Windows Server 2008

TCP/IP Alapok

I. kötet

V2.0

Petrényi József

2.0 verzió, azaz harmadik kiadás

Microsoft Magyarország 2010

Köszönetnyilvánítás:

*Habár a könyv megírásához meglehetősen sok forrást használtam fel, de külön szeretnék köszönetet mondani **Joseph Davies**-nek (Cable Guy), akinek írásai, könyvei a leginkább hatottak a szemléletemre, az elképzeléseimre.*
*Emellett meglepően sok hasznos információt kaptam a **Wikipediából** is. Lehet, hogy politikai területen használhatatlan, de alap informatikai ismeretekben erős.*

"- Akkor beszéljünk nyíltan. Nem hiányzik magának valami?
Tót gondolkozott.
- Csak a meggyfa szipkám - mondta aztán. - De majd faragok másikat.
- Nem tárgyi értelemben gondoltam. Én a tevékenység hiányának káros következményeire céloztam.

Tót megzavarodott. A feleségére nézett, aki azonban szintén várakozó kifejezéssel nézett vissza rá, mintha tőle várná a magyarázatot.

- Látom, nem értik - állapította meg Varró őrnagy. - Nézzék, Tóték. Egy sötét szobában a legkisebb zaj is megsokszorozódva hangzik. Nomármost, a semmittevés úgy hat a szervezetre, mint a sötétség a hallószervekre. Felerősíti a belső neszeket, káprázatokat okoz a látómezőn, zakatolást idéz elő az agyban. A katonáimmal, ha nincs semmilyen elfoglaltságuk, mindig levágatom és visszavarratom a nadrággombjaikat. Ettől tökéletesen megnyugszanak. Remélem, értik, mit akarok mondani?

Tóték összenéztek, még tanácstalanabbul, mint az előbb. Némi habozás után Mariska megjegyezte:
- Ha a mélyen tisztelt őrnagy úrnak le vannak szakadva a gombjai, azokat szívesen felvarrom.
- Maguk teljesen félremagyarázzák a szavaimat - legyintett az őrnagy elégedetlenül. - Legalább azt mondják meg, van-e véletlenül egy darab cukorspárga a háznál, ami jól össze van gubancolódva.
- Az biztosan akad - derült föl Mariska. - Miért van rá szüksége a mélyen tisztelt őrnagy úrnak?
- Mert ki akarom bogozni! - mondta a vendég idegesen. - Én nem bírok olyan tétlenül élni, mint maguk.
- Hát miért nem tetszik szólni? - kiáltott föl csengő hangján Ágika, aki - nem először - jobban átlátott a helyzeten, mint a felnőttek. - Hiszen mi ketten, az anyu meg én, sohasem szoktunk ölbe tett kézzel ülni.
- Hát mit csinálnak?
- Ilyenkor estefelé, amikor nincs jobb dolgunk, dobozokat szoktunk hajtogatni.

Az őrnagy szeme fölcsillant.

- Dobozokat? - ismételte. - Roppant érdekes! - kiáltott föl. - Miféle dobozokról van szó?"

Örkény István: Tóték

TARTALOMJEGYZÉK

1 TANULJ VELEM!

Rendhagyó könyv lesz[1]. Például rögtön itt az elején el fogom árulni, hogy nem vagyok ám olyan nagy felkent tudora a témának. Nyilván az alatt a húsz év alatt, amióta informatikával foglalkozom, ragadt rám ez meg az - de annyira mélyen, mint ahogy most tervezem, sohasem merültem még el a hálózatok működéseinek rejtelmeibe.
Pedig megérné.

Manapság ezen múlik minden. Ma már úgy nézünk egy hálózati kártya nélküli gépre, mint gyerekeink a szódásszifonra. (Jártál már úgy, hogy operációs rendszert telepítettél és az nem ismerte fel a gépedben lévő hálózati kártyákat? Idegtépő kinlódások szoktak ilyenkor következni, amíg pendrive-on valahogy fel nem varázsolod a hálózati kártya driverét.)

Persze, ami egyfelől kényelem, az a másik oldalról tömérdek inkompatibiltás, a harmadik oldalról rengeteg meghibásodási lehetőség, a negyedik oldalról pedig veszély. Ha én hozzáférek a dróton keresztül, akkor más is. Ezzel pedig bele is mentünk a hálózati biztonság témakörébe... abba, mellyel ez a könyv nem fog foglalkozni. Meg úgy egyáltalán, ez az írás nem fog túlzottan foglalkozni semmilyen érintőleges egyéb területtel sem. Még az is lehet, hogy IP cím sem lesz benne[2]. Hogyan kell konfigurálni egy DHCP vagy egy DNS szervert? Rengeteg könyv, whitepaper foglalkozik vele. Keresd meg, olvasd el.
Ebben a könyvben mi leginkább csomagokat fogunk kinyitni, megvizsgálni. Meg csomagokba rejtett csomagokat. Meg azokba csomagolt csomagokat. Meg... ugye, érthető a mintázat?
Gondolom, ijesztően hangzik. Tény, hogy a világ egyik legunalmasabb dolga az RFC-k olvasgatása.

Itt jövök be a képbe én. Egyfelől kifejezetten fontosnak tartom, hogy aki informatikával foglalkozik, tisztában legyen az alapokkal. Ne kezdjen el vakarózni, ha a Network Monitort kell használnia. Másfelől tudatában vagyok annak, hogy nálam is vannak fehér foltok. Azt is tudom, hogy tanulni úgy tudok a legjobban, ha feldolgozom az anyagot, megemésztem, majd elmagyarázom. Ez fog történni ebben a könyvben: együtt fogunk haladni előre, egyre többet és többet megértve a dobozolás rejtelmeiből. Harmadrészt pedig szeretem a kihívásokat: kíváncsi vagyok, sikerül-e egy ilyen száraz, hivatalos szöveget szórakoztatóvá tennem?

[1] Én már úgy látszik minden szakkönyvemet ezzel a mondattal fogom kezdeni. (-:
[2] Oké, ez már tényleg túlzás.

Látható, ez egy integratív könyv lesz. Körbebástyáztam magam szakkönyvekkel, előttem van az internet hatalmas információtömege - és hol innen, hol onnan szedek majd elő valami darabot. Aztán ezekből az infotéglákból fog felépülni a ház.
Én adom a maltert.

Egy nagyon fontos megjegyzés. Magyar nyelvű szakkönyvek esetén a szerzők mindig problémázni szoktak rajta, hogy lefordítsák-e magyarra a szakszavakat, vagy használják az angol eredetiket? Én úgy vágom át ezt a gordiuszi csomót, hogy deklarálom: a könyv nyelve se nem magyar, se nem angol. Hunglish.
Ebben az országban az informatikusok úgyis ezt a nyelvet beszélik.

Végül még egy megjegyzés. Habár az a célom, hogy egy nehéznek tartott területet könnyebben fogyaszthatóvá tegyek, de ez nem jelenti azt, hogy mindent a legvégsőkig szájbarágósan fogok elmagyarázni. Lesznek fogalmak például, melyeket eleve ismertnek fogok feltételezni. Ergo a mű élvezetéhez legalább félműveltnek kell lenni.

Jut eszembe. A könyv több helyen is egyfajta fejlődésregény. Azaz bevezetek egy fogalmat - valahogy. Megértjük. Használjuk. Aztán pár oldallal később kiderül, hogy az a bizonyos fogalom jóval bonyolultabb, és olyan formában, ahogy addig én használtam, valójában nem is működhetne. Viszont ha úgy vezettem volna be elsőre, ahogy valójában kinéz, akkor túl bonyolult lett volna megérteni. Természetesen ezekre a csalafintaságokra menetközben fény derül - de ha csak úgy felcsapod a könyvet és beleolvasol, lehet, hogy égnek áll a hajad.

Mindenesetre nem lesz sétagalopp: papír, ceruza, számológép legyen a közeledben.

Jelen kötethez tartozik egy második kötet, illetve egy 50 oldalas összefoglaló füzet is.
A teljes csomagot innen tudod letölteni:
http://mivanvelem.hu/letoltheto-konyvek/

2 ALAPOZÁS

2.1 HÓFEHÉRKE ÉS A HÉT RÉTEG

Azaz ne mondj a kereskedőknek semmit.

De ne szaladjunk ennyire előre.

A normális hálózati kommunikáció egyáltalán nem olyan egyszerű dolog, mint gondolnánk. Több részfeladatot kell benne megoldanunk, ráadásul ezeknek együtt is kell működniük. Tekintve, hogy gyártó mint égen a csillag, a feladat szabványosítás után kiáltott. Az Open System Interconnection (OSI) szervezet volt az, aki erre a nemes feladatra vállalkozott. Úgy képzelték el, hogy a hálózati kommunikációt rétegekre bontják, ezen belül pontosan definiálják az egyes rétegek feladatát, illetve a rétegek peremén zajló kommunikációt. A többi meg már a gyártók dolga.
A végeredmény egy olyan rendszer lett, mely két modellből állt össze:

- Egyrészt kialakult a 7 rétegből álló struktúra.
- Másrészt definiálták az egyes résztevékenységeket ellátó protokollokat.

A felsorolás fentről lefelé fog haladni, hiszen így alakul majd ki a valóságnak jobban megfelelő ábra. De olvasni alulról felfelé kell.

LAYER7: APPLICATION LAYER (ALKALMAZÁS RÉTEG)
Minden olyan alkalmazás, mely épít arra, hogy hálózatos környezetben dolgozik.

LAYER6: PRESENTATION LAYER (MEGJELENÍTÉSI RÉTEG)
Előemésztés az alkalmazási réteg számára. Az adatokat alakítja olyan formákra, melyeket az alkalmazások már közvetlenül tudnak értelmezni.

LAYER5: SESSION LAYER (VISZONYLATI RÉTEG)
A kommunikációban résztvevő alkalmazások közötti kapcsolat, az ún session menedzselése.

LAYER4: TRANSPORT LAYER (SZÁLLÍTÁSI RÉTEG)
A csomagok, illetve szegmensek tényleges eljuttatása a címzetthez.

LAYER3: NETWORK LAYER (HÁLÓZATI RÉTEG)
Útvonalkeresés, logikai címzések. Routerek.

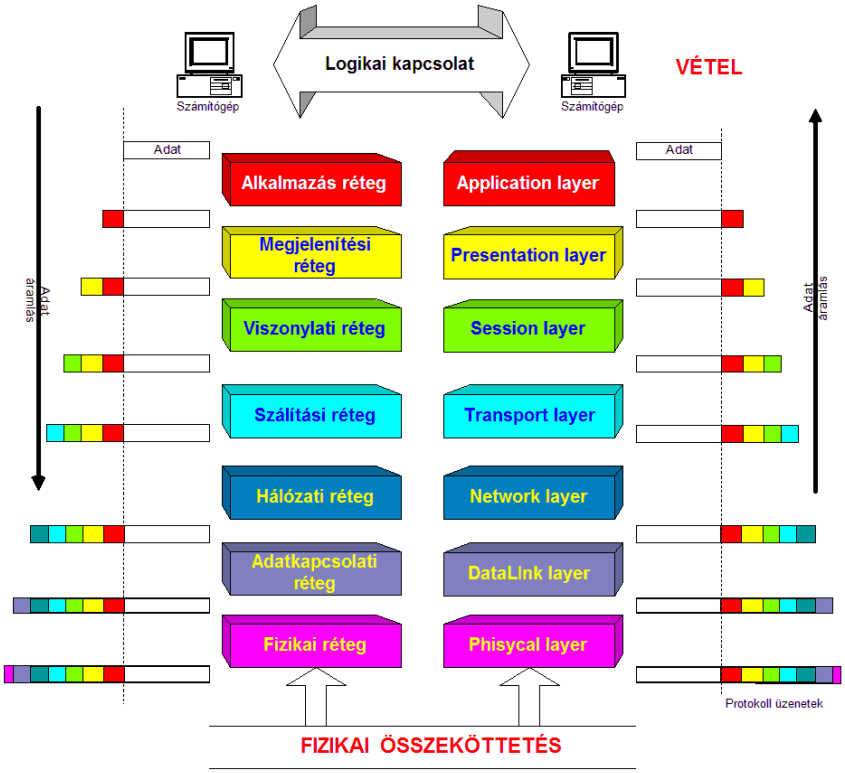
LAYER2: DATA LINK LAYER (ADATKAPCSOLATI RÉTEG)

Itt kezdünk el figyelni arra, hogy a kommunikáció alapvetően két résztvevő között zajlik. MAC address, ethernet.

LAYER1: PHYSICAL LAYER (FIZIKAI RÉTEG)

Az eszközünk és a drót közötti kapcsolat. Csatlakozok, tűkiosztások, feszültségek, kábelek.

Ez a hét réteg mindegyik hálózati szereplőnél megtalálható. Csak éppen aki küldeni akar, az fentről halad rajta végig lefelé, aki pedig fogadni, az fordítva.



2.1. ÁBRA RÉTEGEK ÉS CSOMAGOK (FORRÁS: WIKIPEDIA)

Az ábrán jól láthatóak azok a dobozolások is, melyeket ebben a könyvben boncolgatni fogunk. Látható, hogy van maga az adat. Az alatta lévő réteg ehhez hozzáteszi a saját információit (fejléc), majd újracsomagolja az egészet. A következő réteg már ezt a csomagot kapja meg, mellyel ugyanúgy bánik el, mint az előző réteg. Fogadáskor pedig rétegenkénti kicsomagolás zajlik le.

A hétrétegű modell:
http://www.citap.com/documents/tcp-ip/tcpip006.htm
http://www.leapforum.org/published/internetworkMobility/split/node12.html

No, mi is volt az elején az a mondat azokkal a kereskedőkkel?

Ez egy memoriter. A könnyebb megjegyzés érdekében kreáltak egy példamondatot:
Please **D**o **N**ot **T**ell **S**ales **P**eople **A**nything
Látható, hogy az első betűk alulról felfelé az egyes rétegek neveinek kezdőbetűivel egyeznek meg.

Oké, ez volt a pihenés.

