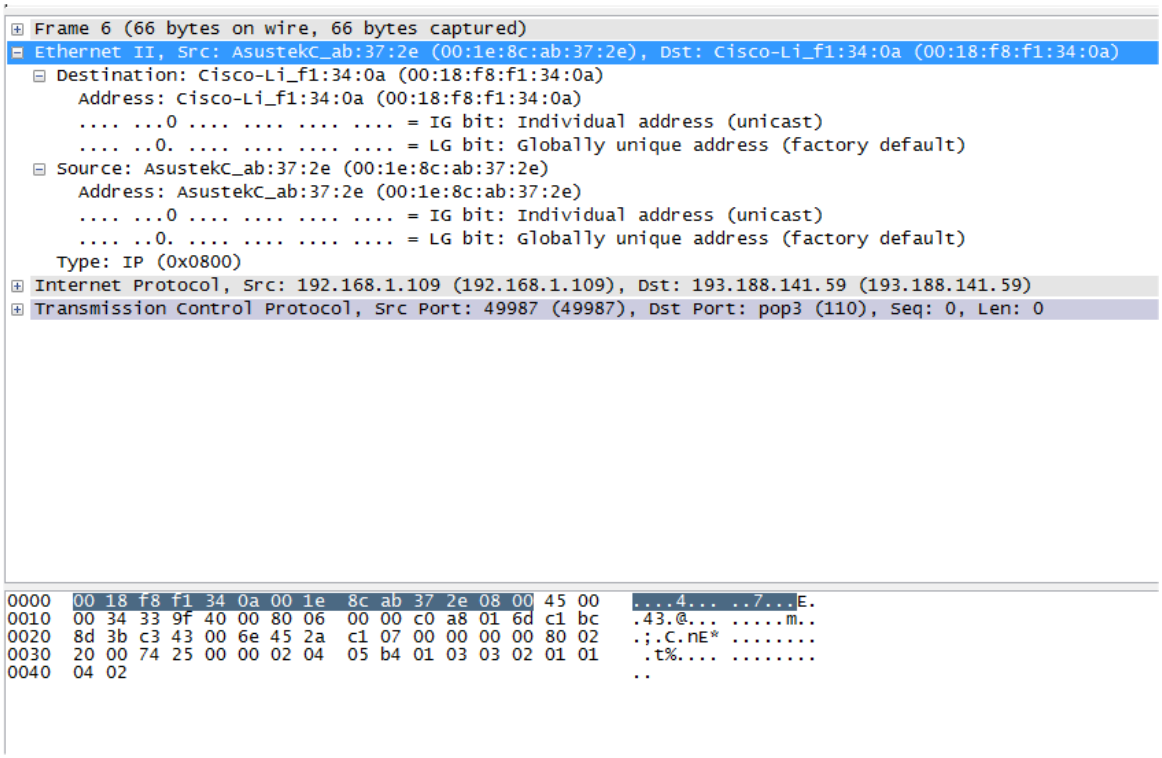
Source Address: A feladó címe.

EtherType: A payloadra vonatkozó azonosító kód.

- 0x0800 - IP datagram
- 0x0806 - ARP csomag.

Payload: Maga a szállított teher. A mérete alulról is, felülről is limitált: minimum 46, maximum 1500 bájt. Ez utóbbi értéket nevezik egyébként MTU-nak, Maximum Transmission Unit-nak is. (A mérete értelemszerűen a keret kialakításától függ. Más kerettípusnál más az érték is.)
Amennyiben a feladott payload mérete nem érné el a 46 bájtot, akkor kipótolják. (A minimális méret előírásának oka a megbízható ütközésdetektálás.)

Frame Check Sequence: Ellenőrző kód. A keretet, mint bitsorozatot (minusz FCS mező) elosztják egy 33 bites prímszámmal, és maga a 32 bites maradék lesz a 4 bájtos FCS érték. A feladó berakja az értéket, a címzett pedig elvégzi ugyanezt a műveletet - és amennyiben a kapott eredmény nem egyezik az FCS mező értékével, akkor eldobja a keretet.
A network monitorozó eszközök ezt a mezőt sem mutatják.



2.5. ÁBRA ETHERNET II KERET MONITOROZVA

Így néz ki egy monitorozó eszközzel (Wireshark) elkapott Ethernet II keret. Látható, nem hazudtam. Minden ott van a helyén. A keret fejléce ki lett bontva, a payload (IP datagram, illetve az abba csomagolt TCP szegmens) most éppen nem.

Nem bírom megállni, hogy ne rohanjak előre. Habár még nem beszéltünk ilyesmikről, de azért folyosói pletyka szinten csak tudunk valamit már az informatikából. Próbáljunk meg mindent kiolvasni a fenti keretből.

- TRANSPORT RÉTEG: A forrás port 49987, ez tipikus kliens port. Nem tartozik a wellknown portok közé. A cél port 110, ez bizony egy POP3-as kérés. A seq és a len egyaránt nulla, az ack alacsony, így ez a kapcsolat első lépése. (Az ún. SYN lépés.)
- INTERNET RÉTEG: A feladó IP címe 192.168.1.109, a címzetté 193.188.141.59.
- NETWORK INTERFACE RÉTEG: A feladó egy Austek hálózati kártya, a MAC címe: 00:1e:8c:ab:37:2e. A címzett egy Cisco hálózati kártya, a MAC címe: 00:18:f8:f1:34:0a

Vissza a hálózati rétegbe. Mik is ezek a MAC címek?

Minden hálózati kommunikációra képes eszköznek van egy azonosítója. Ez az azonosító világszerte egyedi és bele van égetve az eszközbe. (Viszont a legtöbbször elmaszkolható.) A MAC cím - mint az ábrán (*2.4. ábra Ethernet II keret*) is látható, hat bájtos.
Az első három bájt a gyártó azonosítója. A második három bájt pedig a gyártón belül az eszköz egyedi kódja.

A TCP/IP hálózati kommunikációban többször is történik valamilyen névfeloldás. Amikor azt mondjuk, hogy kapcsolatba szeretnénk lépni például az isitchristmas.com webalkalmazást futtató szerverrel, akkor először meg kell határoznunk az IP címét. Ez az alkalmazás réteg dolga. Ha megkaptuk a 66.33.220.210 címet, a következő lépésben meg kell határoznunk az IP címhez tartozó hálózati kártya MAC címét. Ugyanis, mint írtam, a csomagok szállítása ténylegesen a hálózati rétegben történik, itt viszont a MAC címek játszanak fő szerepet.

Az első névfeloldásra (URI -> IP) számtalan módszer létezik. (Lokális cache, DNS, host fájl, WINS.)

A második névfeloldást (IP -> MAC) az Address Resolution Protocol segítségével végezzük el.

2.2.2 Address Resolution Protocol - MAC címek keresése

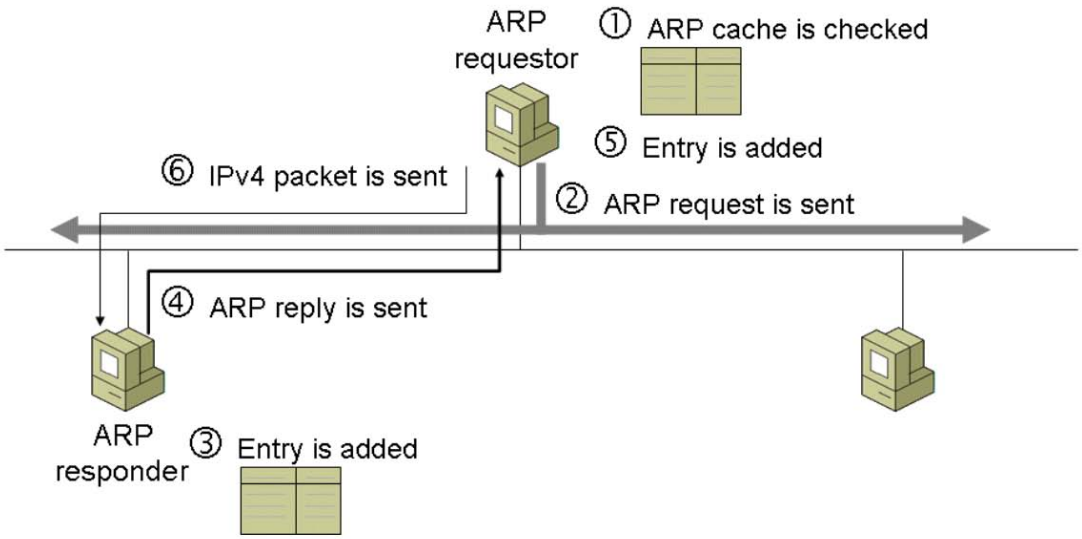
RFC 826

Mint jeleztem is korábban, az Ethernet keretben kétfajta payload utazhat, IP datagram és ARP csomag. Maga az ARP csomag szintén kétféle lehet: kérdés vagy felelet.

- ARP Request: A kérdező küld egy ARP broadcast üzenetet. Ebben szerepel a saját IP címe, a saját MAC címe és a kérdezett IP cím. A kérdezett MAC cím mező üres.
- ARP reply: A kérdezett IP című gép magára ismer, beleírja a válaszba a kérdezett MAC címet, majd immár célzott üzenetként visszaküldi.

Látható, hogy a névfeloldásnak van némi korlátja: a broadcast általában nem tud kitörni a lokális alhálózatról, tekintve, hogy a routerek nem engedik át. De nem is nagyon van rá szükség, mert majd az Internet rétegnél látjuk, hogy a feladó nem a végső címzett MAC címére lesz kíváncsi, hanem az útvonal első ugrását jelentő gép (alhálózaton kívül eső célpont esetén a router) MAC címére.

Nagyjából ez az elv. A gyakorlatban ez meg van bolondítva azzal, hogy a feloldott MAC címek bekerülnek a gépek lokális ARP gyorstárolóiba, hogy ne kelljen mindig broadcastolni.



2.6. ÁBRA ARP NÉVFELOLDÁS

A valóságban a gyorstárazás okoz némi problémát, de ezeket ebben a könyvben nem tárgyaljuk.

2.3 INTERNET RÉTEG

> RFC 791

Jogos lehet a kérdés, miért is van szükség kétféle címre? Ha minden hálózati kártyának a világon egyedi azonosítója van, miért nem lehet pusztán ezt a címet használni? Elég egyedi, nem?
Nos, egyedinek egyedi... de meglehetősen rögzített. Márpedig mi, amikor építjük a rendszereinket, szeretnénk olyan struktúrát összerakni, amely a céljainknak megfelel. Alhálózatokat szeretnénk létrehozni, számítógépeket akarunk elkülöníteni egymástól, alhálózatokat akarunk gerincvonalakkal összekötni, telephelyeink, központjaink vannak. Ehhez kell egy rugalmas címzési struktúra.

Ezt a struktúrát valósítja meg, ezt a struktúrát működteti az Internet réteg.

> Jelenleg az IP version 4 a legelterjedtebb szabvány, de már kint van és kezd terjedni az IP version 6 szabvány is. Erre később külön is kitérek.