Mi jut eszedbe, amikor szabványosításról hallasz? Nem kell szégyenlősnek lenned, magunk között vagyunk. Lassú bürokrata folyamatok... megdermedt formák... és egy csomó hájfejű, akik heteket vitatkoznak egy-egy névelőn. Hát, igen. Csakhogy. Vedd figyelembe, hogy rengeteg pénzről van szó. Az a bizonyos névelő az egyik gyártónak sokmilliós bevételt jelenthet, míg a másiknak sokmilliós bukást. Mert amíg a hivatalnokok az öntőformákkal tökölnek, az élet nem áll meg: a gyártók már termékeket dobnak piacra. Lásd a közelmúltból a VHS-Betamax csatát, vagy mostanában a Blu-Ray vs HD DVD háborúskodást.
Habár a hálózati folyamatok szabványosítása nem mai történet - akkoriban még nem voltak olyan harapósan aggresszívek a gyártók, mint ma - de mégis valami hasonló játszódott le. A publikum - a szabványosítókkal nem foglalkozva - a TCP/IP mellett voksolt. Azaz amikorra végre összecsiszolódott a gyönyörű 7 rétegű szabvány, addigra a világ már a TCP/IP alapú hálózatokat használta. 4 réteggel. Az OSI 1977-ben kezdett el dolgozni a modell kialakításán. Tudomásom szerint az egész még ma sincs befejezve. Ha elkezded bogarászgatni az egyes dokumentumokat, láthatod, hogy van köztük 1983-as-tól 2006-osig mindenféle.

ITU Data networks, open system communications and security, X-series
http://www.itu.int/rec/T-REC-X/en

A TCP/IP v4 pedig 1979-ben jött ki és 1983-ra terjedt el széleskörűen.

De ne hidd, hogy felesleges volt OSI-ék minden igyekezete. A 7 rétegű kommunikációs modell - igaz, hogy túl bonyolult lett - de minden körülmények között megállja a helyét. Platoni értelemben egy idea lett belőle. Egy séma, mely meghatározza a gondolkodásmódot. Azaz elméletben tudjuk, hogy a 7 réteg a tökéletes, de a gyakorlatban megelégszünk néggyel[3].

[3] Ettől függetlenül, ha egy hálózatos kolléga azt mondja, hogy L2 switch, biztos lehetsz benne, hogy a Layer2 ebben az esetben az OSI-ISO féle 7 réteg második rétegét jelenti. Mely nagyjából ugye a TCP/IP első rétegének felel meg.

Hozzá kell még tennem, hogy nem csak TCP/IP létezik a világon. Oké, ez a legismertebb... de ezen kívül virágzik még száz virág.

The Network Protocols Map Poster:
http://www.javvin.com/map.html

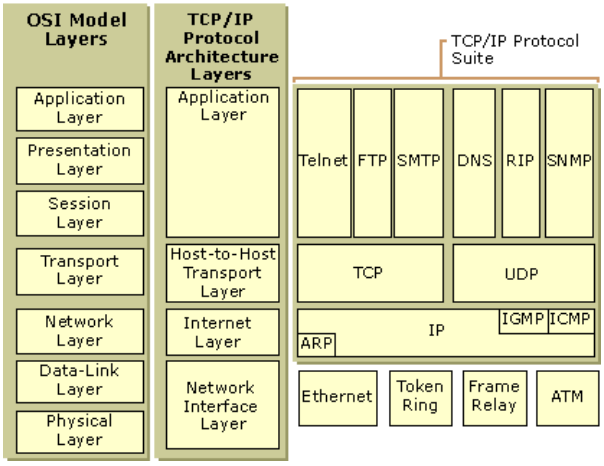
Vizsgáljuk meg végre azt a 4 réteget. Mely tulajdonképpen 5.

- LAYER5: Application layer (applikációs réteg)
- LAYER4: Transport layer (szállítási réteg)
- LAYER3: Internet layer (internet réteg)
- LAYER2: Data Link layer (adatkapcsolati réteg)
- LAYER1: Physical layer (fizikai réteg)

Ha nagyon akarjuk, a két modell rétegeit megfeleltethetjük egymásnak, de túl sok értelme nincs. Ez két különböző modell. Ha memorizálni akarjuk, akkor mondhatjuk azt, hogy az OSI-ISO modell felső három rétegét bezsúfolták egy rétegbe, mondván, hogy ha mindhárom réteg az alkalmazással foglalkozik, akkor ne szedjük már szét ennyire a feladatokat.

Tartozom még egy magyarázattal: miért is öt az a négy? Nos, azért, mert még szakmán belül sem egyértelmű, hogy érdemes-e az alsó két réteget külön kezelni. Van, ahol összevonják, van ahol nem. Mivel ez a könyv a Microsoft TCP/IP megvalósításával foglalkozik, ezért mi speciel összevontan fogjuk e kettőt kezelni, méghozzá Network Interface Layer (hálózati eszköz réteg) néven.

Miután már ennyi mindent tudunk, nézzük is meg, hogyan néznek ki ezek a dobozolások egymás mellett:



2.2. ÁBRA AZ OSI ÉS TCP/IP RÉTEGEK EGYMÁS MELLETT

2.2 FOGALMAK

Kezdők számára lehet, hogy zavaró lesz, hogy itt rögtön olyan fogalmakkal találkoznak, melyek magyarázatához még nem ismert fogalmakat használok fel.
Nem baj.
Valahogy mindenkinek el kell indulni. Lehet, hogy most elsőre valami érthetetlen, de érthetővé válik, ha később lapozunk vissza, amikor már elkezdünk használni egy fogalmat.

2.2.1 SZABVÁNYOK

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 persze 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

2.2.2 Kinek szól az üzenet?

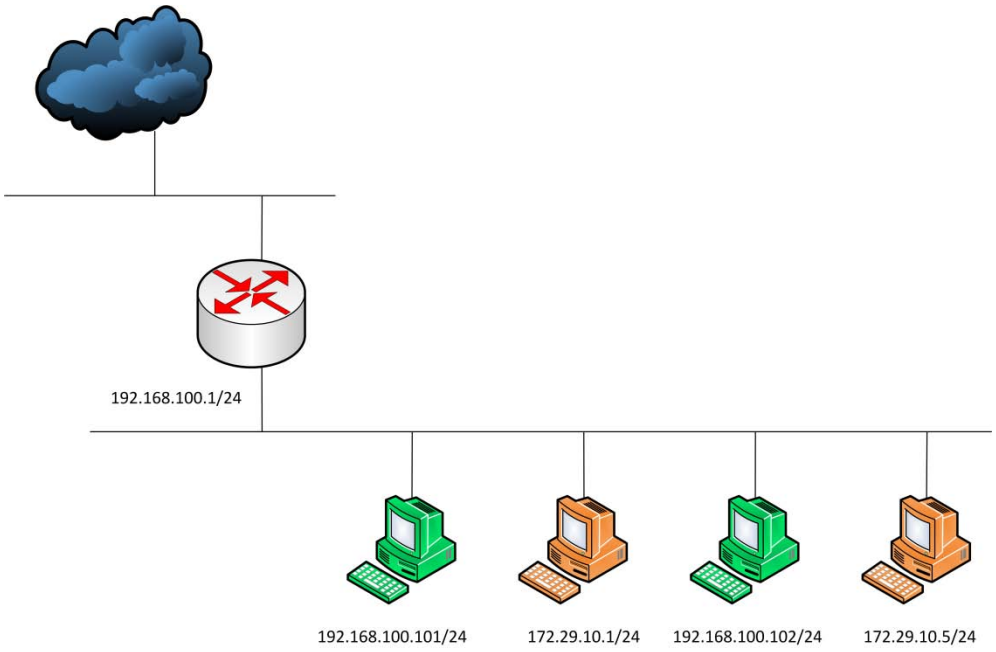
- Unicast : Egy feladó, egy címzett.
- Multicast :Egy feladó, több - kiválasztott - címzett.
- Anycast : Egy feladó, több - kiválasztott - címzett. De ha bármelyik megkapta, akkor a többiekhez már nem jut el.
- Broadcast : Egy feladó, mindenki más címzett.

2.2.3 Host vs node

Ez viszonylag gyors tisztázás lesz. A host, az egy számítógép, mely részt vesz a hálózati kommunikációban. A hostnak vannak hálózati csatolói. A legtöbbször ezek hálózati kártyák. Ezeket a csatolókat nevezzük node-nak.
Magyarul egy host akár több node-dal is rendelkezhet.

2.2.4 SUBNET VS LINK

A subnet (alhálózat) egy nagyobb hálózatnak a része. Legfontosabb jellemzője, hogy a subneten lévő gépek azonos címzési protokollokat használnak, közvetlenül látják egymást, nem kell router ahhoz, hogy kommunikálni tudjanak egymással.
A link pedig olyan node-ok halmaza, melyek egy router mögött helyezkednek el. Egyáltalán nem biztos, hogy a linken lévő node-ok ugyanabba a subnetbe tartoznak.

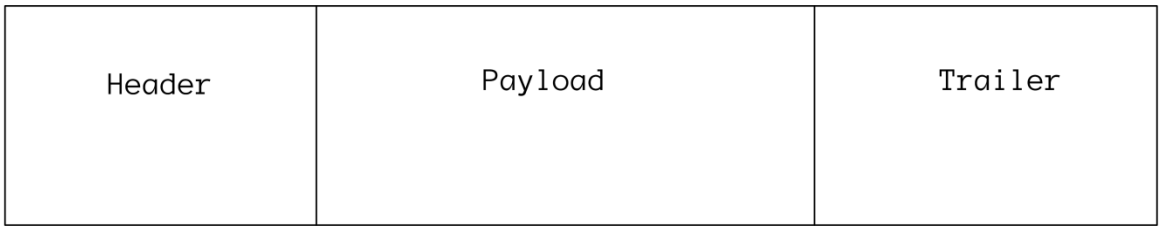


2.3. ÁBRA LINK ÉS SUBNET

A fenti képen mind a négy munkaállomás ugyanazon a linken található. Ennek ellenére a zöld és a narancssárga munkaállomások különböző alhálózatba tartoznak. (Az IP címek alapján pedig csak a zöld munkaállomásoknak van esélyük arra, hogy elérjék az internetet.) Az azonos színű gépek tudnak kommunikálni egymással, a különböző színűek pedig közvetlenül nem.
Ebben a felállásban egy broadcast el fog jutni mind a négy munkaállomáshoz. Ha el szeretnénk kerülni ezt a felesleges forgalmat, akkor az azonos színű munkaállomások számára külön VLAN-okat kell létesítenünk a switchen.

2.2.5 AZOK A FRÁNYA DOBOZOK

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



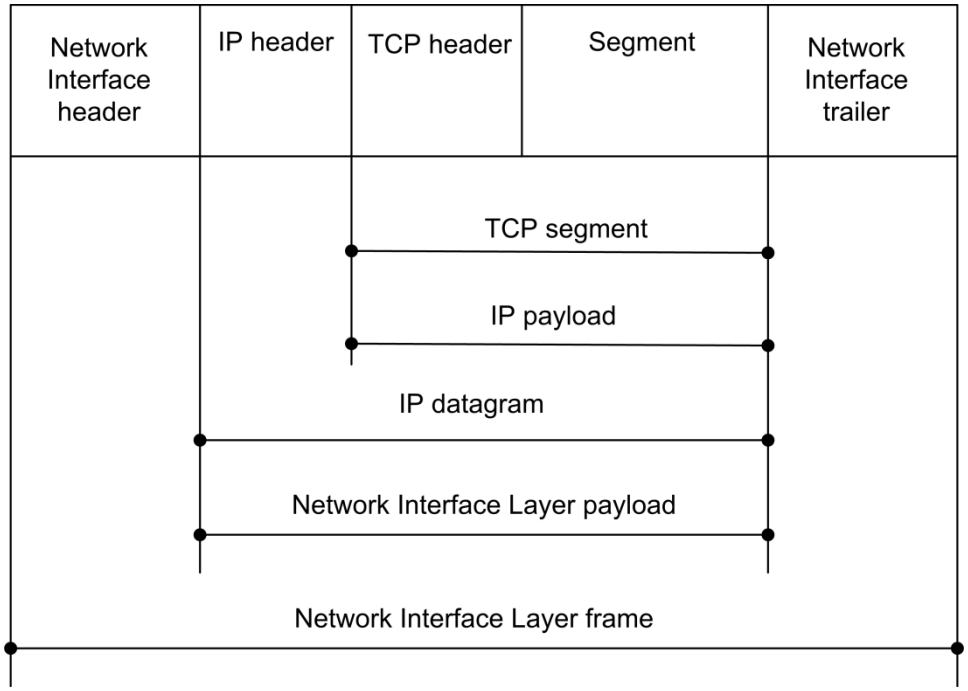
2.4. ÁBRA ÁLTALÁNOS CSOMAGOLÁSI SÉMA

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.



2.5. 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.

2.2.6 HÁLÓZATI ESZKÖZÖK

Milyen eszközökre gondolok? Olyanokra, melyek segítenek kiépíteni és fenntartani egy számítógépes hálózatot - és ezek közül is azokra, melyek hálózatokat, hálózatrészeket kötnek össze valamilyen formában.

2.2.6.1 HUB

Az OSI modell szerint értve L1 szinten működő eszközök, fizikailag kötnek össze node-okat. Minden forgalmazott csomag kimegy minden node-hoz, nincs semmilyen portszeparáció. Gyakorlatilag egy buta elosztó.

2.2.6.2 SWITCH

Annyiból hasonló a hubhoz, hogy ez is elosztó, de már képes szűrni a forgalmat, azaz egy konkrét node-hoz csak az a forgalom megy ki, melyet neki szántak, plusz a broadcast. Tulajdonképpen egy intelligens elosztó, az L2 rétegben.

2.2.6.3 BRIDGE

Szintén L2 eszköz, azonos névterű, azonos protokollokat használó alhálózatokat köt össze. Pontosabban, egy alhálózatot, mely több részre oszlott. Ez azt jelenti, hogy a bridge routolást nem végez, a csomagokat csak forwardolja egyik helyről a másikra. Manapság a bridge és a switch fogalmak meglehetősen összemosódtak.

2.2.6.4 GATEWAY

Különböző protokollokat használó hálózatokat összekapcsoló eszköz. Mind a hét rétegben működhet.

2.2.6.5 ROUTER

L3 rétegben dolgozó eszköz. Különböző névterű - de azonos protokollokat használó - alhálózatokat köt össze. Később részletesen is foglalkozunk vele.

3 A HÁLÓZATI ESZKÖZ RÉTEG PROTOKOLLJAI

Akárhogy is nézzük, a dróton vad elektronok tekernek. Mi a valóságban csak az elektromos jelszint változását tudjuk érzékelni. Ebből a folyamatos áramlásból kell diszkrét csomagokat formálnunk, emberi (hah!) logika szerint.