Az egész struktúrának alapja az IP cím. Az IPv4-ben egy IP cím az egy 32 bites bináris számsorozat. Fontos, hogy ez önmagában nem értelmezhető: minden IP címhez

tartozik egy másik 32 bites bináris számsorozat, az alhálózati maszk (subnet mask) - ez adja meg, hogy az IP címből hány bit határozza meg az alhálózat azonosítóját és hány magának a node-nak a címét.

> Gyors fogalomtisztázás:
> Node : Egy hálózati csatolóeszköz.
> Host : Egy számítógép, melyben akár több hálózati kártya, azaz node is lehet.

Például a 192.168.1.164 IP cím és a 255.255.255.0 alhálózati maszk a következő dolgokat határozza meg:

- A hálózat azonosítója : 192.168.1
- A hálózaton belül a node azonosítója : 164.

Miért? A bináris matek miatt:

- 192.168.1.164 -> 11000000 10101000 00000001 10100100
- 255.255.255.0 -> 11111111 11111111 11111111 00000000

Azt mondjuk, hogy amíg az alhálózati maszk értéke 1, addig tart a hálózat azonosító, amikor pedig 0, akkor ott már a node azonosítót kapjuk. Valamivel matekosabban úgy is mondhatnám, hogy az IP cím és az alhálózati maszk bináris szorzata (AND) adja a hálózat azonosítóját, illetve az alhálózati maszk negáltja (NOT) szorozva az IP címmel adja a node azonosítóját.
Szokták ezt úgy is jelölni, hogy nem írják le az alhálózati maszk minden egyes bitjét, csak megadják, hogy hány darab egyes van benne. (Remélem, az nem lep meg, hogy az alhálózati maszk - az egészen extrém esetektől eltekintve - mindig balra zárt, azaz balról töltődik fel egyesekkel.)

Ekkor a fenti példa így néz ki: 192.168.1.164/24.

Nyilván az sem nagy meglepetés, hogy ebben az esetben a hálózaton 256 különböző node lehet. (Igazából 254, mert a két szélső cím fenn van tartva a hálózat beazonosítására, illetve a broadcast címre.)

Régebben az IP címeket osztályokba (classful) sorolták, voltak A, B, C, D, E kategóriás címek. Ma már classless világban élünk, de ez csak az első 3 kategóriára igaz. A másik két kategória megmaradt.

2.1. TÁBLÁZAT

Osztály	Mettől	Meddig	Micsoda
D	224.0.0.0	239.255.255.255	Multicast[2]
E	240.0.0.0.	254.255.255.255	Vésztartalék

[2] A multicast a pont - multipont kommunikációt jelenti.

Ha két node eltérő alhálózaton van, azaz különbözik a hálózati azonosítójuk, akkor közvetlenül nem tudnak kommunikálni egymással.

Nézzünk erre is egy példát:

Node1 (192.168.1.164/24):

- 192.168.1.164 -> 11000000 10101000 00000001 10100100
- 255.255.255.0 -> 11111111 11111111 11111111 00000000
- A hálózat azonosítója : 192.168.1
- A hálózaton belül a node azonosítója : 164.

Node2 (192.168.12.123/16):

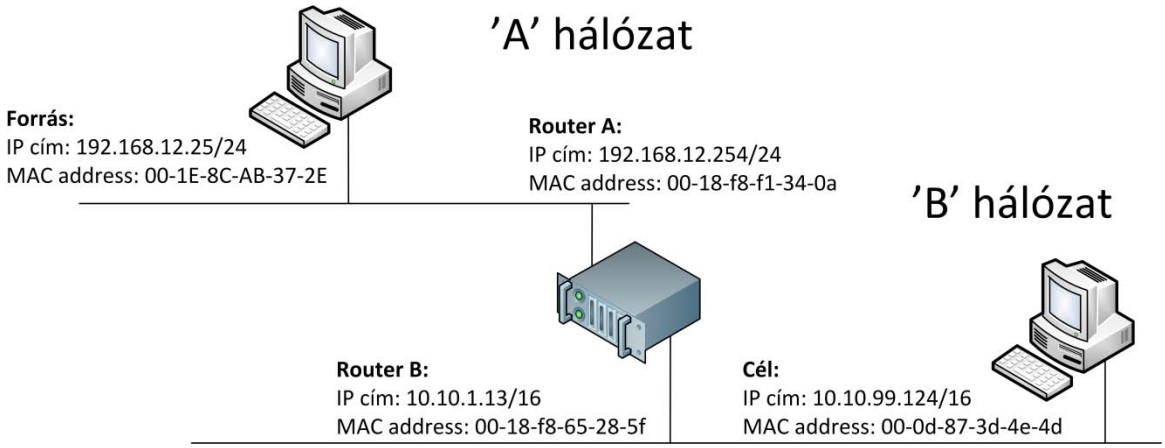
- 192.168.12.123 -> 11000000 10101000 00001100 01111011
- 255.255.0.0 -> 11111111 11111111 00000000 00000000
- A hálózat azonosítója : 192.168
- A hálózaton belül a node azonosítója : 12.123.

Mivel a két hálózati azonosító különböző, ez a két node nem fog tudni közvetlenül kommunikálni egymással. Ide router kell.

2.3.1 Routing

De mi is az a router?

A router az a host, mely több különböző hálózat között biztosít átjárást úgy, hogy mindegyik hálózatba belógat egy-egy hálózati csatolót.



2.7. ábra Routolási folyamat

A fenti ábrán egy teljesen egyszerű szituáció látható. Van két alhálózatunk, azokon egy-egy node . A kettőt egy router köti össze, melyben két hálózati kártya van.

Nézzük részletesen a folyamatot.

1) A Forrás névvel jelölt számítógépnek halaszthatatlan közlendője támad, melyet a Cél nevű számítógéppel szeretne megosztani. A hálózat egyszerű Ethernet.
2) A Cél nevű számítógép IP címét megszerzi valahonnan. (Vagy benne van a programban, vagy segítségül hívja a névfeloldási folyamatot és mondjuk egy DNS szerver megsúgja neki.)
3) Szomorúan veszi tudomásul, hogy abszolút más hálózatról van szó, tehát közvetlenül nem tudja elküldeni a csomagot.
4) Egy - később tárgyalandó algoritmussal - megszerzi annak a node-nak az IP címét, mely felelős az idegen hálózatba küldött csomagok továbbításáért. Ez jelen esetben a Router A lába lesz.
5) Az IP cím birtokában - az ARP segítségével - begyűjti a Router A MAC címét, és így már össze tudja rakni a küldendő csomagot:
    a. Source IP address : 192.168.12.25
    b. Source MAC address: 00-1e-8c-ab-37-2e
    c. Target IP Address : 10.10.99.124
    d. Target MAC address : 00-18-f8-f1-34-0a (!!)
6) Mivel egy fogadott csomag beazonosítása alulról felfelé történik, így a Target IP cím hiába a Cél számítógépé, de a Target MAC cím miatt a Router A magáénak fogja érezni a csomagot. Felszipkázza. A Target IP alapján értelmezi a feladatot, megkeresi, melyik lábán kell továbbítania a csomagot (figyelem, léteznek százlábú routerek is), az ARP segítségével begyűjti a Cél számítógép MAC cím értékét, majd összerakja a következő csomagot:
    a. Source IP address : 192.168.12.25
    b. Source MAC address: 00-1e-8c-f1-34-0a
    c. Target IP Address : 10.10.99.124
    d. Target MAC address : 00-0d-87-3d-4e-4d
7) A Target MAC cím miatt a Cél számítógép felveszi a csomagot, a Target IP cím alapján látja, hogy neki szól. Boldog. Miután megérkezett az összes csomag, összerakja az üzenetet, és az eddig tárgyalt módon válaszol.

Nagyon durván ennyi. De már most is látható, hogy van egy-két zavaros terület:

- Honnét is tudja a Forrás, hogy neki pont Router A számára kell elküldeni ezt a csomagot?
- Honnét is tudja a Router, hogy melyik lábán kell továbbküldenie a csomagot? És mi van akkor, ha a Cél számítógép nem kapcsolódik közvetlenül a Routerhez, hanem van köztük mondjuk még 5 darab köztes router is?

Az első kínzó kérdésre a választ a route tábla adja meg. Minden számítógépben van egy táblázat, mely arra vonatkozik, hogy egyes csomagokat merre kell továbbítani.

```
===========================================================================
Interface List
  8 ...00 1e 8c ab 37 2e ......  Realtek RTL8168B/8111B Family PCI-E GBE NIC
  1 ...........................  Software Loopback Interface 1
  9 ...02 00 54 55 4e 01 ......  Teredo Tunneling Pseudo-Interface
 13 ...00 00 00 00 00 00 00 e0  isatap.{47CE5CAA-223F-4CD4-9A17-D91D7DDC2066}
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0      192.168.1.1    192.168.1.101     20
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    306
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    306
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    306
      192.168.1.0    255.255.255.0         On-link     192.168.1.101    276
    192.168.1.101  255.255.255.255         On-link     192.168.1.101    276
    192.168.1.255  255.255.255.255         On-link     192.168.1.101    276
     192.168.99.0    255.255.255.0      192.168.1.2    192.168.1.101     21
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    306
        224.0.0.0        240.0.0.0         On-link     192.168.1.101    276
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    306
  255.255.255.255  255.255.255.255         On-link     192.168.1.101    276
===========================================================================
Persistent Routes:
  Network Address          Netmask  Gateway Address  Metric
     192.168.99.0    255.255.255.0      192.168.1.2       1
===========================================================================
```

Ez annak a gépnek a route táblája, amelyiken a könyvet írom. (A *route print* parancs adja ki.) A lista első felében a hálózati csatolók látszódnak, utána jön maga az útválasztási táblázat.

Nem kell megijedni, a táblázat csak elsőre tűnik bonyolultnak.
Az első két oszlop értékei tulajdonképpen a feltételeket jelentik, a harmadik/negyedik oszlopok értékei fogalmazzák meg az akciót, az ötödik oszlop értékei pedig prioritást befolyásolnak.

Nézzünk konkrét példákat.

Tételezzük fel, hogy a 192.168.1.123 node számára szeretnék küldeni egy csomagot. Merre induljak?
Mintaillesztéssel kezdjük. Végignézzük az első két oszlopot és megkeressük, hogy a megcélzott cím mely értékre illeszkedik a legszorosabban.

- 0.0.0.0 : Erre illeszkedik ugyan, mert erre minden illeszkedik, csak éppen meglehetősen lazán.
- 127.0.0.0 : Na, erre abszolút nem illeszkedik.
- 192.168.1.0 : Erre viszont egészen jól.

A további soroknál már nincs illeszkedés. Azaz két találatunk van, ezek közül az ötödik sorban lévő illeszkedik szorosabban. Nézzük, ehhez a sorhoz milyen akció tartozik: a csomagnak a 192.168.1.101 hálózati kártyán kimenve (ennek ugye akkor van értelme, ha több hálózati kártya is van a gépben) kell keresnie közvetlenül a 192.168.1.123 node-ot, mivel azonos alhálózaton vagyunk. Nincs router.