Erről szól a hálózati eszköz réteg.

Látható, hogy neki van a legnehezebb dolga. Ő készíti el a legkülső dobozt, ő is érzékeli fogadó oldalon a dobozok határait. A többiek már biztos körvonalakkal rendelkező dobozokat kapnak, melyekbe bátran pakolgathatják bele a saját csomagjaikat.

Történelmileg úgy alakult ki, hogy a konkrét hálózatok kiterjedésétől függően különböző protokollok terjedtek el. Ezt ma már elég nehéz elmagyarázni, tekintve, hogy mostanság nem ritka a több kilométert is átérő lokális hálózat (Local Area Network, LAN), miközben számlázási okokból kis távolságokon is szoktak nagykiterjedésű hálózatokra (Wide Area Network, WAN) jellemző protokollokat (pl. PPP) használni. Én mindenesetre LAN/WAN bontásban fogom sorbavenni ezeket.

3.1 LAN TECHNOLÓGIÁK

Tehát ott járunk, hogy a Network Interface Layerben vagyunk, azaz az első csomagolás el is kezdődik a... Data Link Layerben. Hogy nemrég azt írtam, Windows környezetben nincs is olyan? Dehogyisnem, csak begyömöszöltük a NIL-be.
Mivel még teljesen az elején vagyunk, nem árt, ha sorra vesszük, mit is kell biztosítania egy ilyen csomagolásnak:

- ELHATÁROLÁS. Értem ez alatt, hogy pontosan érzékelnünk kell az egyes csomagok határait, illetve csomagon belül a header/trailer és payload határokat.
- PROTOKOLL AZONOSÍTÁS. Értelemszerűen szintén nagyon fontos. Ha valamit kisnyúlként csomagoltam be, akkor azt ne mikrohullámú sütőnek csomagolják ki, mert annak nagyon béna lesz.
- CÍMZÉS. Ki a feladó, ki a címzett?
- INTEGRITÁS. A címzettnek legyen lehetősége meggyőződni arról, hogy teljesen épen kapta-e meg a csomagot.

Kezdjük élesíteni a fókuszt. Ezen a nagy fejezeten belül a következő hálózati technológiákkal fogunk foglalkozni: Ethernet, Token Ring, FDDI, IEEE802.11. Mindegyik technológián belül különböző *frame formátumokkal* futhatunk össze.

Félő, hogy itt egy kissé zavarossá válhat a fejtegetés. Azt már tisztáztuk, mi a különbség packet és frame között, de milyen a viszony a továbbítási technika, a frame formátuma és a protokoll között?

Nos, konkrétan: az Ethernet az egy kerettovábbítási technika, az Ethernet keretekhez viszont többfajta frame formátum is tartozhat: Ethernet II, Ethernet 802.3, Ethernet 802.3 SNAP. Keverni őket tilos: ha az egyik kommunikáló fél mondjuk az Ethernet II formátumba csomagol, akkor a másik, amelyik az Ethernet 802.3 SNAP formátumra számít, nem fogja érteni, dacára, hogy a keret továbbítási technikája mindkét esetben Ethernet.

IEEE LAN's, azaz 802.x szabványok, részletes ismertetéssel:
http://www.citap.com/documents/tcp-ip/tcpip007.htm

A protokollokkal kapcsolatban általánosságban elmondható, hogy a protokoll az egy előírt viselkedési forma. Mint egy szigorúan rögzített cselekvéssor egy diplomáciai fogadáson. Mint egy klasszikus tánc. "A fiatalúr a hölgy derekára helyezi a jobb kezét, a hölgy a fiatalúr vállára helyezi a bal kezét, a fiatalúr jobb lábával könnyedén előrelép, a hölgy jobb lábát előretéve velefordul." Ez a konkrét tánc szabálya, aki nem így táncol, azt kivezetik a bálteremből.
A frame formátum ezzel szemben egy technikai megegyezés. Egy apróság, mely kell a protokoll szerinti viselkedés megvalósításához. Mint az azonos nyelvű beszéd a tánc közben, vagy a mindkét fél által ismert zászlójelek egy tengeri hadgyakorlaton.

Nézzünk egy példát az informatikából.
A fogadóhoz beesik 10 darab *keret*. Ethernet kártyákat használunk, *a frame formátum* pedig - mind a feladónál, mind a címzettnél - Ethernet II, azaz a fogadó ki tudja csomagolni a keretek tartalmát. A fregmentált IP datagramokat össze tudja rakni (ehhez ismernie kell az *IP protokollt*[4]), így megkapja az IP payload-ot, a TCP szegmenst - abból pedig ki tudja csomagolni az eredeti információt (ismerve a *TCP protokollt*[5]), mondjuk azt hogy "*MAIL FROM: info@cegnev.hu*". Ez pedig, mint tudjuk, *az SMTP protokoll*[6] egyik utasítása, egész konkrétan az elküldendő levél feladóját állítja be.

Protocols and protocol stacks:
http://www.citap.com/documents/tcp-ip/tcpip008.htm

[4] Layer2 - IP
[5] Layer3 - TCP
[6] Layer4 - Applikáció

3.1.1 ETHERNET

Hawai. És aloha.
Ott kezdődött. A hawai egyetemen fejlesztettek ki egy alapvetően rádióhullámok átvitelére optimalizált rendszert. Kőkemény 9.6 Kbps-t tudott. Úgy hívták, hogy ALOHA.

Ethernet:
http://en.wikipedia.org/wiki/Ethernet

Erre a rendszerre alapozva hoztak össze 1972-ben Palo Altoban, a Xeroxéknál (igen, ott ahol az egeret, az ablakos GUI-t és valami fénymásolót is feltaláltak) egy 2.94 Mbps-t tudó hálózatot, melyet Ethernetnek neveztek el. A név arra utal, hogy a hálózati kommunikáció a levegőbe, azaz az éterbe ordibáláson alapul. De erre később még visszatérünk.
1979-ben 3 nagy cég (**D**igital, **I**ntel és a **X**erox) szövetséget kötöttek, hogy általános szabvánnyá neveljék ki az Ethernetet. Ezt a változatot hívják Ethernet II-nek, vagy DIX-nek[7]. Akkor 10 Mbps-t tudott.

1981-ben lett az Ethernet IEEE 802.3 néven nemzetközi szabvány. 1995-ben jött a 100 Mbps sebességű fast Ethernet... és azóta sincs megállás. Jelenleg éppen a 40 Gbps, illetve 100 Gbps változatok fejlesztése folyik szorgalmasan.

Függetlenül a sebességtől, az Ethernet keretek szállítási módszere nem változott. A frame-k ugyanazt a médiát (koax, csavart érpár vagy üvegszál) használják, méghozzá úgy, hogy egyszerre csak egy keret tekerhet rajta. Azaz a feladó beleordítja az információját az éterbe és ha éppen mindenki más csendben volt, akkor az el is jut a címzettekhez. Ha éppen akkor valaki más is beleordított, akkor mindkettő szégyenlősen elhallgat, véletlenszerű ideig várnak, majd újrakezdik.

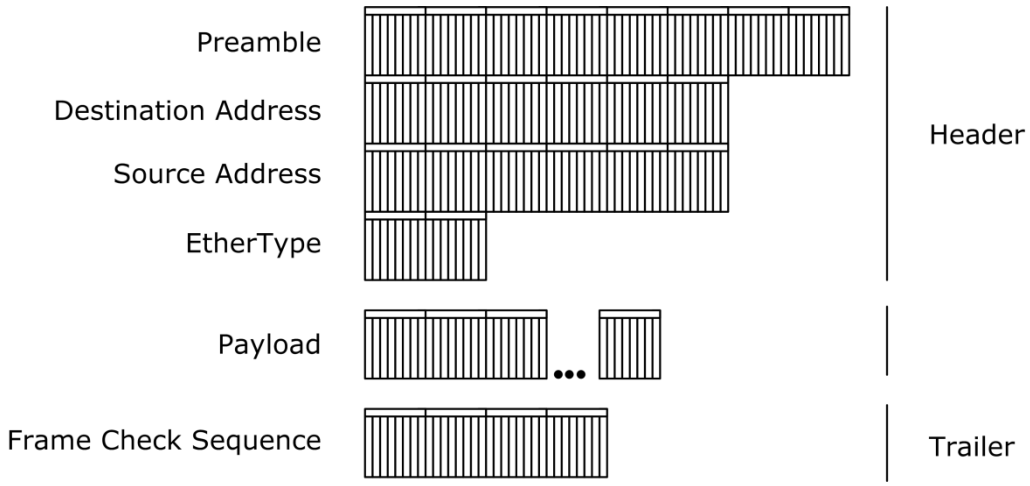
[7] Egész pontosan úgy történt az eset, hogy Robert Metcalfe, a szabvány egyik tulajdonosa otthagyta a Xeroxot, megalapította a 3Com-ot, majd indulásként vette rá a három nagyot a szabvány támogatására.

Szoktak erre egy népszerű hasonlatot is használni: olyan ez, mint egy udvarias baráti beszélgetés egy nagyobb társaságban. Valaki elkezd egy sztorit, befejezi. Megszólal másvalaki, elmondja a történetét, elhallgat. Aztán ketten szólalnak meg egyszerre, észlelik és elhallgatnak. Zavart csend, majd az egyik újrakezdi, később pedig jön a másik is.
Ezt a viselkedést informatikus nyelven úgy hívják, hogy carrier sense multiple access with collision detection, azaz CSMA/CD.

Nos, ennyi bevezető után immár semmi sem menthet meg minket a keretek boncolásától.

3.1.1.1 ETHERNET II

RFC 894



3.1. ÁBRA ETHERNET II FRAME FORMÁTUM

Mivel ez az első frame formátum, így itt mondom el: eszedbe se jusson ezeket megtanulni! Ilyen ábrából legalább kétszáz lesz még. Ha csak nem a keretek, illetve a hálózati datagramok lesznek eljövendő életed családtagjai, bőven elég, ha kimásolod egy pendrive-ra a könyvet, és ha hálózati forgalmat kell elemezned, akkor megkeresed a megfelelő ábrát. A sűrűbben előfordulókat úgyis meg fogod jegyezni. A kevésbé sűrűn előfordulók meg… véleményem szerint az emberi agyat sokkal értékesebb dolgok tárolására tervezték.

Jöjjenek a részletek.

PREAMBLE: Ő a szirénázó felvezető autó a sorban. Szószerint szirénázik: felváltva tartalmaz egyeseket és nullákat. Pontosabban a 8 bájtból 7 bájt sziréna, az utolsó pedig botlik egyet: 10101011, jelezve azt, hogy innentől jön ~~a nagykövet~~ a tartalom. Ez a sorozat a szinkronizáláshoz kell, érdekes módon a network monitorozáskor nem is látszik.

DESTINATION ADDRESS: A címzett MAC címe.

SOURCE ADDRESS: A feladó MAC címe.

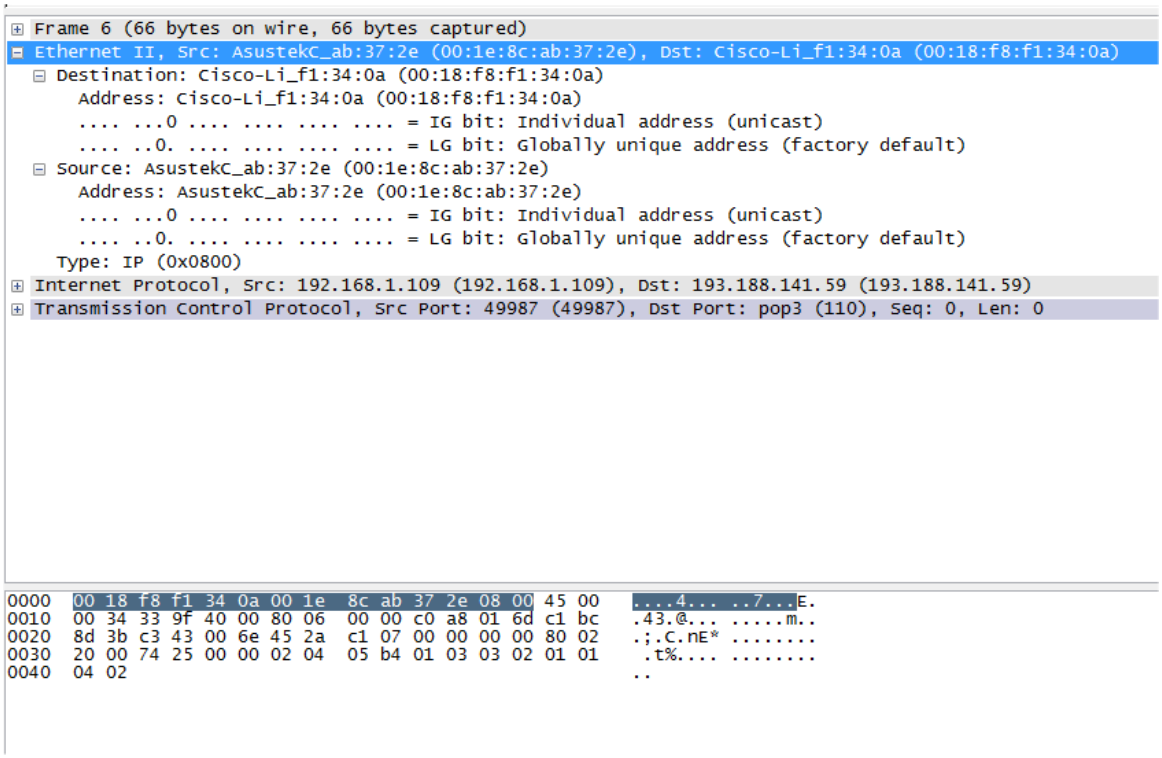
ETHERTYPE: Ez a két bájtos érték mondja meg, hogy a payload kinek a mije, azaz mire számítson a felette lévő réteg. Az esetek nagy részében az értéke 0x0800, mely IP csomagot jelent - de lehet 0x0806 is, mely ARP üzenetet takar.

PAYLOAD: A felette lévő rétegnek szóló információ. Maximális mérete 1500 bájt lehet, a minimális értéke - a megbízható ütközésdetektálás miatt - 46 bájt. Ha nincs ennyi, akkor kipótolják.