Nézzünk egy újabb példát. Legyen a megcélzott node a 192.168.99.15.
Megint mintaillesztés

- 0.0.0.0 : Még mindig jó.
- 127.0.0.0 : Erre megint nem illeszkedünk.
- 192.168.1.0 : Most erre sem.
- 192.168.99.0 : Hoppá.

A további soroknál már nincs illeszkedés. Megint két találatunk van, most a nyolcadik sorban lévő a szorosabb. Akció? A csomagnak a 192.168.1.101 hálózati kártyán kimenve kell keresnie a 192.168.1.2 node-ot. Valószínűleg ez lesz a router.

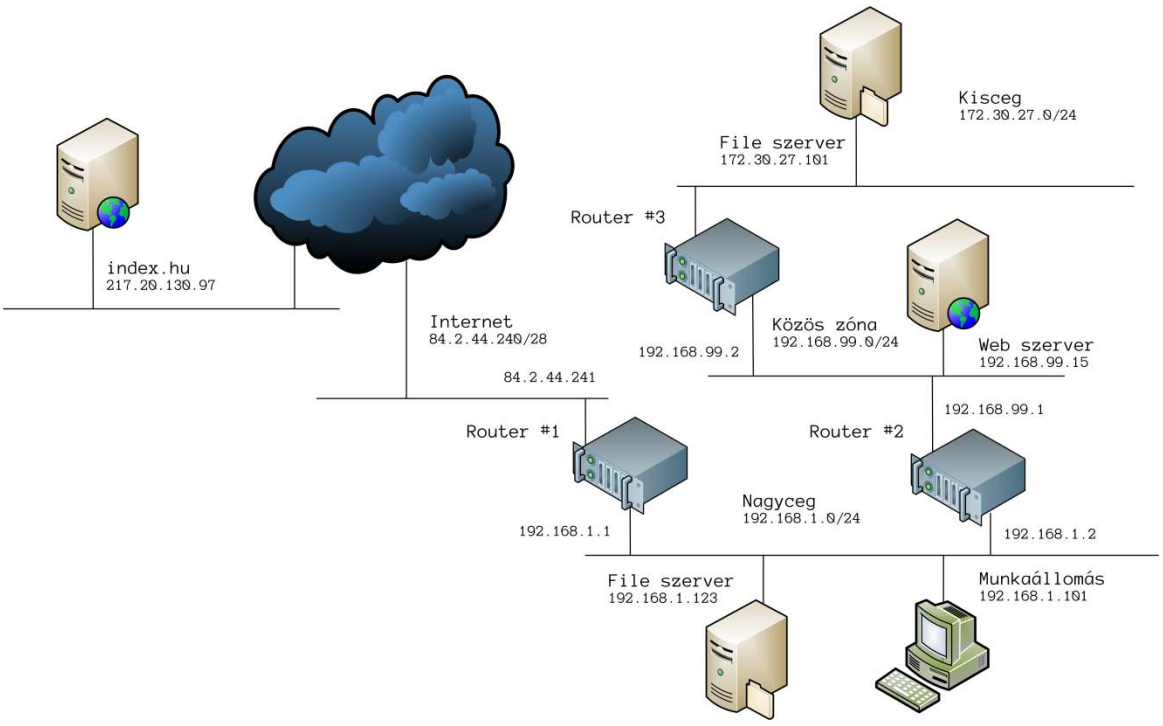
Utolsó példa. Próbáljuk most becélozni a 217.20.130.97 node-ot.

- 0.0.0.0 : Szokás szerint jó, de laza.

És vége. Semmi más nem illeszkedik. Akkor most mi lesz? A csomag a 192.168.1.101 hálózati kártyán fog kimenni és a 192.168.1.1 node-ot fogja keresni. Ez megint egy router.

Tehát egy olyan alhálózatban vagyunk, amelyikben két router is van. Az egyik szigorúan csak a 192.168.99.0/24 alhálózat felé routol (bár nem tudjuk, mi lehet még mögötte), a másik pedig minden egyéb alhálózat felé.

Próbáljuk meg elképzelni.



2.8. ÁBRA KÖZÖS ZÓNA

Ez például egy olyan hálózat, ahol két cég összeolvadt és kialakítottak egy olyan alhálózatot, amelyikhez mindketten hozzáférnek. Ez a közös zóna. Ebben jelenleg egy webszerver fut, melyet mindkét hálózatból el lehet érni.
Mi vagyunk a kicsi zöld munkaállomással, a NagyCeg hálózatában. Az első esetben a velünk egy alhálózaton lévő file szervert próbáltuk elérni, a második esetben a közös zónában lévő webszervert, a harmadik esetben pedig kint valahol az index.hu-t. Amikor ki kellett mennünk az alhálózatunkból, akkor a route tábla alapján tudtuk eldönteni, hogy melyik routert célozzuk be.

Ez mind szép - most már csak azt kellene tisztáznunk, hogyan kerültek bele a bejegyzések a route táblába?

A legegyszerűbb dolgunk a legelső sorral (0.0.0.0) van. Amikor egy hálózati kártya TCP/IP beállításainál beírjuk a Default Gateway értékét, akkor gyakorlatilag ezt a sort adjuk meg. Mit is jelent a Default Gateway? Minden nem specifikált esetben erre kell továbbmenni.
A Default Gateway alóli kivételnél már egy kicsit bonyolultabb a módszer.
Használhatjuk például a route parancsot:

```
route add -p 192.168.99.0 mask 255.255.255.0 192.168.1.2 metric 1
```

Ez a parancs adja hozzá a fenti példában azt a plusz sort, mely a 192.168.99.0 irányra vonatkozik. Felhívom a figyelmet a -p kapcsolóra: ekkor a bejegyzés perzisztens lesz, azaz kikapcsolás után is megmarad.

A route táblát tudjuk piszkálni a netsh paranccsal is, de igazán nagy hálózatoknál már komolyabb módszereket szoktak alkalmazni. (Route/Switch Processor, vagy az Internet réteg ICMP szolgáltatásai.)

Viszont az ábrán az is látszik, hogy a hálózat nem ér véget a közös zónánál. Elméletileg a 3-as routeren keresztül be is tudnánk menni a KisCeg hálózatába - mely hálózatról egyelőre semmi információnk sincs. Tételezzük fel, hogy van náluk is egy file szerver (172.30.27.101) melyet el kellene érnünk. Honnan fogjuk tudni, hogy most melyik router felé kellene mennünk? Hát úgy, hogy beírjuk a route táblánkba, hogy ebben az esetben is a Router2 felé kell menni. Honnan tudja majd a Router2, hogy neki merre kell továbbmennie? Beírjuk az ő route táblájába is azt a bizonyos sort.
Ezt hívják úgy, hogy statikus routing. De mi van akkor, ha mi vagyunk egy óriási multi cég és többszáz routerünk van?
Gond egy szál se. A routerek beszélgetnek egymással. Ki tudják cserélni a routolási információikat. Ezt hívjuk dinamikus routingnak.

Ettől a rövid ismertetőtől nagyon messze áll, hogy belemenjek a routerek társalgását leíró protokollok ismertetésébe. Van néhány. Ezek között van olyan, mely az Internet rétegben működik, van amelyik az alkalmazás rétegben (RIP - lásd *2.1. ábra Rétegek a Microsoft TCP/IP megvalósításában*.) Van olyan, mely korlátozott számú ugrást tud kezelni, van, amelyik korlátlant. Van gyors, van lassú. Van kerek csokoládé, van lyukas csokoládé. És így tovább.
Ami nekünk fontos, az az, hogy a mai routerek már okosak, ma már nem kell 15 ujjal gépelnie egy network rendszergazdának ahhoz, hogy a routerekben lévő route táblák mindig összhangban legyenek a tényleges hálózattal. Így viszont maga az Internet réteg algoritmusai mindig ki tudják választani a megfelelő utat a forrás és a cél között.

Mindig?

Nem feltétlenül. A routerek ugyanis nem ész nélkül engednek át magukon forgalmakat. Természetesen szabályokkal tarthatjuk kordában, hogy melyik irányból, melyik irányba, kicsoda, mit csinálhat a micsodájával. Ez teljes mértékben a rendszergazdákon múlik.

2.3.2 CÍMFORDÍTÁSOK (NAT, PAT)

Vannak például speciális tartományok, melyeket bizonyos routereken kötelezően tiltani kell.

RFC 1918

2.2. TÁBLÁZAT

Kezdő cím	Végső cím	Címek száma
10.0.0.0	10.255.255.255	16777216
172.16.0.0	172.31.255.255	1048576
192.168.0.0	192.168.255.255	65536

Illetve később bővült a klub.

RFC 3927

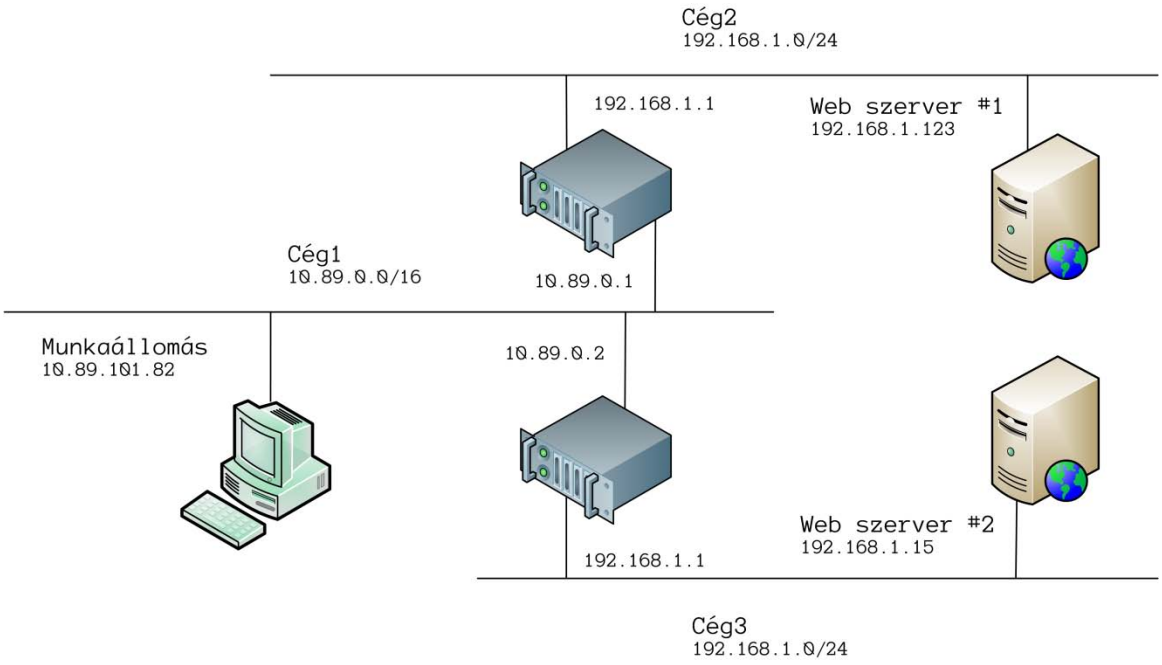
2.3. TÁBLÁZAT

Kezdő cím	Végső cím	Címek száma
169.254.0.0	169.254.255.255	65536

Az első táblázat tartományait nevezzük privát tartományoknak, az alsó táblázat tartományát pedig IPv4 link-local tartománynak. (Leánykori neve APIPA, Automatic Private Internet Protocol Addressing.)

Mi értelme lehet egyáltalán egy tartomány tiltásának a routeren?
Hát például az, hogy két helyen is létezik.

Persze jogos lehet a kérdés, miért akarunk két vagy több ugyanolyan címtartományú hálózatot működtetni? Nos, azért mert a világszerte egyedi címekből kevés van és egyébként is drágák. Ezért találták ki a privát tartományokat: ezekből mindenki annyit használ a saját szemétdombján belül, amennyit akar - de az internet határait őrző routerek ezeket a tartományok soha nem fogják ráengedni az internetre. (Pontosabban már a jól konfigurált céges edge routerek sem engedik el az internetes edge routerekig a privát tartományokat.)



2.9. ÁBRA GUBANC AZ ALHÁLÓZATOK KÖZÖTT

Csak hogy érthetőbb legyen, miért nem lehet kapcsolatban két ugyanolyan tartomány. Szokás szerint megint a zöld munkaállomásról nyomulunk. Melyik router IP címét is kellene beírnunk a route táblánkba, hogy elérjük a 192.168.1.123-as IP című Webszerver #1 gépet? És mi van, ha közben a Cég3 hálózatába is felvesznek egy 192.168.1.123 IP című gépet, mondjuk egy file szervert?
Ilyen esetekben csak a tiltás jöhet szóba[3]. Vagy a Cég2, vagy a Cég3 - vagy mindkettő - hálózatát letiltjuk a routereken.
Igenám, de a letiltás után hogyan fognak ezek a szerencsétlenek dolgozni? Hogyan fogom elérni a webszervereket a kicsi zöld munkaállomásomról?
Ha a teljes céges hálózatom privát tartományban van, hogyan fognak az alkalmazottak böngészni a neten?

RFC 1631

Erre találták ki a címfordítási technikákat.

Ilyenkor a router nem közvetlenül engedi át magán a forgalmat, hanem lop, csal, hazudik. Emlékszünk (*2.7. ábra Routolási folyamat*), sima routolás esetén a router csak a MAC címekkel játszik. Címfordításnál már bevonja a játékba az IP címeket (Internet réteg, Network Address Translation, azaz NAT), illetve a portokat (Transport réteg, Port Address Translation, azaz PAT)

[3] Illetve természetesen az IP tartományok megváltoztatása.

Egy kicsit megint előreszaladunk. A port fogalma a Transport rétegben jelenik meg de igazából az alkalmazás rétegnél lesz bővebben kifejtve.

Pár szót azért itt is ejthetünk róla.

Tételezzük fel, hogy van a cégnél egy többfunkciós szerverünk: fut rajta egy webes alkalmazás, emellett DNS szerver is, no meg fájlszerverként is funkcionál. Én a kis zöld munkaállomásomról böngészem a szerver weblapjait, miközben le akarok kapni egy Excel táblázatot, amely a szerver egyik megosztásában van.
Nem akadnak itt össze a szálak? Az én IP címem, MAC címem fix, hasonlóan a szerveré is. Honnan fogják tudni a közlekedő csomagok, hogy melyik melyik alkalmazáshoz fog tartozni?

Erre találták ki a portszámot.

Azaz amikor például egyszerre böngészek és másolom a táblázatot, így alakulnak a viszonyaim:

2.4. TÁBLÁZAT

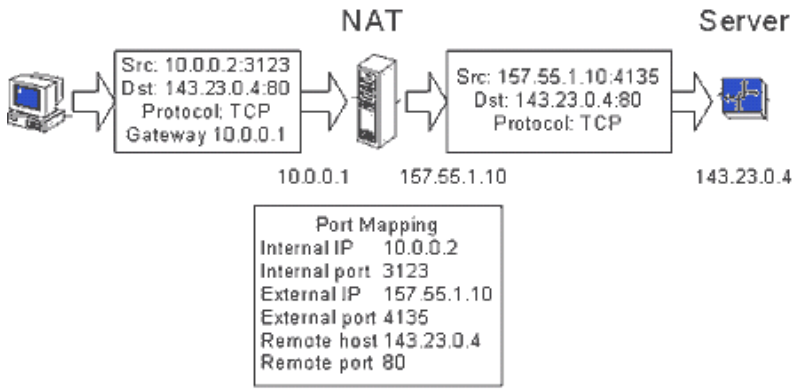
	Weblap böngészés	Exceltábla letöltése
Source IP	10.89.101.82	10.89.101.82
Source Port	45789	33478
Target IP	192.168.1.123	192.168.1.123
Target Port	80	445

Oké, vissza a címfordításhoz. Azt mondtam, a router lop, csal, hazudik. Úgy bizony. Címfordítás esetén úgy viselkedik, mintha ő, a megfelelő alhálózatba belógó lábával indította volna el a kérést - azaz kicseréli a forrás IP címet a saját, megfelelő IP címére. Persze a router nem fog tudni sokat kezdeni mondjuk egy netről letöltött katewinslet.jpg fájllal, emlékeznie kell, ki is kérte azt eredetileg és továbbítania neki.

Amikor a címfordítás úgy történik, hogy csak az IP címet cseréli ki a router, akkor beszélünk NAT-ról. Bár kicsi az esélye, de előfordulhat, hogy két node fordul ugyanazon webszerverhez, más IP címről, de ugyanazt a kliensportot használva - ilyenkor a router azért zavarba hozható.
Emiatt inkább elterjedtebb az a címfordítás, amikor a router nemcsak a feladó IP címét, hanem a portszámát is kicseréli. Ekkor beszélünk PAT-ról.

> Illetve megkülönböztetünk olyan verziót, amikor a router csak a source adatokat piszkálja (SNAT) és van olyan, amikor a router a cél adatokat babrálja (DNAT). De a számítástechnikai köznyelv ezt az egész bagázst (NAT, PAT, SNAT, DNAT) nevezi egész egyszerűen csak NAT-nak.

A portfordítós NAT-ot érdemes részletesebben is megnézni.



2.10. ÁBRA NAT, PORTFORDÍTÁSSAL

A 10.0.0.2 forrás IP címről (port: 3123, tipikus kliens port) egy webszerverhez szeretnénk kapcsolódni: 143.23.0.4, a portszám: 80 (tipikus webszerver port). Csakhogy közben van egy natoló router, mely eltárolja a feladó adatait egy port mapping táblázatban, majd kicseréli a feladó adatait a csomagban: beírja a saját külső lábának IP címét és egy másik kliens portot. Mindezt szintén rögzíti a port mapping táblázatban, sőt, ide kerülnek a címzett adatai is. A módosított feladótól elmegy a kérés a címzetthez, az vissza is válaszol neki. Itt kap szerepet a port mapping táblázat, a router abból keresi ki, hogy ki is volt az eredeti feladó, felülírja a csomagban a címzett címét és továbbítja visszafelé a csomagot.

> Aki szeretne jobban is elmélyülni a különböző címfordításokba:
> http://en.wikipedia.org/wiki/Network_address_translation

Amit még érdemes megjegyezni, hogy a NAT-nak erősen megvannak a korlátai. Nem egy olyan protokoll létezik, ahol nem csak az IP datagram fejléce tartalmazza az IP címeket, hanem maga az alkalmazás is belerakja egyes adatcsomagjaiba ezeket.(Pl. FTP protokoll, PORT v. PASV parancsok.) Ilyenkor a natoló eszköz vagy fel van készítve arra, hogy itt is cserélni kell az IP címeket, vagy elhasal az alkalmazás.

2.4 Szállítási (Transport) réteg

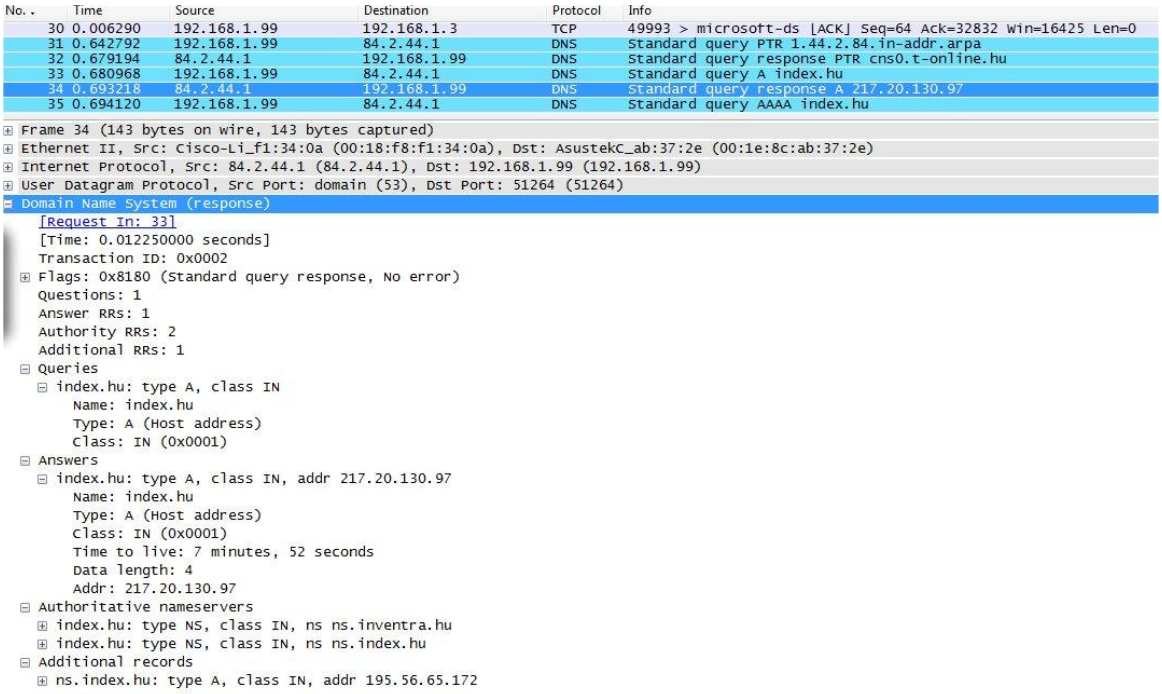
Dacára a nevének, nem ez a réteg mozgatja a csomagokat. Az a Network Interface réteg. A Transport réteg *szervezi* a szállítást.

Méghozzá kétféle szervezőelv szerint képes dolgozni.