FRAME CHECK SEQUENCE (FCS): Magyarul CRC[8]. Ez tulajdonképpen egy lenyomat[9] magáról a frame-ről[10], a frame mögé tűzve. A fogadó ugyanúgy ráküldi az algoritmust és amennyiben ugyanazt az értéket kapja, mint ami az FCS mezőben van, akkor a frame tartalma biztosan nem változott meg útközben.
Az FCS szintén nem látszik a network monitorozásánál.

Akkor mi látszik? Nézzük meg[11].

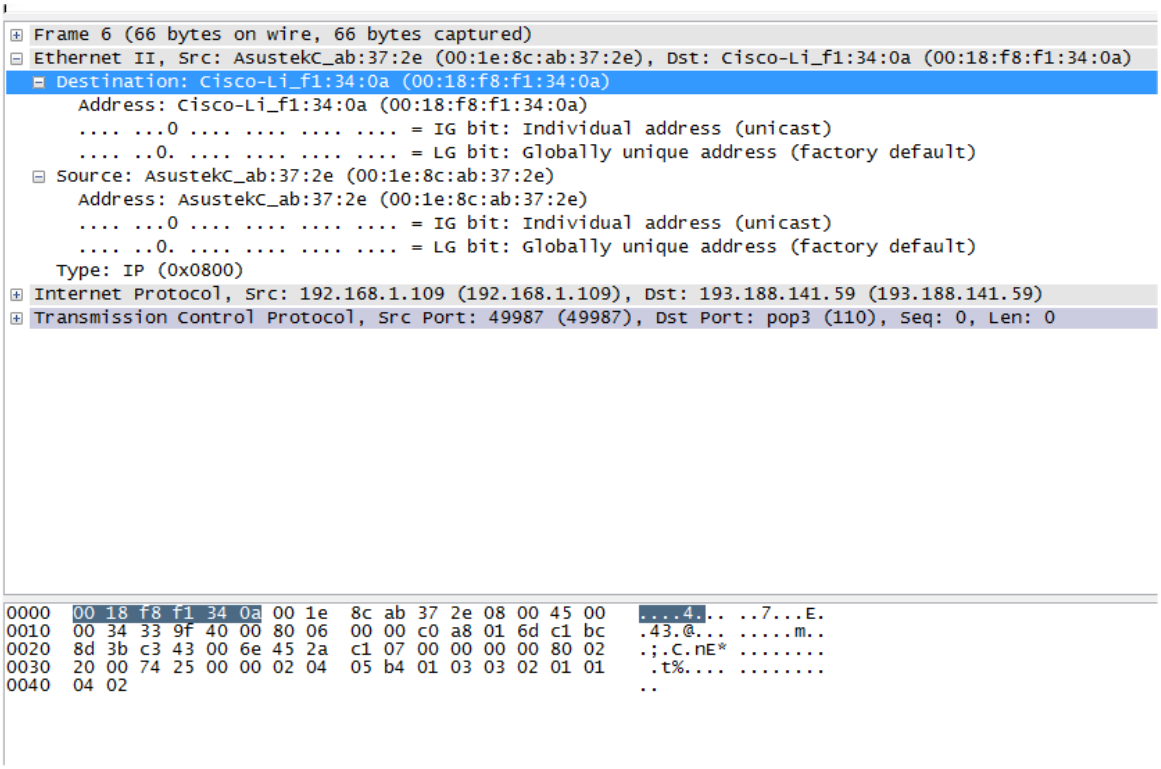
[8] Cyclical Redundancy Check.
[9] Egész pontosan az történik, hogy a keretet, mint bitsorozatot elosztják egy 33 bites prímszámmal, majd a mindig 32 bites maradék kerül az FCS mezőbe.
[10] Nyilván nem beleértve az FCS és a Preamble mezőket.
[11] Az ábrák előállításához a Wireshark programot használtam.



3.2. ÁBRA EGY ETHERNET II FRAME

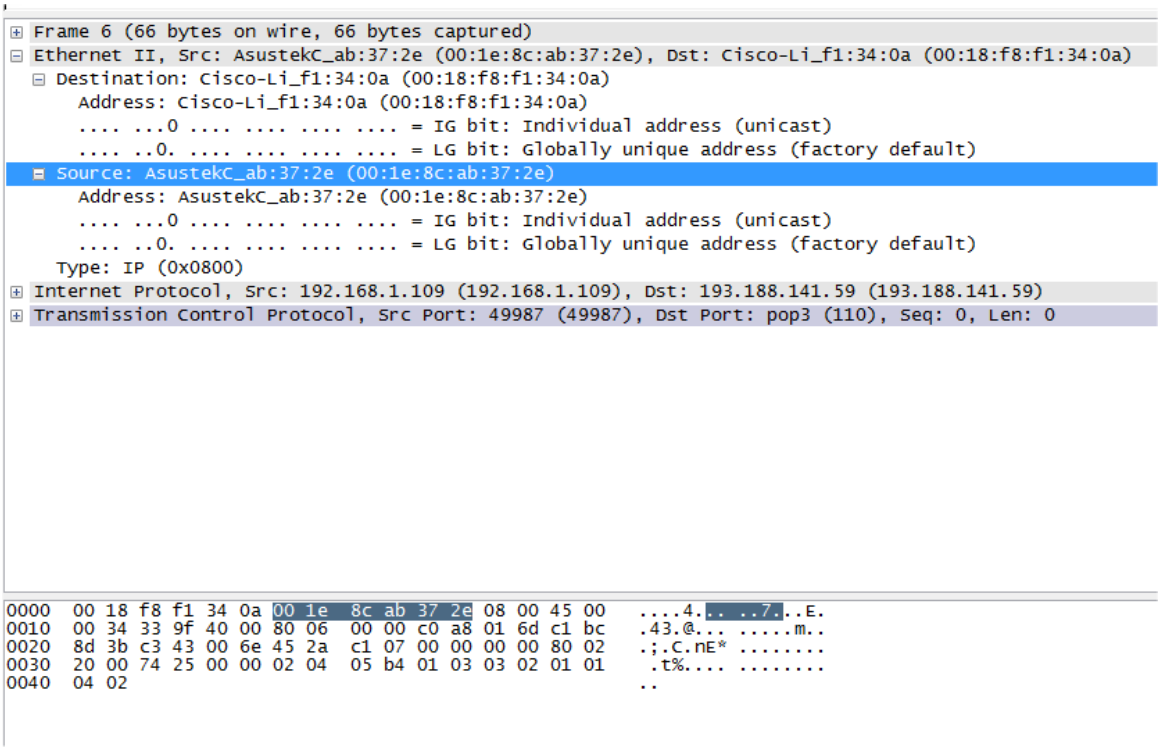
Később már nem fogok minden egyes capture ábrát ennyire szájbarágósan elmagyarázni, de mivel ez az első, ezért itt egy kicsit bő lére eresztem a szöveget.
A fenti ábrán az elkapott hálózati folyam hatodik keretét látjuk. (Ha a legfelső sort lenyitnám, akkor ott egy összefoglalót találnék a keretről.)
A második sor lett lenyitva, ez ugyanis maga az Ethernet II keret fejléce.
Az ábra legalján pedig maga a teljes frame található, a hexa editorokban megszokott bájt formátumban. A bemázolt rész mindig azt mutatja, amit a felette lévő ablakban kijelöltünk.

Keressük meg az ábrán (*3.1. ábra ETHERNET II frame formátum*) említett részeket.



3.3. ÁBRA DESTINATION ADDRESS

Mint írtam, a Preamble mező nem látható, így a legelső mező a Destination Address. Fent a program szépen kibontja, hogy milyen eszközről is van szó (a MAC cím gyártóspecifikus), alul viszont csak a hexadecimális, hat bájtnyi érték látható.



3.4. ÁBRA SOURCE ADDRESS

Ugyanez látható a Source Address mezőnél is.

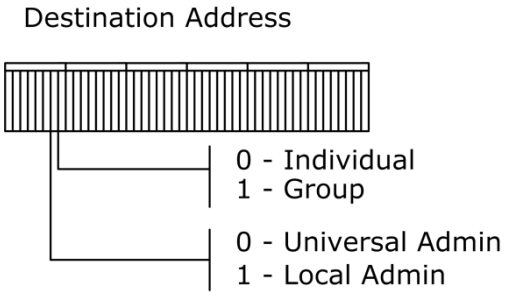
```
⊞ Frame 6 (66 bytes on wire, 66 bytes captured)
⊟ Ethernet II, Src: AsustekC_ab:37:2e (00:1e:8c:ab:37:2e), Dst: Cisco-Li_f1:34:0a (00:18:f8:f1:34:0a)
  ⊟ Destination: Cisco-Li_f1:34:0a (00:18:f8:f1:34:0a)
      Address: Cisco-Li_f1:34:0a (00:18:f8:f1:34:0a)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
  ⊟ Source: AsustekC_ab:37:2e (00:1e:8c:ab:37:2e)
      Address: AsustekC_ab:37:2e (00:1e:8c:ab:37:2e)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    Type: IP (0x0800)
⊞ Internet Protocol, Src: 192.168.1.109 (192.168.1.109), Dst: 193.188.141.59 (193.188.141.59)
⊞ Transmission Control Protocol, Src Port: 49987 (49987), Dst Port: pop3 (110), Seq: 0, Len: 0

0000  00 18 f8 f1 34 0a 00 1e  8c ab 37 2e 08 00 45 00   ....4... ..7...E.
0010  00 34 33 9f 40 00 80 06  00 00 c0 a8 01 6d c1 bc   .43.@... .....m..
0020  8d 3b c3 43 00 6e 45 2a  c1 07 00 00 00 00 80 02   .;.C.nE* ........
0030  20 00 74 25 00 00 02 04  05 b4 01 03 03 02 01 01    .t%.... ........
0040  04 02                                             ..
```

3.5. ÁBRA AZ ETHERTYPE MEZŐ

Végül az utolsó ábrán láthatjuk az EtherType mező értékét, mely határozottan 0x0800-nak látszik, azaz IP csomagról van szó.
A következő két sor - melyeket nem bontottam ki - alkotja a payload-ot. Ezekkel ráérünk később is foglalkozni.

És ennyi. Kérdés lehet még, hogy ha frame, akkor muszáj, hogy legyen trailer is. Hol van?
Elbújt. Mint írtam, a trailer itt a Frame Check Sequence, mely a capture fájlban nem látható.

Ha már itt kerültek elő, ejtsünk még pár szót a MAC címekről.



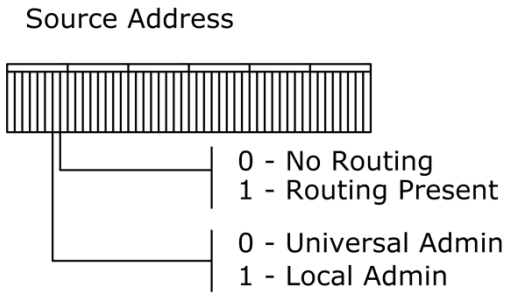
3.6. ÁBRA DESTINATION MAC ADDRESS

A MAC Address alapvetően egy hatbájtos érték, ahol az első három bájt a hardver gyártójára utal, a második három pedig a konkrét eszköz azonosítója.
Az első bájt legalsó két bitje speciális szerepkörrel bír, mind a feladó, mind a címzett címében.

DESTINATION ADDRESS, I/G BIT (INDIVIDUAL/GROUP)
Azt határozza meg, hogy a megcélzott cím egyedi (unicast, azaz individual), vagy csoportos (multicast, azaz group). Unicast esetben a bit értéke 0, multicast esetben 1. A broadcast (FFFFF) ebben az esetben a multicast alesete.

DESTINATION ADDRESS, U/L BIT (UNIVERSAL/LOCALLY ADMINISTERED)
Azt határozza meg, hogy a megcélzott cím eredeti, beégetett (universal) vagy lokálisan felülírt (locally administered). Alaphelyzetben egy hálózati eszköznek a világon egyedi MAC címe van - bizonyos esetekben viszont felülírhatjuk ezeket az értékeket.

Nézzük a forrás oldalt.



3.7. ÁBRA SOURCE MAC ADDRESS

Vészcsengő. Bejelzett? Nem? Az baj.
Lapozz csak vissza ehhez az ábrához: *3.4. ábra Source Address.* Láthatod, hogy a capture fájlban gyönyörűen ki van részletezve mindkét oldalon mind a két bit jelentése. Márpedig a forráscímnél ugyanúgy I/G bitnek értelmezi az első bitet... miközben a fenti rajzon én azt rajzoltam be, hogy No Routing/Routing.

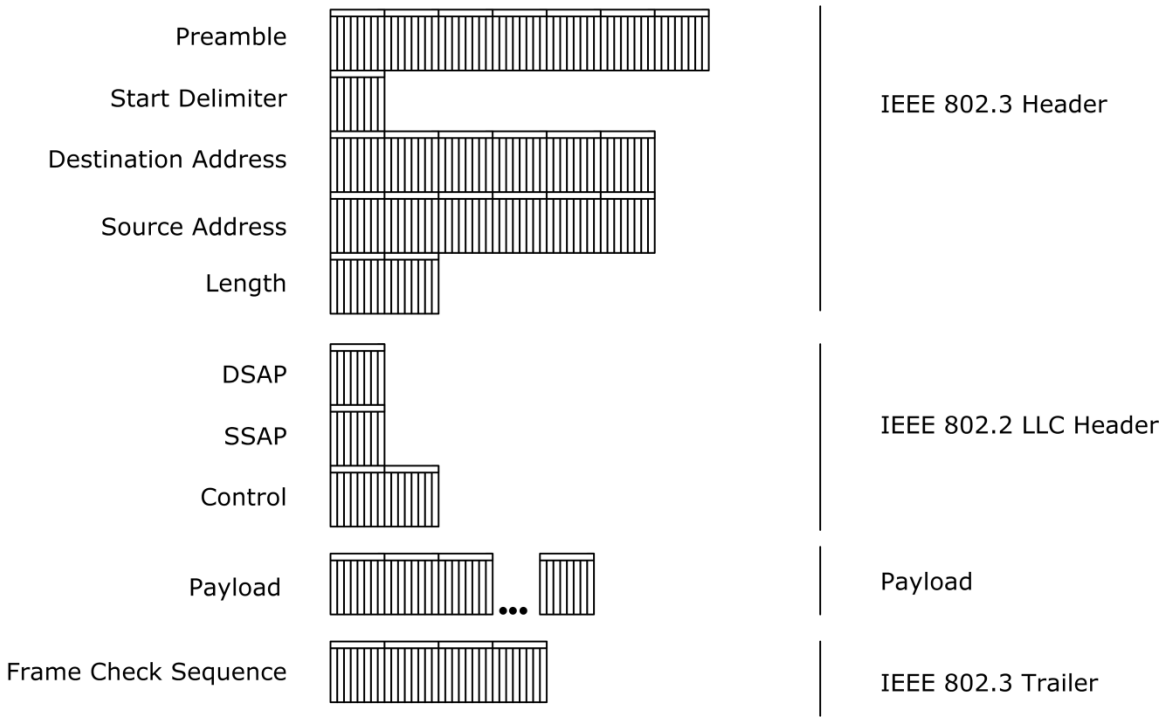
Akkor most mi van?