2.4.1 User Datagram Protocol (UDP)

RFC 768

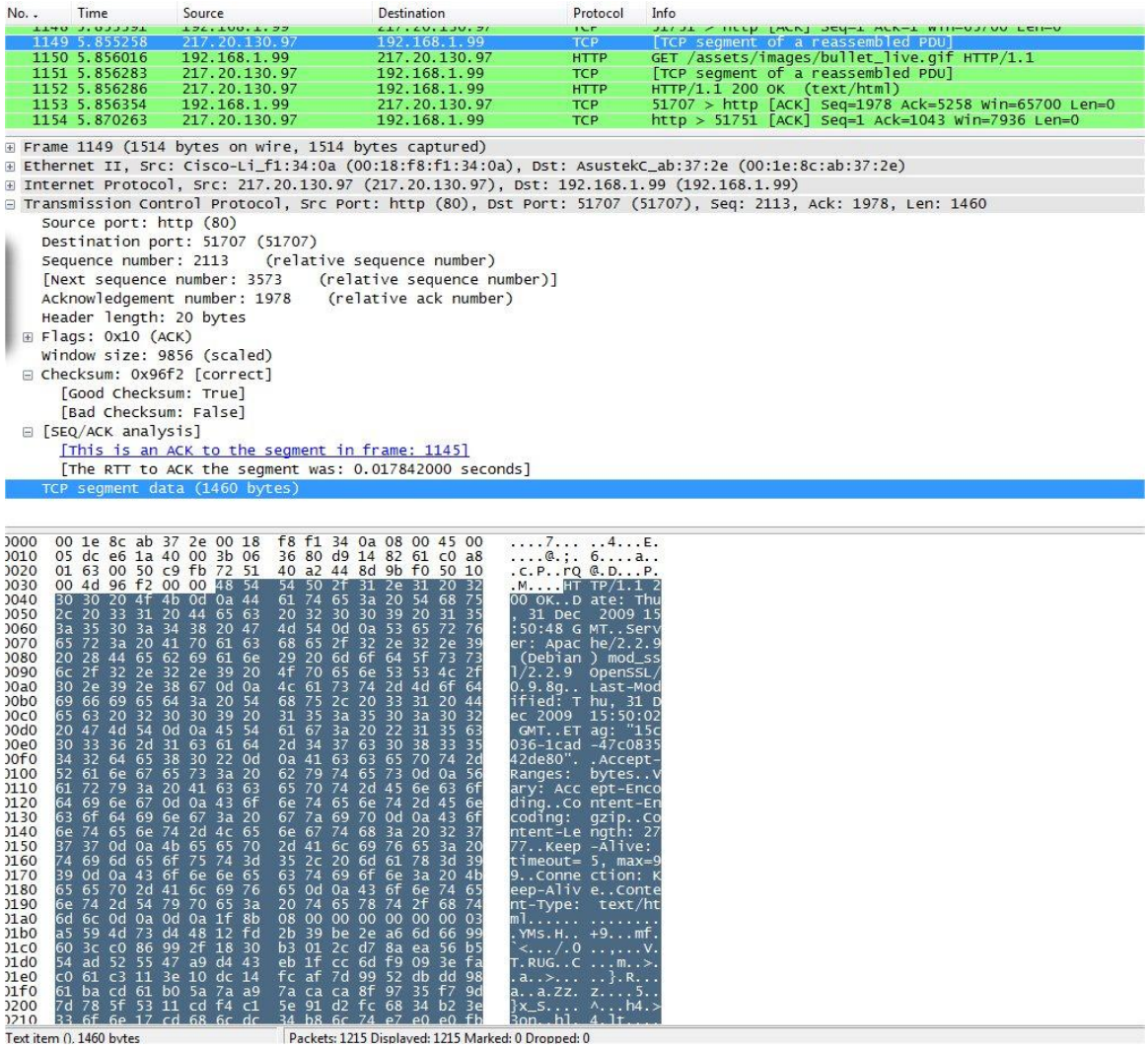
Mint a nevéből is látszik, datagram. Azaz olyan csomag, mely kompakt. Ha kinyitom a payload-ot, akkor önmagában is értelmezhető blokkot tudok kivenni belőle.



2.11. ábra Egy UDP csomag

Tessék megnézni. Megkérdeztem az index.hu IP címét - és visszakaptam, hogy az 217.20.130.97. Emellett megtudtam azt is, hogy kik az index.hu name szerverei. Az információ kompakt, sehol sem ír olyat a wireshark, hogy a további részletekért menj tovább az xy sorszámú csomaghoz.

2.4.2 TRANSMISSION CONTROL PROTOCOL (TCP)



2.12. ÁBRA TCP CSOMAG

Szemmel láthatóan kompaktságról szó sincs. A csomag payload-jában egy adatszegmens van (1460 bájt), a tartalmát csak alul a bináris részben tudjuk megnézni. Ránézésre meg tudjuk tippelni, hogy ez egy HTTP üzenet fejléce lesz, de a végét nem látjuk. Nem is láthatjuk, hiszen csak egy 1460 bájtos darab utazik a csomagban.

Nyilván nem árulok el titkot, hogy a TCP szervezési mód a bonyolultabb. Egész pontosan sokkal-sokkal bonyolultabb. Ha elveszik egy UDP csomag, attól még senki nem dől a kardjába. Ha elveszik egy TCP szegmens, akkor az egész adatfolyamnak lőttek - és ez már tud fájni egy 100 megás zip fájlnál. De a TCP nemcsak megbízhatóságban jobb, egy csomó extra lett még beleépítve.

2.5. TÁBLÁZAT

	UDP	TCP
Kapcsolat	**Kapcsolatmentes.** A feladó elküldi a csomagot, a címzett megkapja. Ezen a tranzakción kívül semmit nem beszélgetnek.	**Kapcsolatcentrikus.** Még a tényleges adatküldés előtt a felek kiépítenek egy kapcsolatot és azt ápolják is.
Megbízhatóság	**Megbízhatatlan.** A küldő/fogadó alkalmazásokra bízza, hogy vegyék észre a csomagvesztést.	**Megbízható.** Az adatfolyam darabjai számozottak és azokat leltár szerint át kell vennie a fogadó félnek.
Puffer	**Nincs.** A küldő alkalmazás egyből küldi a csomagot és az ahogy megérkezik, egyből megy is tovább a fogadó alkalmazáshoz..	**Van.** Mind küldő oldalon, mind fogadó oldalon létezik egy átmeneti tárolóhely, mely segít a csomagok elrendezésében.
Tördelés	**Nincs.** Az UDP külön nem foglalkozik vele. Ha a csomag nagyobb lett az MTU-nál, akkor majd tördel az IP réteg. A maximális mérete 64 KB.	**Van.** A TCP képes kommunikálni az IP réteggel és a csomagok összerakásánál figyelembeveszi az MTU értékét.
Folyamatszabályozás	**Nincs.** Nem tudja érzékelni, hogy dugó van és lassítania kellene. Hasonlóképpen azt sem, hogy szabad a pálya, lehet gyorsítani.	**Van.** Rendszeresen méri a csomagok beérkezési idejét és ehhez hangolja a sebességet.
Többnejűség	**Poligám** Képes a pont-multipont kapcsolatra. Multicast kommunikációnál csak UDP jöhet szóba.	**Monogám** Csak pont-pont kapcsolatra jó, multicasthoz használhatatlan.
Sebesség	**Gyors** Mivel nincs benne ez a sokfajta védelem, meg ellenőrzés.	**Lassú.** Mivel ebben viszont benne van az a sokfajta védelem, meg ellenőrzés.
Hatékonyság	**Jó** A csomagok tartalmához képest nem túl nagy a vízfej.	**Gyenge** Mind a csomagok fejlécét, mind az extra elküldött csomagokat tekintve nagy a vizfej.

A táblázatból remekül kiolvashatóak a tipikus felhasználási területek is. Nem akarom a végtelenségig elnyújtani, így csak egy markáns példát emelnék ki.

A DNS névfeloldás felváltva használja az UDP, illetve TCP protokollokat, méghozzá úgy, hogy ha az információ belefér egy UDP csomagba, akkor úgy megy (tipikusan a DNS lekérdezés, illetve válasz), ha nem, akkor jön a TCP (tipikusan zónatranszferek).

2.5 ALKALMAZÁS (APPLICATION) RÉTEG

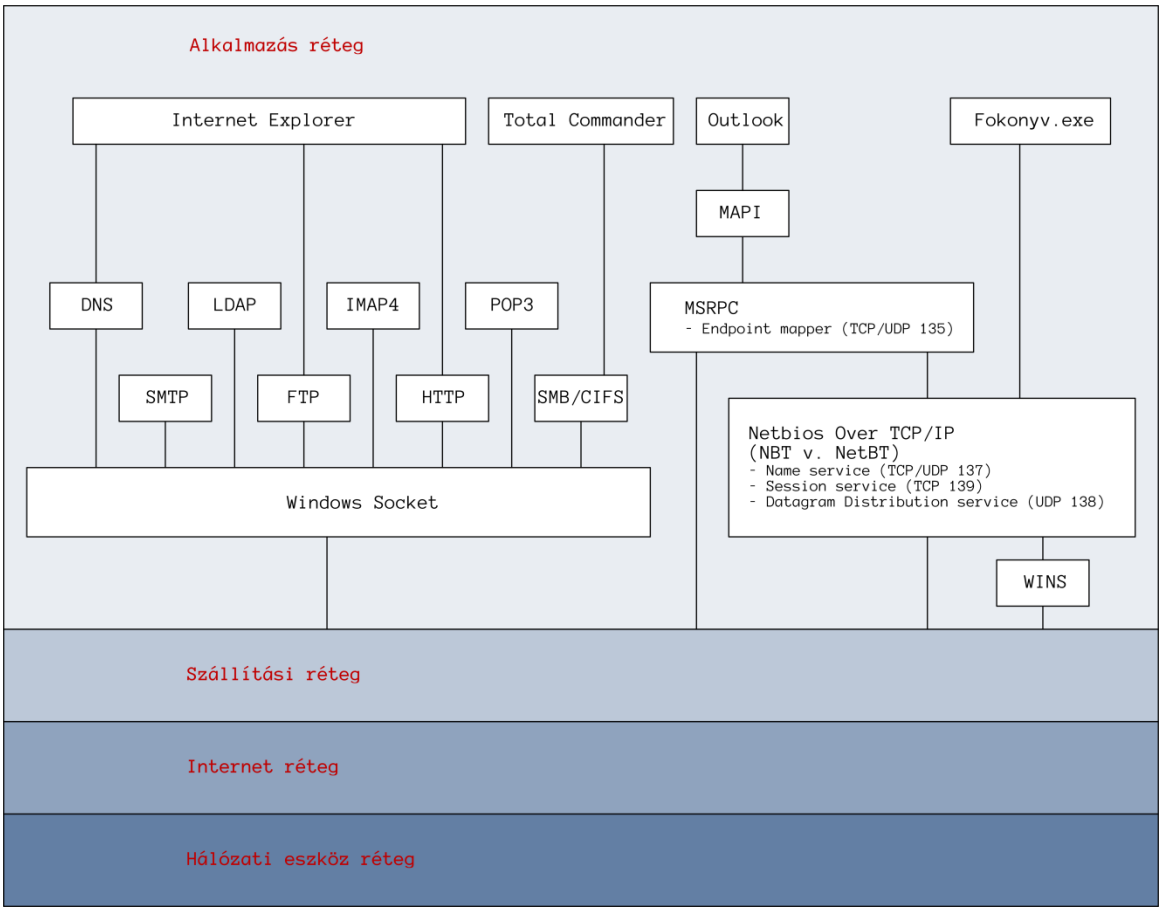
Megérkeztünk a dzsungelbe. Eddig ugyanis meglehetősen rögzített formák között zajlott az élet - innentől viszont elszabadul a pokol. Kismillió protokoll - és egyáltalán nem kötelező, hogy ezek szabványosítva legyenek. A Szomszéd Lacika kitalálja, hogy ír egy szerveralkalmazást, mely az 59999-es porton szolgáltat, majd ír hozzá kliens alkalmazásokat, melyek ezen a porton érik el a szervert. Ez már az alkalmazás réteg.