Az van, hogy csaltam. Az I/G bit a feladónál igazából semmilyen jelentéssel nem bír. Gondold el, létezhet egyáltalán multicast feladó? Na ugye. Ebből kifolyólag az Ethernet technológiákban az értéke masszívan nulla, ezt látjuk a capture fájlban is.
De van, amikor használjuk. A Token Ring például MAC szinten routolgat, neki kifejezetten jól jön, hogy ezen a biten tud jelezni.

3.1.1.2 ETHERNET 802.3

RFC 1042

Itt már egyből a lecsóba csapunk.



3.8. ÁBRA AZ ETHERNET 802.3 FRAME FORMÁTUM

Ez egy elég érdekes öszvér. Mint látható, van ugyan keret (IEEE 802.3), de abban egy újabb csomagolás, úgynevezett kapszula van. Ez a belső kapszula a Network Interface Layer már korábban említett egyik alprotokollja, a Logical Link Control (IEEE 802.2).

Most nézzük csak a külső keretet. Nagyjából ugyanazt látjuk, mint az előző frame formátumnál: a Preamble ugyan két részre szakadt, de ez csak logikai szétválasztás, hiszen annyi történt mindösszesen, hogy az utolsó, különböző bájtot máshogy neveztük el. A Destination Address, a Source Address teljesen ugyanaz[12] és az FCS is. A különbség a Source Address után kezdődik. Itt egy két bájton ábrázolt hossz értéket találunk, miközben az Ethernet II esetében ezen a helyen a protokollazonosító volt. (A hosszt úgy kell elképzelni, hogy az LCC header első bájtjától a payload utolsó bájtjáig terjedő bájtok száma. Vegyük észre, hogy ez gyakorlatilag megegyezik az Ethernet II esetében a payload-dal.)

[12] Azzal az apró eltéréssel, hogy a szabvány megengedi a 2 bájtos MAC címeket is.

Most képzeljünk el egy olyan szervert, mely mind a két frame formátumot ismeri. Honnan fogja az tudni, hogy ez a két bájt most hossz érték, vagy protokolltípus? Nagyon egyszerűen. Tudjuk, hogy a payload értéke minimum 46 bájt, maximum 1500. Egész egyszerűen ebben az intervallumban nincsen protokollazonosító kód. Tehát ha a Source Address utáni érték mondjuk 1000, akkor Ethernet 802.3 frame formátummal állunk szemben.

Nézzük most magát a 802.2 LCC datagramot.

DESTINATION SERVICE ACCESS POINT, DSAP: A címzett oldalán milyen protokollra számíthatunk.

SOURCE SERVICE ACCESS POINT, SSAP: Ugyanez, forrás oldalon.

Nyilván ezek kódszámok lehetnek. Ami minket a leginkább érdekel, az az, hogy az IP csomag értéke 0x06. Kellemetlen tény, hogy az iparban gyakorlatilag ezt a kódot senki nem használja, helyette a 0xAA kód terjedt el. De ezzel majd a következő frame formátumnál foglalkozunk bővebben.

CONTROL: Ez egy egy- vagy kétbájtos kód. Arra vonatkozik, hogy magát az LCC kapszulát hogyan készítettük el. Értékei:

- TYPE1: A kapszula egy LCC datagram, a Control bájt értéke 0x03. Ez egy meglehetősen megbízhatatlan kapszulázás, nem foglalkozik a szállítás megbízhatóságával.
- TYPE2: A kapszula egy LCC session része, hibadetektálás, sorbarendezés, meg ilyenek.

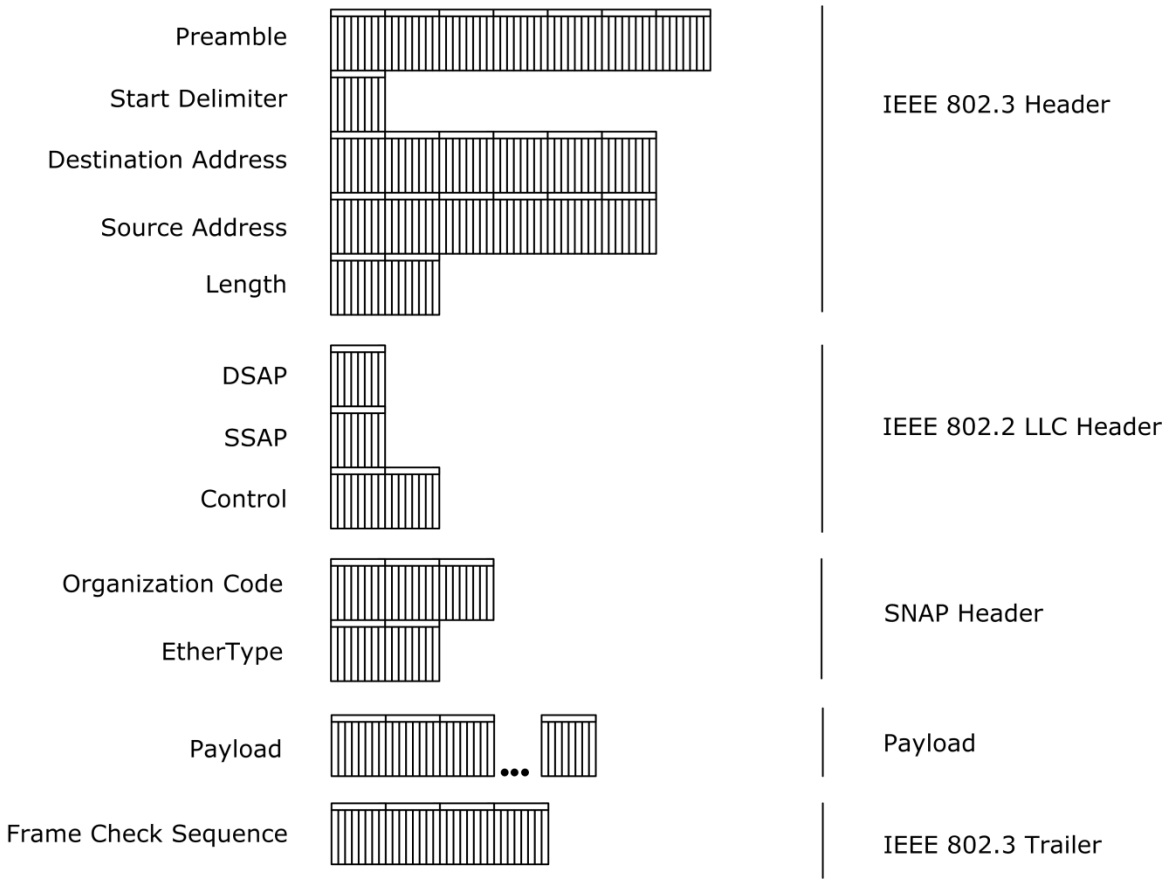
Az IP datagrammok és ARP csomagok szállítása kizárólag TYPE1 módon történik, tehát nekünk elég a 0x03 értéket megjegyeznünk. (Állítólag van egy TYPE3 is, de ezzel még ritkábban találkozhatunk, mint a TYPE2-vel.)

PAYLOAD: A felettünk lévő réteg eredeti datagramja. Természetesen a kipótlás itt is játszik.

3.1.1.3 ETHERNET 802.3 SNAP

RFC 1042

Bonyolítsuk tovább a tornyot.



3.9. ÁBRA IEEE 802.3 SNAP FRAME FORMÁTUM

Mint korábban is írtam, az Ethernet 802.3 frame formátum DSAP/SSAP mezőjében a 0x06 értéket ritkán használják az IP protokoll megadásához. Helyette egy úgynevezett Sub-Network Access Protocol-on (SNAP) keresztül adják meg a payload protokollját. Ekkor a DSAP/SSAP mezőkben a 0xAA értékkel jelezzük, hogy nem itt kell keresni a protokollazonosítót, hanem egy újabb kapszulázáson belül, az SNAP headerben. És látható, ott is vigyorog a jó öreg EtherType.

Menjünk sorban. Az IEEE802.3 header teljesen ugyanaz, mint korábban, ne is vesztegessünk sok szót rá.

DSAP, SSAP: Jelentése ugyanaz, mint korábban, a fix értékük: 0xAA.

CONTROL: Jelentése ugyanaz, a fix érték: 0x03.

ORGANIZATION CODE: Ez valamikor jelentett valamit, de mára már nem sok értelme maradt. A három bájtból az első mondta meg, hogy melyik szervezet felelős a további két bájt értelmezéséért. Jelenleg a fix érték: 0x00-00-00.

ETHERTYPE: Értelmezése - és értékei - megegyeznek az Ethernet II frame formátum leírásánál írtakkal.

PAYLOAD: Továbbra is maga az IP datagram. De számoljunk csak: a különböző dobozolásokra azért már elment némi bájt, azaz a payload méretkorlátai is megváltoztak. A határok immár: 38-1492 bájt. (Az utóbbi szám nem ismerős? Nem, nem Kolumbusz. MTU? Az bizony. Mi is pontosan ez az MTU? Maximum Transmission Unit, azaz a dróton átküldhető maximális hasznos teher. De hát nem pont erről beszéltünk?)

Az IEEE802.3 trailer maradt a régi.

Ezzel végig is futottunk az Ethernet frame formátumokon. Mit saccolsz, melyik a legelterjedtebb? Nyilván a legegyszerűbb, az Ethernet II. Nem is igazán értem, mi húzódott meg amögött a logika mögött, mely létrehozta a 802.3 szabványt, mely egy újabb tekeredéssel visszatért az Ethernet II formátumhoz, elpazarolva egy csomó hasznos bájtot, gyakorlatilag nulla előnyért cserébe.

A Windows Server 2008 és a Windows Vista fejlesztői is hasonlóképpen gondolkodtak, ugyanis alapbeállításban az Ethernet II formátumot preferálják. Megértik a 802.3 formátumot is, csak utálják. Azaz ha bármilyen kérés (ARP) érkezik hozzájuk 802.3 formátumban, akkor Ethernet II formátumban válaszolnak vissza. Ha az érdeklődő veszi a lapot és ő is átáll Ethernet II-re, akkor minden rendben. Ha nem, akkor úgy járt. A szerver/Vista nem fog vele kommunikálni.

Ezen a hozzáálláson egy registry változó átállításával lehet módosítani.
HKLM\System\CurrentControlSet\Services\Tcpip\Parameters
Itt kell létrehozni a következő változót:
ArpUseEtherSNAP [reg_dword]
Ha nulla az értéke, akkor letiltja a 802.3 SNAP frame formátumot. Ha egy, akkor engedi.

3.1.2 TOKEN RING: IEEE802.5 ÉS IEEE 802.5 SNAP

RFC 1042

A modellvasút.

Ha az Ethernet kommunikáció esetében azt mondtam, hogy az udvarias baráti társaság beszélgetéséhez hasonlít, akkor a Token Ring leginkább egy modellvasúthoz. Egy olyanhoz, mely egy hosszú, tekergős, de azért zárt pályán jár körbe-körbe. A szerelvényen van egy mozdony és egy tehervagon - ez a Token - az állomások pedig felpakolják rá, illetve leszedik róla a kereteket. Egyszerre csak egy szerelvény megy a pályán, és egyszerre csak egy keretet tud szállítani.
Pontosítsuk a hasonlatot. Nem egy darab szerelvény - token - létezik. Igaz, hogy egyszerre csak egy tekerhet a síneken, de bármelyik állomás kiveheti a tokent a forgalomból és beállíthat egy sajátot.
Pontosítsuk még tovább a hasonlatot. A síneken vagy token (üres szerelvény) vagy frame közlekedik. Az az állomás, amelyik küldeni akar, leszedi a tokent és betesz egy frame-t. Amelyik megkapta a neki szóló frame-t, leszedi és elindít egy üres tokent. [13]

Hasonlítsuk össze, a működési módból kifolyólag milyen elvi különbségek lesznek az Ethernet technológiához képest?

- Kevésbé lesz érzékeny az állomások számának növelésére. Amíg az Ethernet esetében 30 állomás fölött tragikus kakofóniává fajult a kommunikáció, addig a Token Ring esetében nagyjából 200 munkaállomásnál kezdett el a rendszergazda switch vásárlásán gondolkodni. (A maximális érték viszont 250 körül volt.)
- Az ütközésdetektálás elmaradása miatt nincs minimális frame méret.
- A Token Ring átviteli sebessége kevésbé függött az állomások számától. Meglehetősen stabilan hozta a maximum közeli értéket, míg az Ethernet sávszélessége [14] drasztikusan csökkent az állomások csatlakoztatásával. Mindezt úgy, hogy a Token Ring a 16 Mbps sebességet tartotta az Ethernet meg csak akkor tudta a 10 Mbps-t, ha két gépet kötöttünk össze... cross kábellel.

[13] Jogos lehet a kérdés, mi történt vezetékszakadáskor? Hiszen ezzel megszűnne a gyűrű, azaz minden állomás leesne a hálózatról. A válasz a csatlakozóban rejlik. A TR csatlakozók olyan spéci bonyolult szerkezetek, hogy már a patch panelben észlelik, ha baj van (elromlott a kártya, a túloldali csatlakozó, megszakadt a vezeték) és rövidzárral kiközölik a hibás szakaszt. Nem véletlenül voltak ezek olyan drágák.
[14] Kaptam olyan visszajelzést, hogy ez nem sávszélesség, hanem adatátviteli sebesség. Teljesen igaz, de az informatikai közbeszédben ez a változat is meggyökeresedett.

Azaz minden tekintetben jobb volt a Token Ring, mint az Ethernet. Én hat évig dolgoztam akkoriban olyan cégnél, ahol tisztán Token Ring hálózatunk volt - és végig úgy éreztem magam, mint aki egy kiválasztott cégnél dolgozhat. Három pesti kerületben elhelyezkedő, öt telephelyes hálózatot fogott át egy gyűrűnk. Nem is titkoltan néztem le az ethernetes kollégákat, amikor sorolták, milyen problémáik vannak. Nekünk ugyanis csak egy volt: valahogyan meg kellett keresni a hálózat üzemeltetésére a pénzt. A Token Ring ugyanis piszok drága. Még annál is drágább. A kábelezés, a csatlakozók... na és persze a hálózati kártyák. Az ezredforduló táján egy Token Ring kártya 45-50000 forint körül volt, miközben egy Ethernet kártya kábé a tizedébe fájt.
Aztán egyszercsak a világ megrázta magát és elrohant a Token Ring technológia mellett. Az Ethernet árai még jobban lezuhantak, miközben a sebessége drasztikusan megnőtt. Persze a bekiabálós technikából következő 'max. 30 node egy linken' korlát továbbra is megmaradt - de megjelentek az olcsó és okos switch-ek, melyek drasztikusan lecsökkentették az ütközések esélyeit[15]. Innentől kezdve a francot sem érdekelte az ingadozó sebesség, ha az egyébként böszme nagy érték környékén ingadozott. Ha egyáltalán.

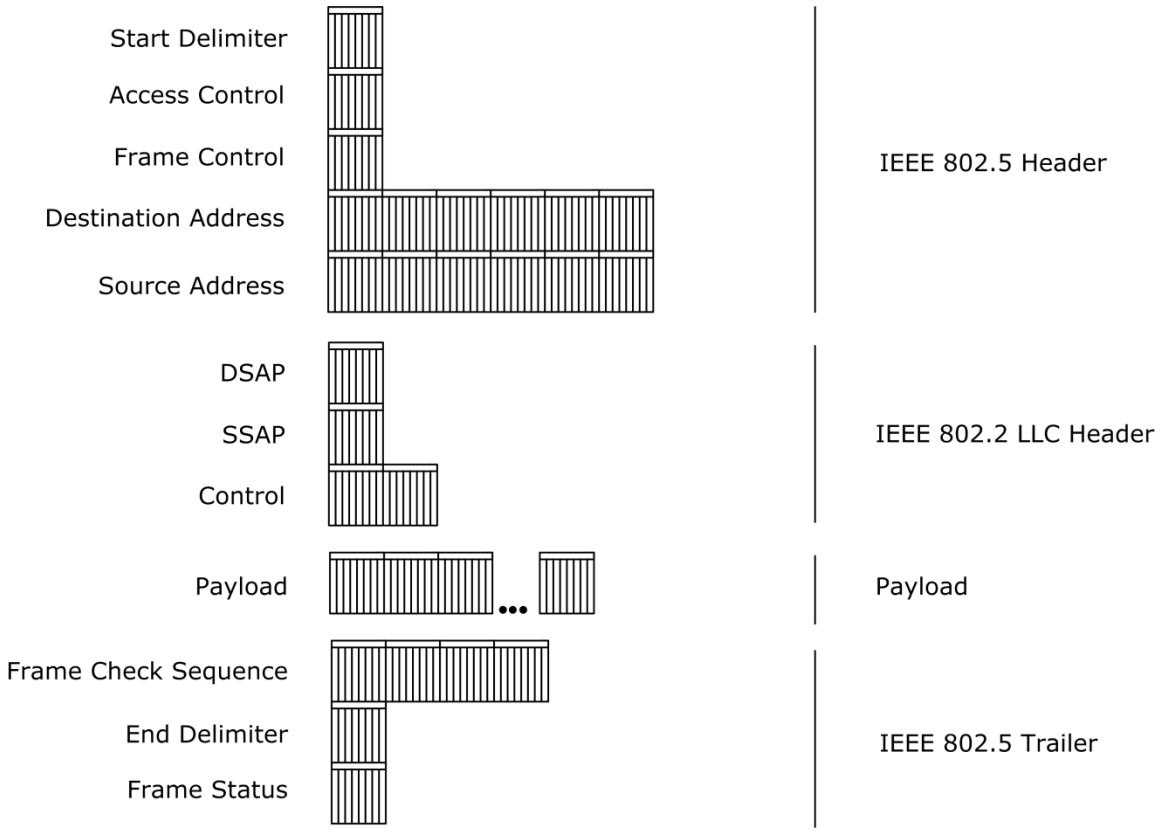
Igaz, időközben a Token Ring is kijött egy 100 Mbps sebességű technológiával (üvegszálon), az árolló is csökkent (ma már 15e körül lehet kapni 4/16/100 Mbps TR kártyát), de ha megnézzük, az Ethernet jelenleg milyen sebességhatárokat ostromol, hosszú távon nem sok esélyt adok a Token Ringnek.

És akkor most jöjjön egy kis történelemóra.

Magát az elvet Olaf Soderblum dolgozta ki, még 1969-ben. Az IBM megvette tőle, majd Token Ring néven dobta piacra, 1984-ben. (Hogy közben mit csináltak? Gondolom, jó alaposan átgondolták.) A technológiából nemzetközi szabvány 1985-ben lett, a keresztségben az IEEE 802.5 nevet kapta.

Ennek a frame szerkezetét mutatja be az alábbi ábra.

[15] Szaknyelven szólva különválasztották a broadcast domaint a collision domaintől. Azaz a switch megjegyezte, hogy melyik lábán milyen MAC címmel rendelkező állomás lóg (illetve ha kaszkádolva lettek a switchek, akkor milyen állomások) és csak a nekik szóló csomagokat irányította arra. Ezzel jelentősen lecsökkent az ütközések valószínűsége. Ettől függetlenül persze a broadcast kimegy minden munkaállomásra, mely a switchen lóg. Illetve ez csak akkor igaz, ha a switchen egy darab VLAN (Virtual LAN) fogja össze az összes portot. Amennyiben a portokat több VLAN-ba szedjük szét, akkor a broadcast VLAN-on belül marad. Azaz a VLAN jelenti egyben a broadcast domaint is.
Fontos, hogy ne keverjük össze a VLAN-t a subnettel. Az előző definiálása és kezelése L2 szinten (OSI modell) történik, MAC address alapján és jellemzően egy switch végzi, míg az utóbbié L3 szinten, IP címek alapján és jellemzően egy router a felügyelő. (Habár ma már léteznek L3 switchek is.)



3.10. ÁBRA IEEE 802.5 FRAME FORMÁTUM

START DELIMITER: Egy bájt, mely jelzi a keret kezdetét. A bitek helyén roppant rejtélyes nevű szimbólumok találhatók, bizonyos J és K ügynökök. (Annyit tudtam kibogozni róluk, hogy valami Manchester kódolású adatok, melyeket egyébként a rádiótechnikában használnak.) Túl sokat nem kell foglalkoznunk velük, az Ethernetnél megszokott módon ezek sem látszódnak a capture fájlban.

ACCESS CONTROL: Szintén egy bájt, melyben a biteknek külön-külön van értelme.

- Első három bit: A token prioritását állítja be. Izgi, nem? Frame szinten tudjuk szabályozni, mennyire fontos adatról van szó. Hét különböző értéke lehet.
- Második három bit: Tulajdonképpen ez egy prioritás reset bitsorozat. Azt a prioritást adja meg, melyre a token akkor áll, miután leszedték róla a szállított frame-t.
- Hetedik bit: Egyre áll, ha a szerelvény elhalad az úgynevezett monitorozó állomás előtt. Amennyiben a monitorozó állomás elé olyan szerelvény ér, melynek már magasra van állítva ez a bitje, akkor az állomás kikapja a forgalomból.
- Nyolcadik bit: Jelzi, hogy ami most jön, az token (0) vagy frame (1).

FRAME CONTROL: Egy bájt hosszú. A bitek jelentése:

- Első két bit: Azt mondja meg, hogy a frame LLC frame, vagy MAC frame. (Ugye emlékszünk, a Token Ring varázsol MAC szinten is.)
- Következő négy bit: Amennyiben MAC management frame-ről van szó, akkor jelzi, hogy a MAC alrétegen belül konkrétan milyen tipusúról. Ezek lehetnek Purge, Claim, Token vagy Beacon tipusúak.
- Utolsó két bit: Csak úgy foglalt.

Purge, Claim, Token és Beacon:
http://www.techfest.com/networking/lan/token.htm

DESTINATION ADDRESS: A megcélzott állomás MAC címe. Vigyázat, lehet MAC management multicast cím is.

SOURCE ADDRESS: A feladó MAC címe.

Az IEEE 802.2, azaz LLC header totálisan ugyanaz, mint az Ethernet keretek (802.3, 802.3 SNAP) esetén.

PAYLOAD: Azt már írtam, hogy a Token Ring frame-nek nincs alsó korlátja. (Oké, negatív szám nem lehet.) De mi a helyzet a felső értékkel? Mennyi a Token Ring MTU-ja? Attól függ. Mitől? Egy bonyolult algoritmustól, mely még tán a napéjegyenlőségek ciklikus váltakozását is figyelembe veszi. A pontos leírása az RFC 1042-ben található.

FRAME CHECK SEQUENCE: A már jól ismert négybájtos CRC.

END DELIMITER: Egybájtos érték, a már korábban említett J és K szimbólumokkal. A két utolsó bitjének értelmezhető értékei vannak.

- Intermediate Frame Indicator: Ez a frame az utolsó a sorban(0), vagy vannak még(1)?
- Error Detected Indicator: Jelzi, hogy ellenőrzéskor épp volt-e a CRC.

FRAME STATUS: Egy bájt, a következő értékekkel:

- Address Recognized Indicator: Amennyiben a megcélzott állomás észreveszi a küldési szándékot, akkor ezt a bitet magasra állítja, jelezve ezzel, hogy jó volt a címzés.
- Frame Copied Indicator: Amikor a megcélzott állomás hálózati kártyája felkapja a frame-t, magasra állítja ezt a bitet, jelezve ezzel, hogy minden oké, megkapta a keretet.

Mindkét indikátor megduplázva szerepel, mivel ezt a mezőt már nem védi a CRC ellenőrzés.

Az utolsó három mező - Frame Check Sequence, End Delimiter és a Frame Status - nem látszódnak a capture fájlban.

A címben szerepel az SNAP név is. Gondolhatnád, hogy itt majd fogok foglalkozni a Token Ring SNAP frame formátumával... de tévedsz. Nem fogok vele különösebben foglalkozni. Az Ethernet 802.3-nál már leírtam, hogyan kapunk a normál, LLC-vel megbolondított frame formátumból SNAP frame formátumot, és mivel a kavarás lényege az volt, hogyan jelezzük, hogy a payload jelenleg *IP datagram* - amely ugye egy szinttel feljebb van - így az átalakítás, azaz a sznaposítás a Network Interface Layerben is ugyanúgy történik, függetlenül a továbbítási technológiától. A DSAP/SSAP értékek ugyanúgy 0xAA-ra váltanak, a Control 0x03-ra, bejön az értelmezhetetlen Organization Code mező és jóbarátunk, az EtherType is. Igen, Token Ringnél is EtherType-nal hívják. Fura, mi?

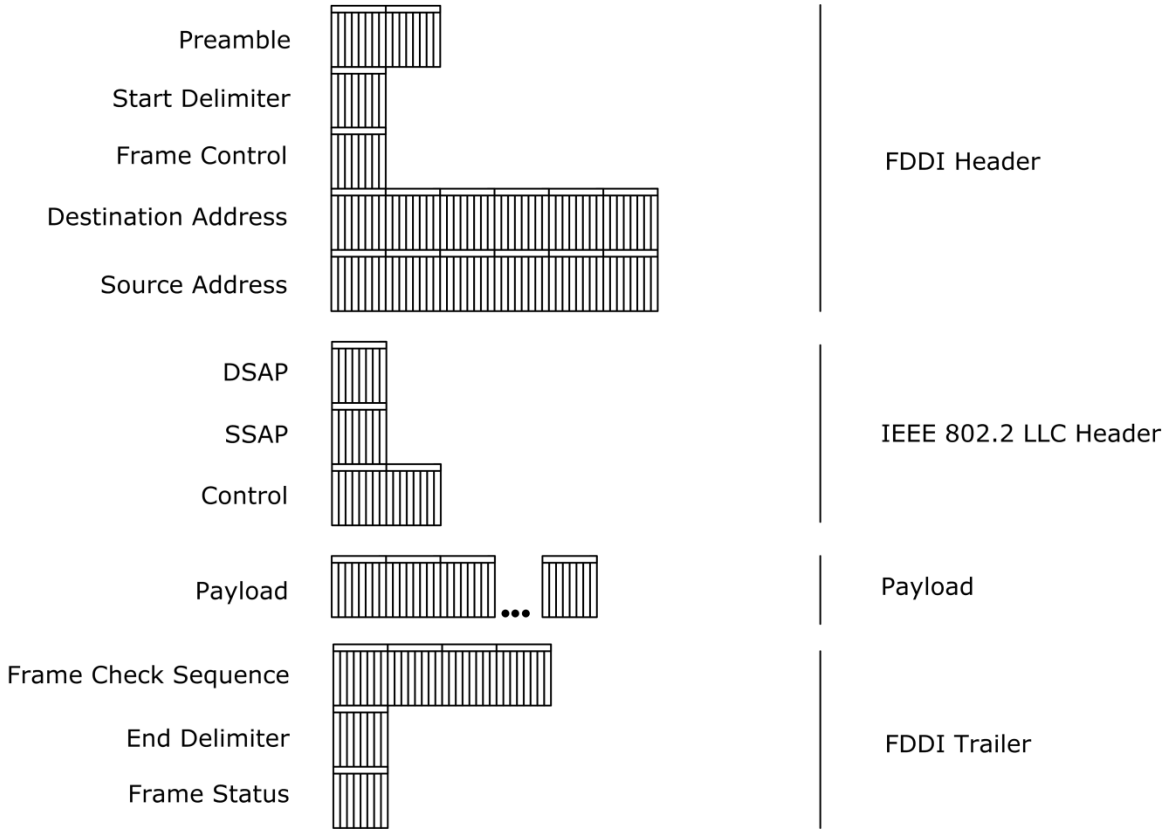
3.1.3 FIBER DISTRIBUTED DATA INTERFACE: FDDI

RFC 1188

A Token Ringhez hasonlóan itt is modellvasutazunk. Csak éppen itt két sínpálya is fut egymás mellett, biztos, ami biztos alapon. És mindkettő piszkosul gyors - azaz a valóságban üvegből van. (A technológia létezik réz vezetéken is, de akkor CDDI-nek hívják.)
Mik a fontosabb jellegzetességei?