Tipikus vadnyugat. Még akkor is, ha Lacika már TCP-t vagy UDP-t fog használni, a címek kezelését az IP réteg fogja végezni, a szállítást pedig a hálózati réteg.

Természetesen a protokollok jó része szabványosított. Nincs semmi értelme újra meg újra felfedezni a spanyolviaszkot. (Pl. megírni egy alternatív DHCP-t.)
A szabványosított (RFC, ugye) protokollok viszont rögzített portokon kommunikálnak. Persze ne gondoljon senki kőbevésésre, minden további nélkül üzemeltethetek webszervert, mely nem a 80-as porton működik - csak éppen elveszítek mindenkit, aki olyan céges tűzfal mögül jön, ahol csak a 80-as portot engedélyezték böngészésre.

Ezeket a szabványos, de legalábbis erősen ajánlott portokat nevezik wellknown portoknak.

2.5.1 Térkép az alkalmazás réteghez



2.13. ábra Térkép az alkalmazás réteghez

Természetesen nem teljes. Irgalmatlanul nem teljes. Nyilván nem csak ennyi protokoll létezik. Nyilván nem csak a böngésző használja a DNS-t. A Total Commander nyilván nem csak az SMB-t használja - ha nevet kell feloldania, fordulhat DNS-hez és WINS-hez is. A fokonyv.exe-ről meg legtöbbször a fejlesztője sem tudja, mit is használ pontosan.

De az elv, ahogy egymásba épülnek a kockát, remélhetőleg jól látható.

Apropó, SMB. Nem, nem sajtóhiba: a Windows Server 2000 óta már nem a Netbios-on keresztül dolgozik, hanem közvetlenül fordul a TCP-hez (TCP 445). Innentől nevezik CIFS-nek is.

2.6 Tűzfalak

Ezek azok az eszközök, melyek képesek valamilyen szinten kontrollálni a rajtuk keresztülmenő forgalmat. A routereknél már említettem, hogy le lehet tiltani rajtuk bizonyos forgalmakat. Erre a célra jogosultsági listákat (ACLs) használunk és az egyszerűbb feladatokhoz ez bőven elég is.

De akadnak jócskán összetettebb feladatok is. Amikor egy jogosultsági lista már kevés. Ilyenkor hívjuk segítségül a tűzfalakat.

Egy tűzfal többféleképpen tudja kontrollálni a rajta átmenő forgalmat:

- csomagszűréssel, azon belül is
  - állapottér nélküli (stateless), vagy
  - állapotteres (stateful) csomagszűréssel, illetve
- proxyzással (application layer, protocol proxy).

Kihasználom, hogy a fejezet alapvetően bevezetés jellegű, így akár durván egyszerűsíthetek is.

A csomagszűrést képzeljük el úgy, hogy áll a biztonsági őr a repülőtéri kapunál, mindenkit megmotoz és megnézi mi van nála. Egy golyóstoll? Mehet. Egy kulccsomó? Mehet? Egy üveg kóla!? Riadó, terroristaveszély!
Nyilván ez úgy történik, hogy a biztonsági őrnek ki van adva, hogy mit engedhet fel. Mérlegelési joga nincsen. Nem fogja megvizsgálni, hogy az egy játékpisztoly, vagy igazi - nem engedi fel. Ez a stateless változat.

A sateful csomagszűrés már intelligensebb. Maradva az előző példánál, itt is jönnek sorban az emberek. A biztonsági őr itt is átvizsgál mindenkit. Golyóstoll? Felírtam, mehet. Kulccsomó? Felírtam, mehet. Egy üveg kóla? Felírtam, mehet. Beazonosíthatatlan vasdarab? Felírtam, mehet. 1 deci Moyra körömlakk? Lássuk csak, valaki már vitt fel egy üveg kólát, márpedig a kóla keverve ezzel a körömlakkal, robbanóanyagot alkot[4]. Kidobjuk, és visszahívjuk a kólás fazont, hogy ő is dobja ki a

[4] Nyilván nem... ez csak egy példa. Nehogy nekiállj ezekkel otthon napalmbombát barkácsolni.

kóláját - mert elképzelhető, hogy ezek ketten rosszban sántikáltak. (Vigyázat, nagyon durva egyszerűsítés.)

Látható, hogy a stateful változat sokkal barátságosabb és rugalmasabb. Cserébe viszont őrült nagy noteszt kell kezelnie a biztonsági őrnek, jól kell tudnia kombinálni, és naprakész technológiai adatbázis kell mögé, hogy tudja, milyen alkatrészekből milyen fenyegetéseket lehet összerakni.

Nézzük a proxy tűzfalakat.

A pilóta már bent ül a pilótafülkében, hamarosan indulnak - de ekkor megkíván egy hamburgert. Odaint egy biztonsági őrt, ad neki pénzt, az őr pedig kimegy a repülőtér elé. Tudja, hol van hamburgeres - mert a pilóta elmagyarázta neki - megveszi a hamburgert, alaposan leellenőrzi, hogy az tényleg csak egy ártalmatlan hamburger legyen, majd beviszi és odaadja a pilótának.
Azért hívják ezt a technikát proxynak, mert a küldő és a fogadó fél közvetlenül nem találkoznak. Egy helyettesítő személy - a proxy - közvetít közöttük. Az egyik elmagyarázza neki, mit szeretne, a proxy ekkor kimegy a vad, veszélyes terepre, begyűjti a begyűjtendőket, majd ellenőrzés után odaadja a megbízójának.

Látható, hogy a direkt kapcsolat hiánya miatt a proxy jóval biztonságosabb, de ennek a technológiának is megvannak a hátrányai. Mi van, ha a pilóta hirtelen egy talajgyalut szeretne? A biztonsági őr csak hamburgerre lett kiképezve. Csak egy hamburgerboltban tudja, hogyan kell viselkednie. Csak a hamburgert tudja átvizsgálni, hogy tényleg hamburger-e. Csak a hamburgerhez van kis rózsaszín műanyag doboza, amelyben biztonságosan tudja szállítani. Ha a pilótának talajgyalu kell, akkor egy másik biztonsági őrt kell odahívnia, aki otthon érzi magát egy exkavátor szaküzletben, aki meg tud győződni arról, hogy tényleg talajgyalut kapott, nem pedig afgán öngyilkos merénylőt, és akinek van olyan rózsaszín dobozkája, amelyben biztonságosan tudja szállítani a talajgyalut. Aztán a fókabébihez harmadik biztonsági őr kell. És így tovább. Ráadásul mindegyik őrt folyamatosan képezni kell, mert a fejlődés soha nem áll meg, a hamburger, a talajgyalu, a fókabébi állandóan változik, átalakul. Arról nem is beszélve, hogy a fizetőeszköz is, és a pilóta lehet, hogy egy idő után bankkártyát ad - a biztonsági őrnek meg kell tanulnia azt is használni. Végezetül remélem az is látszódik, hogy a proxy technika jóval lassabb is.

Az SPF (Stateful Packet Filter) technológiának markáns képviselői a Checkpoint tűzfalak, míg a proxy technológiára hirtelen a magyar Zorp Gateway tűzfalat tudnám megemlíteni.

> Checkpoint:
> http://www.checkpoint.com/
> Zorp Gateway:
> http://www.balabit.hu/network-security/zorp-gateway/

A Microsoft TMG (és az ISA vonulat is) hibrid termék. A webes protokollokat proxy technikával kezeli (webproxy service), a többi hozzáférést pedig SPF módon.

3 TCP/IP v6

RFC 1719

Ha hiszed, ha nem, az RFC 1719 már 1993-ban elkészült, az IPv6 - mely gyakorlatilag az IP réteg teljesen más szemléletű kezelését megvalósító szabványgyűjtemény - pedig 1995-ben állt össze. 15 év!
Ehhez képest nem mondanám, hogy olyan nagyon közismert, nagyon elterjedt dologról lenne szó.

Történelmileg úgy alakult, hogy a kilencvenes évekre kezdtek elfogyni az IP címek. Az IETF-nél gyorsan nekiálltak kidolgozni egy újabb szabványt, közben workaroundként bedobták a CIDR szemléletet és a NAT technológiákat. Gondolom, nem először hallasz ilyesmiről, de a workaround annyira jól sikerült, hogy később, amikor már elkészült a végleges megoldás, senki sem érezte szükségét lecserélni. (Ne mondd, hogy soha nem hallottál még kávéfőző-gumival hajtott vízpumpáról.)
Nagyjából mostanra értünk el odáig, hogy az ideiglenes megoldás kezd elérkezni lehetőségei határaihoz. Azért akárhogy is nézzük, a NAT meglehetősen erőforrásigényes és az IPv4-nek nem is egy komoly hátránya van az IPv6-hoz képest.

Óriási változásról van szó. Mármint mennyiségben. Az internet az utóbbi 15 évben átlépett minden elképzelhető határt. És ebben a hatalmas hálózatban kellene komplett lecserélni az IP réteget egy másikra.
Nyilván nem fog menni egyik napról a másikra. Először el kell jutni addig, hogy minden résztvevő gyártó alapból is támogassa. Aztán le kell cserélni azokat az eszközöket, amelyek nem támogatják. Végül szép finoman, óvatosan el kell kezdeni átállítgatni a rendszereket. Természetesen a mérnököknek, a rendszergazdáknak addigra képbe kell kerülniük, meg kell tanulniuk az új IP réteget - hiszen hosszú évekig fognak egymás mellett üzemelni IPv4 és IPv6 hálózatok.

Izgalmas időszak lesz - és van egy tippem, hogy a minesweeper-t be fogja előzni a népszerűségi listán a calc.exe.

Kezdjük a leglátványosabb változással: jóval nagyobbak lesznek a címek. Eddig 32 bites címeket használtunk (ezeket írtuk le négy bájtos formában) - mostantól 128 bites címeink lesznek. Anélkül, hogy elmerülnénk a matek rejtelmeiben, ez nem négyszeres változás - sokkal inkább negyedik hatványos. Ennyi cím valószínűleg akkor is elég lesz, ha Kínában az összes kenyérpirítónak saját egyedi IP címet kell adni.