- Elsősorban a megbízhatóság. Normál esetben mindig csak az egyik gyűrű működik, ha az megszakad, akkor lép be a körbe a második. Ha mindkét gyűrű megszakad ugyanazon a ponton, akkor a két gyűrű összekötésével továbbra is működik az adatátvitel. (A forgalom iránya ellentétes a két gyűrűn.)
- A leginkább elterjedt változat sebessége 100 Mbps. Ez valamikor fantasztikus sebességnek számított, manapság már csak olyan "szódával elmegy" kategória. Ha nem foglalkozunk a megbízhatósággal és mindkét gyűrűt használjuk egy időben, akkor 200 Mbps-t érhetünk el.
- Az üvegszál miatt nagy távolságokat tud lefedni. 200 kilométer még nem probléma.
- A viszonylag nagy adatátviteli sebesség, a nagy távolság és a nagy megbízhatóság miatt elsősorban gerinchálózatok (backbone) kiépítésére használják.

A frame szerkezete meglehetősen hasonlít a Token Ring technológiánál megismert frame szerkezetére.



3.11. ÁBRA FDDI FRAME FORMÁTUM

PREAMBLE: Nem hiszem, hogy az eddigiek után bármi újat is tudnék mondani. Ő a tülkölés a kanyarban.
START DELIMITER: Teljesen hasonlatos a Token Ring hasonló mezőjéhez, olyan szinten, hogy itt is megtaláljuk J és K ügynököket.
Sem a Preamble, sem a Start Delimiter mezők nem láthatók a monitorozó programokban.

FRAME CONTROL: egy bájt, a következő bontásban:

- Class bit: Szinkron (1) vagy aszinkron (0) keretről van-e szó? A szinkron keret garantált sávszélességet és válaszidőt jelent, míg az aszinkron kerettel elért sávszélesség dinamikusan változik, a lehetőségeknek megfelelően.
- Címhosszúság bit: Hány bájtosak a címek? 2 (0) vagy 6 (1)? Megjegyzem, az első változat gyakorlatilag kizárt, hogy a vadonban is előforduljon.
- Maradék hat bit: formátumleírás, illetve kontroll bitek. (Választék akad bőven, tokenből is és keretből is jutott rendesen.)

DESTINATION ADDRESS: Teljesen megegyezik az Ethernet fejezetben tárgyaltakkal.

SOURCE ADDRESS: Dettó.

IEEE 802.2 LCC HEADER: A szokásos.

PAYLOAD: Az IP Datagram.

FRAME CHECK SEQUENCE: A jól ismert CRC.

END DELIMITER: A Start Delimiterhez hasonlóan J és K szimbólumokat tartalmaz. Tekintve, hogy az FDDI keretnek nincs rögzített hossza, ez a mező jelzi a keret végét.

FRAME STATUS: két bájt, a következő bontásban:

- ADDRESS RECOGNIZED INDICATOR: Amennyiben a megcélzott állomás észreveszi a küldési szándékot, akkor ezt a bitet magasra állítja, jelezve ezzel, hogy jó volt a címzés.
- FRAME COPIED INDICATOR: Amikor a megcélzott állomás hálózati kártyája felkapja a frame-t, magasra állítja ezt a bitet, jelezve ezzel, hogy minden oké, megkapta a keretet.
- ERROR INDICATOR: Ha az FCS hibás, akkor az állomás magasra állítja ezt a bitet.

Természetesen ebből a keret formátumból is létezik SNAP változat.

FDDI:
http://www.javvin.com/protocolFDDI.html
http://www.laynetworks.com/FDDI.htm

3.1.4 IEEE 802.11, AZAZ WIFI

Kezdjük azzal, hogy ez nem is egy szabvány. Hanem sok.

3.1. TÁBLÁZAT

IEEE szabvány	Megjelenés	Működési frekvencia [GHz]	Jellemző sebesség [Mbps]	Maximális sebesség [Mbps]	Hatótávolság beltéren [m]	Hatótávolság kültéren [m]
802.11	1997	2,4	0,9	2	~20	~100
802.11a	1999	5	23	54	~35	~120
802.11b	1999	2,4	4,3	11	~38	~140
802.11g	2003	2,4	19	54	~38	~140
802.11n	2010[16]	2,4/5	74	248	~70	~250
802.11y	2008	3,7s	23	54	~50	~5000

A vezetéknélküli hálózat node-jai kétféle módon kommunikálhatnak egymással:

- Ad-hoc mód: Két node közvetlenül kapcsolódik egymáshoz.
- Infrastructure mód: Több node kapcsolódik egy Access Point-hoz (AP).

Bármelyik módban is működik a vezetéknélküli hálózat, mindig kell, hogy legyen egy neve. Ezt Service Set Identifier-nek, azaz SSID-nek nevezzük. (Maximális mérete 4 bájt.)

Gondolom, nem ettől koppan az állkapcsunk a padlón: a vezetéknélküli hálózatok broadcast alapúak. Ebből persze az is következik, hogy nagyon könnyen lehallgathatók. Több kisérlet is történt a hekkerek[17] elhessegetésére:

- Wired Equivalent Privacy, WEP: fúj.
- Wi-Fi Protected Access, WPA: so-so.
- Wi-Fi Protected Access 2, WPA2: yes.

WPA, WPA2:
http://hu.wikipedia.org/wiki/WPA

Általában a wifi:
http://en.wikipedia.org/wiki/802.11
http://hu.wikipedia.org/wiki/Wi-Fi

WIFI keret:
http://wifi.cs.st-andrews.ac.uk/wififrame.html

Nézzük akkor a kereteket.

[16] Várhatóan. Amikor ezeket a sorokat írom - 2009 márciusában - a Draft8 az aktuális verzió.
[17] Definíció: a hekker az a gonosz hacker.

3.12. ÁBRA IEEE 802.11 FRAME FORMÁTUM

FRAME CONTROL: Kétbájtos mező, meglehetősen sok információtartalommal.

- Protocol version (2 bit)
- Type (2 bit): Lehetőségek: management frame (00), control frame (01) és data frame (10).
- Subtype (4 bit): A fenti tipuson belül az altípusok.
- To DS (1 bit): Ha magas, akkor az azt jelenti, hogy ez egy olyan csomag, melyet egy wireless Access Point küldött valamilyen vezetékes hálózat felé.
- From DS (1 bit): Na, vajon?
- More Fragments (1 bit): Ha magas, akkor azt jelzi, hogy a keret egy nagyobb, fregmentált keret töredéke. Ha alacsony, akkor azt, hogy a keret vagy nem fregmentált, vagy ez az utolsó töredék.
- Retry (1 bit): Ha magas, akkor ez egy újraküldött keret.
- Power Management (1 bit): Ha magas, akkor a feladó node energiatakarékos üzemmódban működik.
- More Data (1 bit): Ha magas, akkor a feladónak még van mondanivalója. (Azaz nem üres a kimenő puffere.)
- WEP (1 bit): Ha magas, akkor a payload titkosított.
- Order (1 bit): Ha magas, akkor a kereteket sorrendben kell feldolgozni.

DURATION/ID: Kétbájtos mező, gyakorlatilag egy időérték, mikroszekundumban. Mennyi időre van szükség a keret és az ACK továbbítására. (Ez utóbbi a MAC alréteg része.)
ADDRESS1: Hat bájt. Vagy a célállomás MAC címe (amennyiben a cél egy wireless node), vagy egy SSID (amennyiben a cél egy Access Point).

ADDRESS2: Hat bájt. Amennyiben a feladó egy wireless node, akkor MAC cím. Amennyiben egy Access Point, akkor SSID.

ADDRESS3: Hat bájt. Bármilyen MAC cím vagy az SSID található benne. A fogadó állomás használja szűrési célra.

SEQUENCE CONTROL: Két bájt. A töredezett keretek kezelésére szolgál.

- Fragment Number (4 bit): A töredék sorszáma.
- Sequence Number (12 bit): Tulajdonképpen ez is a keret sorszáma, de 4095 után újraindul nulláról.

ADDRESS 4: Hat bájt. Opcionális mező, az eredeti feladó MAC címét tartalmazza. Tipikusan akkor van értéke, ha a To DS, From DS mezők nem üresek.

FRAME CHECK SEQUENCE: CRC.

IEEE 802.2 LCC HEADER: Ezt már ismerjük.

PAYLOAD: Alapvetően három tipusú hasznos teherről beszélhetünk:

- Control Frame: Request To Send (RTS), Clear To Send (CTS) és Acknowledgement (ACK).
- Management Frame: Authentication, Association, Beacon, Probe és kismillió egyéb.
- IP Datagram: A klasszikus IP csomag a felettünk lévő rétegből.

A payload maximális értéke 2312 bájt lehet, melyet a titkosítás még tovább csökkenthet, tekintve, hogy bármelyik módszert is használjuk, be kell áldoznunk pár bájtot a payloadból.

Végül a SNAP változat. Nos, egyfelől ugyanúgy képződik, mint az összes többi hálózati technológiánál, ezért megint nem rajzolok. Másfelől pedig tudni kell, hogy a használata kötelező. Már amikor.
Mikor is? Csak az IP datagramoknál. Emlékezzünk vissza, mikor sznapoltunk? Amikor az IP datagramra utaló paramétert máshol, egész konkrétan az EtherType mezőn keresztül szerettük volna bemutatni. Logikusan: ha nincs IP datagram, nyilván nincs SNAP sem.

3.1.5 ALRÉTEG PROTOKOLLOK: LLC ÉS MAC

Az a bizonyos nem létező Data Link Layer két alréteggel is bír. Ezek egész konkrétan: Logical Link Control (LLC) és Media Access Control (MAC).

A 802.2 LLC protokollt már jól ismerjük. A korábbi fejezetek majd mindegyikében felbukkan, mint egy kapszula. Tulajdonképpen a payload-ra tesz rá egy headert. (A DSAP, SSAP, Control mezők az IEEE 802.3 keret részletezésénél lettek kibontva.)

LLC:
http://www.erg.abdn.ac.uk/users/gorry/course/lan-pages/llc.html

De mi a szösz ez a MAC alréteg? Az azonos hangzáson kívül van-e más köze is a MAC címhez?

Van:a MAC alréteg címzési mechanizmusát hívják MAC címnek.

Irkáltam itt mindenfélét a baráti társaság udvarias beszélgetéséről, na meg a modellvasútról. Ezeket összefoglalóan Channel Access Control Mechanism néven illetik... és a MAC alréteg valósítja meg, egész konrétan az ún. Multiple Access Protocol (MAP) segítségével. Mint a neve is mutatja, ez a protokoll hivatott biztosítani azt, hogy több eszköz is képes legyen kommunikálni ugyanazon a médián. Maga a MAP lehet már akár az ütközésfigyelés, akár a token köröztetés.

Media Access Control:
http://en.wikipedia.org/wiki/Media_Access_Control

MAC Address:
http://en.wikipedia.org/wiki/MAC_address

3.2 ADDRESS RESOLUTION PROTOCOL, ARP

RFC 826

Mint a neve is mutatja, itt bizony címfeloldásról lesz szó. Ez egy olyan protokoll, mely az eggyel fentebb lévő réteg jellemző címéhez - IP cím az Internet rétegben - keresi meg a Data Link rétegbeli címet, mely, mint nemrég írtam, a MAC address.

Kiskáté jön.

Mely hálózati technológiák használják?
Az eddig felsoroltak például nagyon. A PPP-n alapuló WAN technikák meg nem igazán - és akkor is fordítva.

Használható IP-MAC feloldáson kívül másra is?
Igen. Csak nem használják.

Milyen hatókörben használható?
Csak a legközelebbi hop-ig. Tehát olyan IP cím MAC addressét nem tudjuk megkérdezni tőle, mely nem a saját subnetünkön van. Nyilván nem is akarjuk.

A Windows Server 2008-ban lévő ARP implementáció különbözik-e a Windows Server 2003-ban lévőtől?
Bizony, igen. De erre később még visszatérünk. Stay tuned.

Magát az ARP kommunikációt az RFC 826 írja le. Ez alapján kétféle üzenettípust különböztethetünk meg:

- ARP REQUEST: Egy node szeretné megtudni, hogy az általa megcélzott IP címhez milyen MAC address tartozik. Küld egy MAC szintű broadcast üzenetet, benne a kért IP címmel.
- ARP REPLY: Az a node, aki magára vette a kérést - azaz az IP címe megegyezett az ARP request broadcast üzenetben lévő IP címmel - küld vissza egy ARP reply unicast üzenetet.

Rögtön láthatjuk, miért is nem jó más subnet-en lévő node-ok esetén: broadcast. Az nem szokott átmenni a routereken. HUB, Bridge, L2 Switch - mind oké.
IP router nem.