Nézzünk egy példát:

```
11000000 10101000 00010111 00001100
```

Ez a 192.168.23.12, binárisan ábrázolva.

Csak a hecc kedvéért írjunk fel egy IPv6 címet binárisan:

```
11000000 10101000 00010111 00001100 11000000 10101000 00010111 00001100
11000000 10101000 00010111 00001100 11000000 10101000 00010111 00001100
```

Ez a 192.168.23.12.192.168.23.12.192.168.23.12.192.168.23.12 cím.
Ilyen címekkel a fejünkben fogunk szaladozni.

Nyilván nem. Kénytelenek leszünk egy átláthatóbb ábrázolást választani. Persze ezzel is lesz gondunk, hiszen úgy megszoktuk már a régi, tizes számrendszeren alapuló ábrázolást - most meg lehet majd hexázni.

```
C0A8:170C:C0A8:170C:C0A8:170C:C0A8:170C
```

A fenti IP címet ilyen formában fogjuk leírni, megtanulni, használni.

3.1. TÁBLÁZAT

Hexadecimális	Decimális
C0	192
A8	168
17	23
0C	12

Némi rövidítés még lehetséges.

- Például - az IPv4 címekhez hasonlóan - nem kell kiírni a vezető nullákat. Azaz a ":0FB9:" címdarab teljesen egyenértékű az ":FB9" címdarabbal.
- A teljesen nulla darabokat össze is lehet vonni.
  A C0A8:170C:C0A8:0000:0000:170C:C0A8:170C cím helyett írhatjuk azt is, hogy C0A8:170C:C0A8::170C:C0A8:170C. Nyilván ilyen összevonást csak egy helyen tehetünk egy címben, hiszen egyébként nem tudnánk, az adott helyen hány nulla volt.

A subnet mask szerepe megmaradt, de már nem irkáljuk azt a rengeteg F betűt meg nullát. Maradunk a /xxx formátumnál.
Azaz a C0A8:170C: C0A8:170C: C0A8:170C: C0A8:170C/64 cím azt jelenti, hogy az első 64 bit a hálózatazonosító, a második 64 pedig az alhálózaton belül a node azonosítója.

A végén nézzünk egy elrettentő példát. Hogyan néz ki a localhost cím az IPv6-ban?
::1/128. Így. Kibontva: 0000:0000:0000:0000:0000:0000:0000:0001/128.

Nagy változás, hogy az IPv6-ban az alapvetően kétsíkú IPv4 hierarchia széthúzódik. Egy IPv6 címről első látásra meg fogjuk tudni állapítani, hogy melyik zónába esik:

- Világon egyedi
- Site szintű
- Lokális

Eddig annyit tudtunk, hogy egy IPv4 privát címről meg tudtuk mondani, hogy az lokális cím. A publikus címre azt mondtuk, hogy az valószínűleg nem lokális cím. Az IPv6-nál eleve bejött egy közbenső szint, a site. Elég hosszú a cím, ki tudunk alakítani háromszintes struktúrát is. (Nem kötelező. Lehet. De az ISP-k valószínűleg használni fogják.)

Fussuk át, milyen IPv6 címek lehetségesek.

UNIQUE-GLOBAL: Világszerte egyedi cím. Az első 3 bit jelzi, hogy ez unique-global cím (001), a következő 45 bit egy globális azonosító (pl. az ISP-m világon egyedi azonosítója), majd jön 16 bit site azonosítási célra (például az ISP-m csinál egy külön Budapest site-ot), végül a maradék 64 bit lesz a node azonosítója. A node azonosítóját generálhatjuk a MAC címéből (IEEE EUI64), de lehet véletlenszám is.
Ez gyakorlatilag azt jelenti, hogy a világszerte egyedi címhez a rendszergazdáknak hozzá sem kell nyúlni, generálódik magától.

LINK-LOCAL: Olyan cím, mely csak lokálisan értelmezett. A routerek nem fogják kiengedni az internetre. (Meglehetősen analóg az IPv4 privát címeivel.) Az első 10 bit jelzi, hogy ez link-local cím (1111111010), utána jön 54 darab nulla, majd a korábban is említett node azonosító. Azaz FE80::/64 konstans prefix, a végén pedig a node azonosító. Ez a cím is legenerálódik magától.

SITE-LOCAL: Nem vagyunk kötelesek elfogadni azt a site struktúrát, melyet a unique-global cím biztosít számunkra. Mi magunk is ki tudunk alakítani a magunk számára egy site struktúrát. Természetesen az ebben a struktúrában használt IP cím nem lesz világszerte egyedi - igazából nem is kerülhet ki a site-on kívülre. Az első 10 bit jelzi, hogy ez site-local cím (1111111011), a következő 54 bit lesz a site azonosítója, végül jön a jól ismert node azonosító. Ha a site IPv6 routereit felokosítjuk, akkor ez a cím is magától generálódik.

RFC 4193

UNIQUE-LOCAL: Olyan, világszerte nagyon nagy valószínűséggel egyedi címekről van szó, melyek lokálisan (maximum site szinten) értelmezettek, ezen a határon nem léphetnek át. Látszólag faramuci dolog ez. Akkor használják, ha cégek összeolvadásáról van szó, vagy össze-vissza kóborló mobil felhasználókról. Az első hét bit jelzi a cím kategóriáját (1111110), a következő bit egy flag (L). Ezt követi egy

40 bites azonosító, a Global ID. Ettől lesz a cím nagy valószínűséggel világszerte egyedi. Ez egy pszeudorandom szám, a generálása az RFC4193-ban le van írva, de a net is tele van olyan oldalakkal, ahol ugyanezzel az algoritmussal Global ID-t generálhatunk magunknak. Ha az L flag magas, akkor ezt az algoritmust használtuk a címadáshoz. Jelenleg ugyan más algoritmus nem létezik, de ha lesz, akkor az L flag alacsony állapota fogja jelezni, hogy az újabbat használtuk. Ez után jön 16 bit site azonosító, majd a jól ismert 64 bites node azonosító. Mondanom sem kell, ha a routernek megadjuk a szükséges ID értékeket, akkor ez a cím is automatikusan osztható ki.

Eddig tartottak az unicast címek. Léteznek természetesen multicast címek is. Ezek 8 darab magas bittel kezdődnek (11111111), ezt követi négy flag (RFC2373, RFC3306, RFC3956), majd jön négy bit, melyek a cím szkópját adják meg, végül jön egy 112 bites csoportazonosító. A multicast címek tipikusan FF0 kezdetűek, a flag-ek ugyanis általában nulla értékűek.

Néhány példa.

3.2. TÁBLÁZAT

Multicast cím	Fix érték	Scope	Group ID	A cím értelme
FF02::1	FF0	2	::1	link-local scope all nodes
FF02::2	FF0	2	::2	link-local scope all routers
FF05::2	FF0	5	::2	site-local scope all routers

A teljes választék:
http://www.iana.org/assignments/ipv6-multicast-addresses/

Korábban elejtettem egy olyan megjegyzést, hogy egy címről ránézésre is meg fogjuk tudni mondani, melyik kategóriába esik. Nos, a lehetőség megvan rá… de ehhez matekozni kell egy kicsit. Szerencsére a táblázat kitöltéséhez minden információ adott, csak vissza kell lapozni a címek leírásához.

3.3. TÁBLÁZAT

Címtipus	Címtartomány eleje	Címtartomány vége
Unique global	2000::	3FFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF
Link local	FE80::	FEBF::FFFF:FFFF:FFFF:FFFF
Site local	FEC0::	FEFF::FFFF:FFFF:FFFF:FFFF
Unique local #1[5]	FD00::	FDFF::FFFF:FFFF:FFFF:FFFF
Unique local #2[6]	FC00::	FCFF::FFFF:FFFF:FFFF:FFFF
Multicast[7]	FF01::0001	FF35::FFFF:FFFF

[5] Pszeudorandom algoritmussal.
[6] Jövőbeni algoritmussal - ergo ez a tartomány még nem él.
[7] Nem folytonos tartomány, még a definiált résztartományok közül sem aktív mindegyik.

Hogy mi van akkor, ha olyan címet látunk, melyet nem fed le a táblázat? Jelenlegi tudásommal, tapasztalatommal azt tudom csak javasolni, hogy óvatosan nyúlj fel a levegőbe és kezd el tanácstalanul vakarni a fejed tetejét.

Komolyabbra fordítva a szót, a fentiek alap címtipusok. Vannak bőven más címek is. Itt van például a solicited node cím. Az IPv6-ban ugyanis nem csak a címzés jelenti a nagy változást, hanem a különböző automatizmusok bevezetése is. Ilyen automatizmus a Neighbor Discovery (ND), azaz az a folyamat, melynek során az azonos linken lévő node-ok - mindenféle ARP varázslat nélkül - meg tudják találni egymást. Ennek a folyamatnak egyik eleme a solicited node cím. De ebbe most nem fogok belemenni, hasonlóképpen a Multicast Listener Discovery (MLD) folyamatba sem.

RFC 4862

Az autokonfigurációról viszont illik pár szót szólni. Eddig ugyanis elejtettem olyan megjegyzéseket, hogy majd a router legenerálja a címet.

Őszintén, nálatok ez a megszokott? Ha van dinamikus címkiosztás, akkor ott egy DHCP szerver teszi a dolgát, az osztja a címeket. Ha nincs, és nem statikus IP címeket használtok, akkor még működhet a korábban említett APIPA módszer.
De a router...? Micsoda perverzitás már ez?

Pedig nem az, csak szokatlan. Gondolj bele, melyik hálózati elem az, amelyik mindent tud magáról az alhálózatról? Természetesen a router. Miért ne osztogathatna címeket?
Ennek ellenére nem osztogat. Csak megmondja a hozzáforduló node-oknak a globális azonosítókat vagy a site azonosítót - azaz a prefixeket. A címeket ezek alapján már maga a node generálja le magának. (Valójában ennél sokkal többet mond, de most nem megyünk bele mélyebben.)

Jó kérdés, hogy ezek után mi szerepe lehet a DHCPv6-nak? Mert létezik.

A két dinamikus címosztás párhuzamosan működhet egymás mellett. A DHCP ugyanis tud olyan beállításokat is terjeszteni, amelyeket a router nem. Gondolj csak bele a DHCP Options lehetőségekbe (Domain name, WINS server name, WINS node type, lease time... meg ilyenek.)

http://www.iana.org/assignments/bootp-dhcp-parameters/

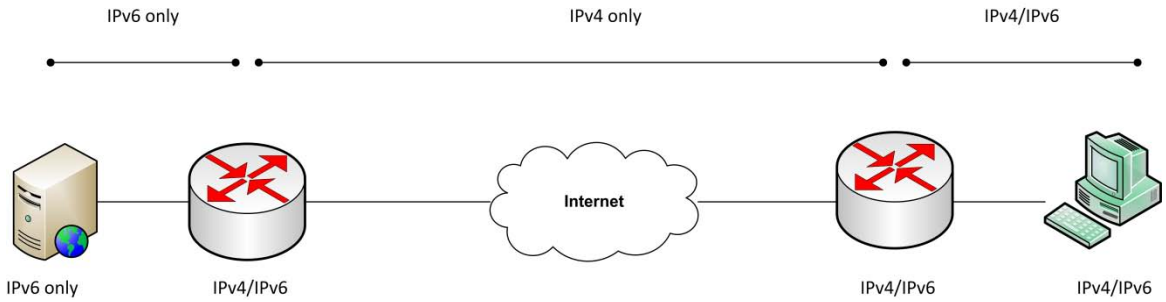
A következő router/DHCP kombinációk léteznek:

- Nincs DHCP, mindent a router oszt.
- DHCPv6 Stateless: Van router és van DHCPv6 is. A prefixeket a routerek osztják, minden más infót pedig a DHCPv6 szerver.
- DHCPv6 Stateful: Minden infót, azaz az IP címeket is, a DHCPv6 szerver osztja.

4 IPv4-IPv6 EGYÜTTMŰKÖDÉS

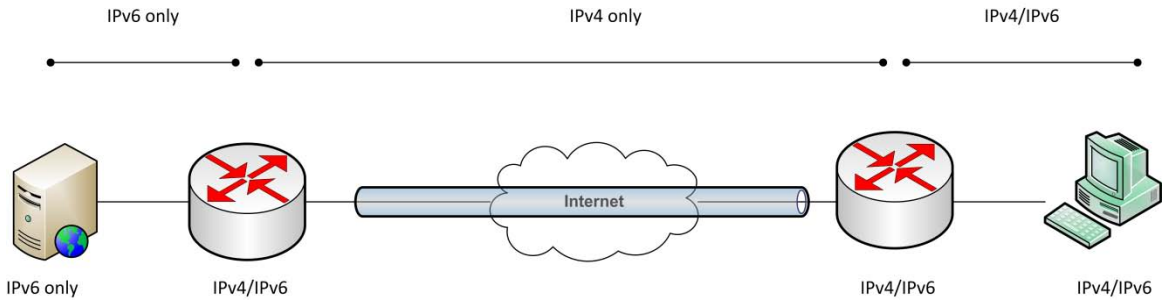
Ez nem csak azért bonyolult, mert önmagában is bonyolult technológiákat jelent - hanem azért is, mert ezeket a technológiákat a hatalmas és roppant változatos internet teljes egészén kell tudni majd alkalmazni.

Nézzünk egy példát.



4.1. ÁBRA IPV4 ÉS IPV6

Mit saccolsz, ez a felállás az internet jelenlegi állapotában (2010.01.01) működhet-e? El fogom tudni érni a kis zöld munkaállomásomról az IPv6 only webszervert? Gondolom, te is kiszúrtad, hogy a szűk keresztmetszet maga az internet. Hiszen a routerek le tudják kezelni mindkét IP verziót. De a neten csak az IPv4 megy át. Egész egyszerűen jelenleg ugyanis nem tudjuk garantálni, hogy nem kóborol az IPv6 csomag olyan szakaszra, ahol csak IPv4 only router van.

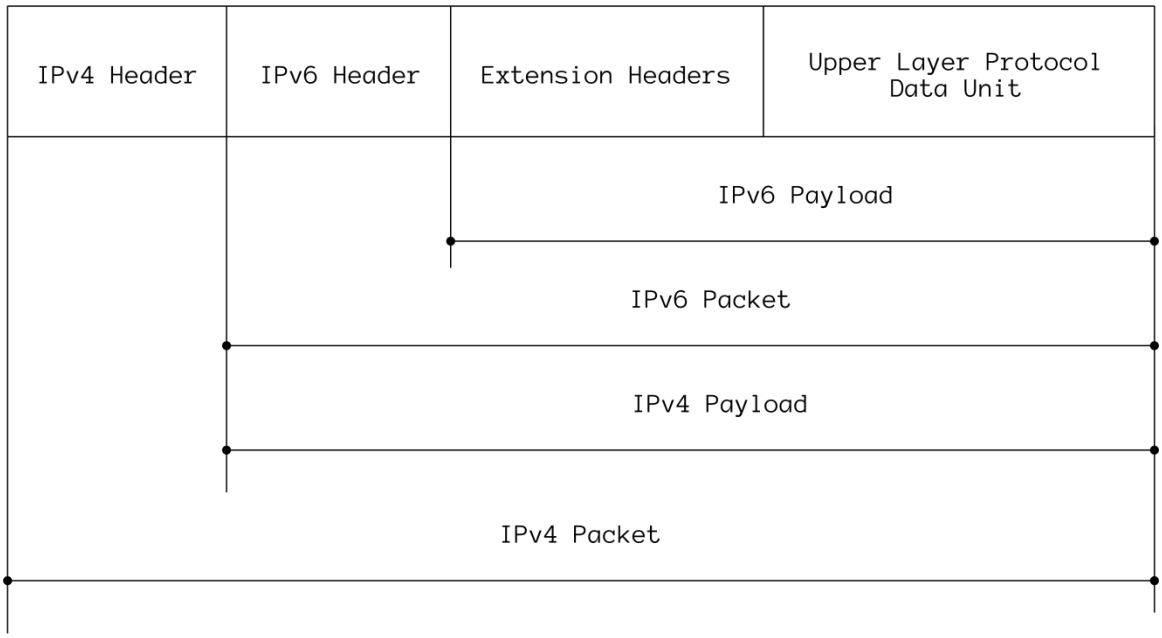


4.2. ÁBRA IPV4 - IPV6 TUNNEL

De ezt tudjuk kezelni. A két biztos pont között kiépítünk egy alagutat (csatornát, nevermind) - és ekkor tökmindegy, mi van útközben.

Hogyan is kell ezt elképzelni? Előveszek fizikailag egy vakondot. felhúzom kulccsal, beállítom a megfelelő irányba - és már fúrja is az utat az interneten keresztül?

Nem igazán. Sokkal inkább arról van szó, hogy a megszokott csomagszerkezetbe (*2.3. ábra Csomagok hátán csomagok*) valamelyik jól meghatározott ponton berakunk egy plusz csomagolást. Amit így becsomagoltunk, azt megóvtuk attól, hogy útközben bárki piszkálja. Ez a plusz csomagolás lehet autentikációs fejléc, lehet titkosítás (később látunk mindegyikre példát) - és lehet egyszerűen inkompatibilis információkat elrejtő csomagolás.



4.3. ÁBRA IPV6 CSOMAG AZ IPV4 CSOMAGON BELÜL

Ez látszólag egy sima IPv4 csomag. Van neki fejléce és van neki tartalma. Aztán hogy a tartalmon belül egy IPv6 csomag van becsomagolva, azzal egy IPv4 only router nem fog foglalkozni. (Azért jelezzük neki - Protocol 41.) Az alagút két oldalán lévő IPv4/IPv6 routerek viszont érteni fogják a helyzetet és képesek lesznek kicsomagolni az IPv6 csomagot az IPv4 csomagból. (Nekik muszáj lesz, mert a TCP/UDP csomag az IPv6 payloadban utazik, a jelen ábrán éppen az Upper Layer Protocol Data Unit név mögé bújtatva.)

Jó hírem van: ezekkel az alagutakkal túl sokat nem kell foglalkoznunk. Akár még a saját kicsi zöld kliensgépünk is képes arra, hogy teljesen magától, elengedett kézzel, automatikusan kiépítsen egy, az adott szituációhoz éppen illeszkedő alagutat.

Természetesen lehetőségünk van manuálisan is alagutat építeni. Netsh. Miért kérded?
Ebben az esetben informatikus szakik hoznak létre routerek között egy csatornát, mindkét oldalon beállítják a kapcsolódó alhálózaton a megfelelő route szabályokat - és már működik is a két szélső IPv6 hálózat közötti kommunikáció, az IPv4 hálózaton áthúzódó csatornán keresztül.

Az automatikus csatornaépítésre inkább az jellemző, hogy kliens és kliens/szervergépek között jönnek létre, ad-hoc jelleggel.
Amit mindenképpen tudnunk kell, az a két node IPv4 címe, illetve két IPv6 cím is, mely az IPv6 felől azonosítja be a csatornát. Jó lesz erre a gép saját IPv6 címe?
Visszakérdezek: melyikre gondoltál? Egy IPv6 node-nak lehet egy csomó IPv6 címe - miközben persze egyáltalán nem garantált, hogy ezek mindegyike létezik.
A megoldás az, hogy a meglehetősen egyértelmű IPv4 címből generálunk különböző, egyértelmű algoritmusok alapján IPv6 címeket és ezeket használjuk a különböző csatornatípusokhoz.

INTRA-SITE AUTOMATIC TUNNEL ADDRESSING PROTOCOL, ISATAP

RFC 4214

Az IPv4 címünből link-local IPv6 címet gyárt - ebből következően csak site-on belül használható.

6TO4

RFC 3056

Habár itt már elméletileg global-unique cím keletkezik, de mivel a képzéshez felhasználja a host IPv4 címét is, gyakorlatilag csak akkor jön létre, ha az IPv4 cím publikus tartományba esik. Más szóval, ha a gépemnek privát IP címe van, én pedig natolás után érem el az internetet, akkor ez csatornatípus nem használható.

TEREDO

RFC 4380

Az ultimate megoldás. Global-unique cím keletkezik, melyhez nem a host IPv4 címét használjuk fel, hanem egy külső, privát IP címet és egy hozzá tartozó portot. Ehhez persze kell némi külső segítség is, ezt egy teredo szerver (teredo.ipv6.microsoft.com) biztosítja.
Ez abban is különbözik a többi megoldástól, hogy nem a korábban bemutatott csatornázást használja (*4.3. ábra IPv6 csomag az IPv4 csomagon belül*), hanem az IPv6 részt megfejeli egy UDP-be csomagolással - így a csomag azokon a routereken/tűzfalakon is átmegy, amelyek nem tudnak mit kezdeni az IPv6 payload-dal.

5 ELBÚCSÚZÁS

Nos, ennyi.

Nem tudom, mérted-e stopperrel az időt… de itt a végén azért már elárulhatom, hogy abszolút nem jelent semmit, mennyi időbe tellett végigolvasnod. A helyzet az, hogy volt egy remek vonatfényképem és ehhez pont passzolt ez a cím.

Te viszont nyugodtan olvasd végig a füzetet, lassan, alaposan megrágva minden kifejezést. És ha szeretnél jobban elmélyülni a témában, akkor ismételten csak a jóval bővebb könyvet tudom ajánlani:

> TCP/IP Alapok I-II:
> http://mivanvelem.hu/letoltheto-konyvek/

Amennyiben pedig bármilyen észrevételed lenne a könyvvel kapcsolatban, az alábbi címen tudsz elérni:

Petrényi József
jpetrenyi@gmail.com

6 JAVÍTÁSOK

Nocsak, még ebben a nyúlfarknyi írásban is akadtak javítanivalók.

6.1 1.1 VERZIÓ

- A 'Javítások' fejezet beszúrása.
- A 'világméretű' kifejezés kiirtása a szövegből.
- A 16. oldalon sajtóhiba egy IP címben.