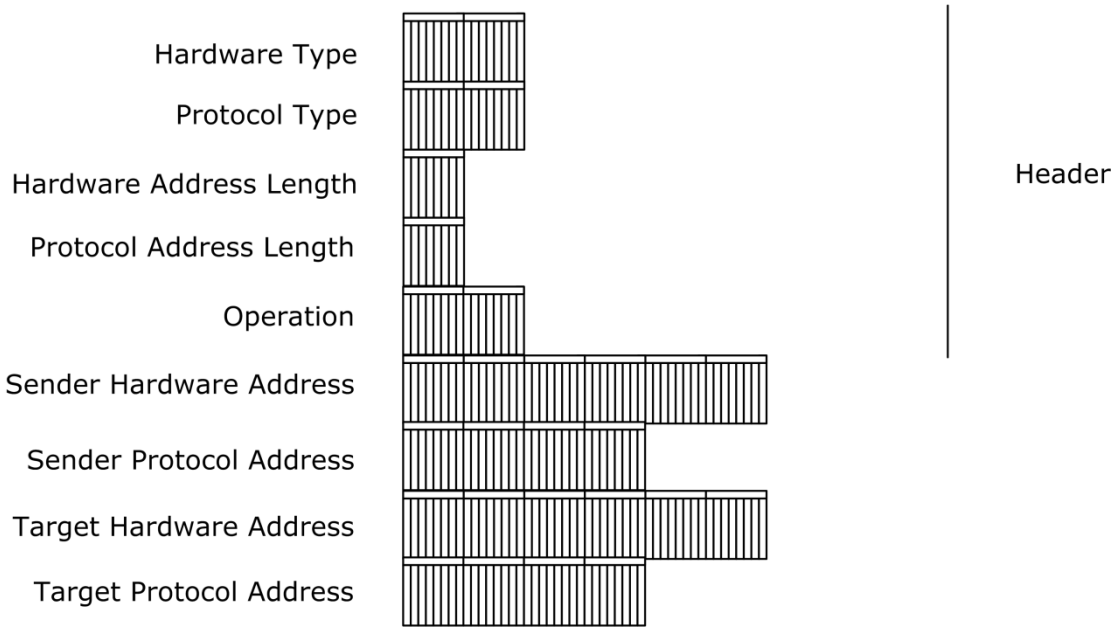
No.	Time	Source	Destination	Protocol	Info
1	0.000000	AsustekC_ab:37:2e	Broadcast	ARP	Who has 192.168.1.111? Tell 192.168.1.100
2	0.000272	LgElectr_04:d6:0c	AsustekC_ab:37:2e	ARP	192.168.1.111 is at 00:e0:91:04:d6:0c
3	0.000291	192.168.1.100	192.168.1.111	ICMP	Echo (ping) request
4	4.880722	192.168.1.100	192.168.1.111	ICMP	Echo (ping) request
5	9.881020	192.168.1.100	192.168.1.111	ICMP	Echo (ping) request

3.13. ÁBRA ARP KÉRDEZZ-FELELEK

Az ábrán egy pingelés folyamata látszik. Az első csomag egy ARP request, mely az Austek hálózati kártyát tartalmazó számítógépről (192.168.1.100) jön, és a 192.168.1.111 IP címhez tartozó MAC addresst kérdezi. A második csomag egy ARP reply az LG hálózati kártyát tartalmazó géptől, benne a kért IP címhez tartozó MAC address. Utána már mehet is a ping.

Nézzük meg az ARP keretek felépítését.

De először pozícionáljuk helyére az ARP keretet. Hol található, azaz kinek is a payload-ja? Lapozzunk vissza a megfelelő fejezethez (*3.1.1.1 Ethernet II*), láthatjuk, hogy például az Ethernet II keret belseje lehet IP datagram (0x0800) vagy ARP keret (0x0806). És persze nem csak az Ethernet II keret jöhet szóba, hanem minden egyéb eddig tárgyalt keret is. (Sőt, olyanok is, melyek eddig nem kerültek említésre: ATM, HIPPI, stb.) De a lényeg akkor is látható: az ARP nem része az IP-nek, hanem azzal párhuzamosan tevékenykedik - azaz a Network Interface rétegben a keretek tartalma lehet az eggyel fentebbről jövő IP datagramm, vagy a rétegen belül mozgó ARP csomag.



3.14. ÁBRA ARP KERET

HARDWARE TYPE: Két bájt, arra vonatkozik, hogy a Network Interface rétegben milyen hálózati technikát használunk. (Aranyos, ugye: az Ethernet frame fejléce utal arra, hogy ARP csomag van benne, az ARP csomag fejléce meg arra, hogy Ethernet keretben utazik.)
Az Ethernet kódja az egyes (0x00-01), a többi kódot az IANA weblapjáról lehet lelopni.

Hardware Type kódok:
http://www.iana.org/assignments/arp-parameters/

PROTOCOL TYPE: Két bájt, arra utal, hogy az ARP milyen protokoll számára biztosítja a névfeloldást. Elméletileg ugyanazok lehetnek az értékei, mint az EtherType mezőnek - gyakorlatilag ha nem 0x08-00 (IP), akkor a node eldobja a csomagot.

HARDWARE ADDRESS LENGTH: A hardware address jelen esetben a MAC address. Mint tudjuk, ennek az értéke gyárilag beégetve 6 bájt, de vannak olyan esetek, amikor 2 bájt is lehet. (Pl. Frame Relay.)

PROTOCOL ADDRESS LENGTH: Ez pedig az IP cím hossza. Értelemszerűen az értéke fixen 4[18].

OPERATION: Két bájt, az ARP keret tipusát adja meg. Az ARP request tipusa 0x00-01, az ARP reply tipusa pedig 0x00-02. A többi lehetséges értéket a korábban is említett weblapon találhatjuk meg.

SENDER HARDWARE ADDRESS, SHA: A feladó MAC címe.

SENDER PROTOCOL ADDRESS, SPA: A feladó IP címe.

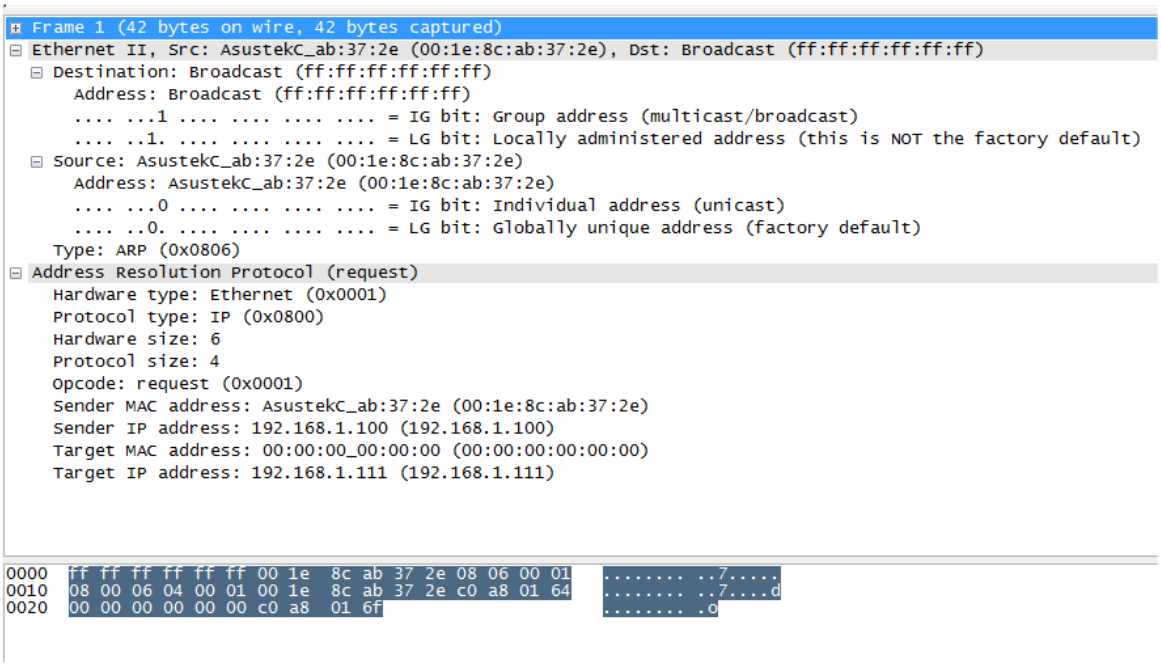
TARGET HARDWARE ADDRESS, THA: A címzett MAC címe.

TARGET PROTOCOL ADDRESS, TPA: A címzett IP címe.

Na, kérem. Próbáljuk meg elképzelni, hogyan is néz ki egy ARP request. Addig oké, hogy az OPERATION kódja 0x-00-02. De az utolsó négy mezőből vajon melyiknek nem lesz értéke? Ugye, tudjuk a feladó MAC address-ét, IP címét, tudjuk a címzett IP címét... logikus, hogy aTHA mezőnek kell üresnek lennie.

Ellenőrizzük le.

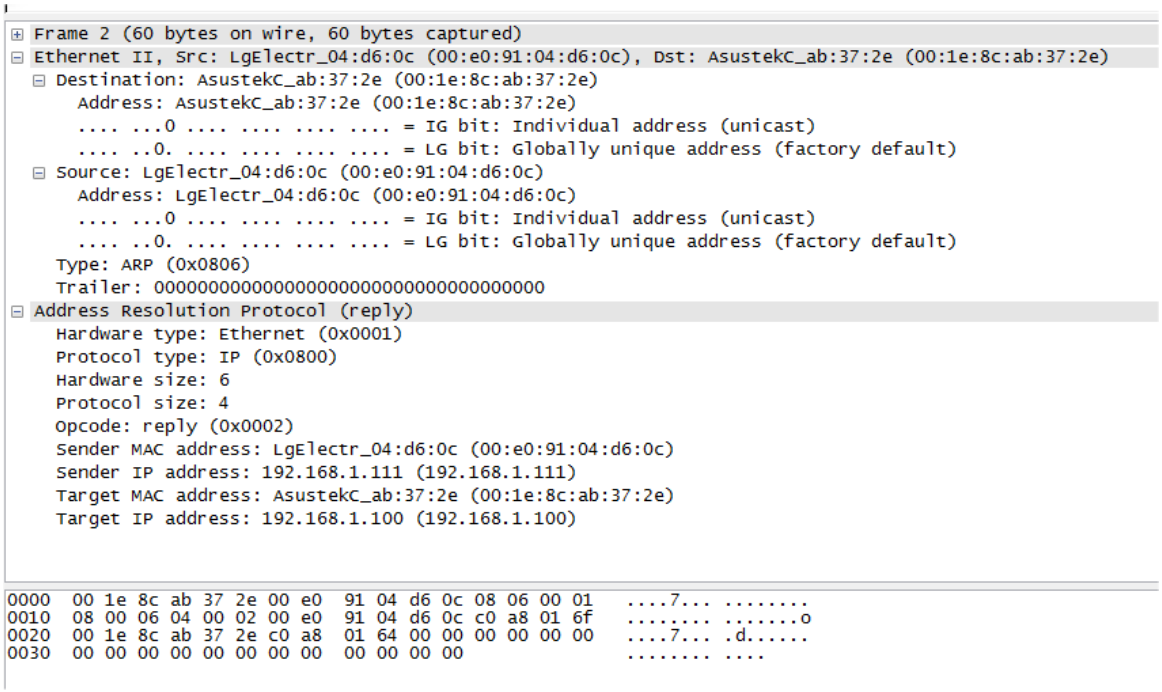
[18] IPv6 esetén vajon mennyi? Vigyázz, beugratós kérdés: az IPv6 nem használ ARP-ot.



3.15. ÁBRA ARP REQUEST

Itt látható a korábbi ping capture kibontva. Miket is látunk benne:

- Az Ethernet frame fejlécében ott van a 0x0806 kód, tehát a payload egy ARP keret.
- Az ARP keret szerint a hardvertipus Ethernet, a protokolltípus IP és az ARP keret tipusa request.
- A méretekkel nem kell foglalkozni, a 6/4 páros gyakorlatilag konstans.
- Végül, ha jobban megnézzük, látható, hogy a THA mező értéke tényleg csupa nulla.



3.16. ÁBRA ARP REPLY

Ez pedig az ARP reply válasz. Nem akarok már túl részletesen belemenni, az előző kép alapján te is értelmezhetsz rajta mindent. A lényeg, hogy az SHA érték tartalmazza a helyes választ.

Egy apróságra hívnám még fel a figyelmet. Az Ethernet frame végén szerepel egy Trailer mező, csupa nullával. Ez a Padding. Mint írtam, az Ethernet keretnek mindig van minimális mérete. Amennyiben a küldendő csomag ezt nem éri el, nullákat löknek a végére.

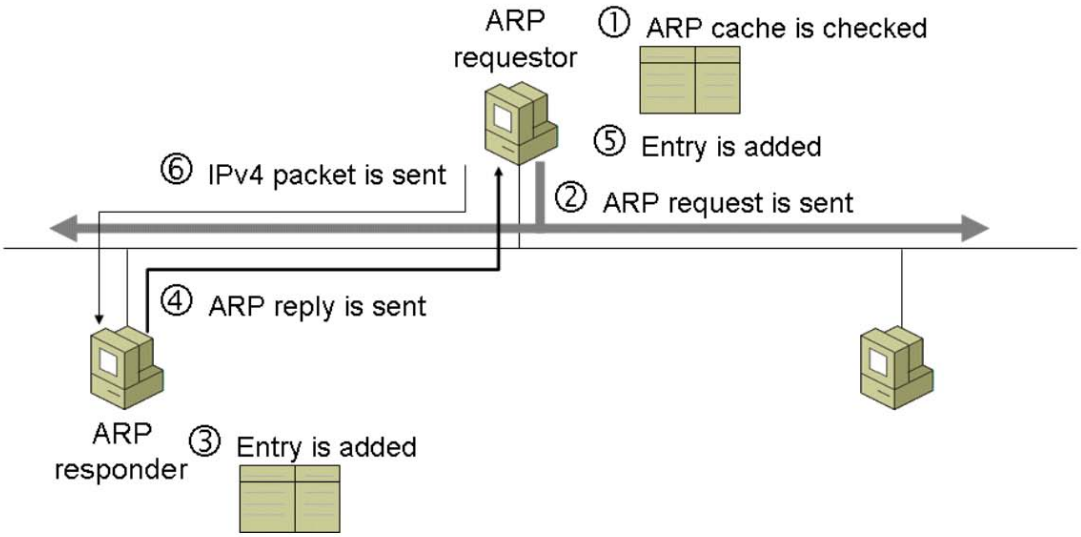
Address Resolution Protocol:
http://en.wikipedia.org/wiki/Address_Resolution_Protocol

Ennyi volt a bemelegítés. Vizsgáljuk most meg, hogyan is történik konkrétan az ARP névfeloldás megvalósítása. Hiszen, mint tudjuk, az ördög a részletekben.

3.2.1 ARP NÉVFELOLDÁS

Magát a névfeloldást már láthattuk az előző oldalakon. Broadcast ARP request üres THA értékkel, ARP reply helyes SHA értékkel. A világ egyszerű.
Azért a kérdező még megejt egy röpke ellenőrzést, megnézi, hogy a reply-ban visszakapott SPA megegyezik-e a request-ben kiküldött TPA értékkel. Ha igen, akkor beteszi a címpárt az ún Neighbor cache-be[19].

Természetesen a válaszoló is berakja a saját cache-ébe a kérdező adatait. Hadd gyűljenek a fontos információk.
A folyamat teljes részleteséggel az alábbi ábrán látható.



3.17. ÁBRA A HAGYOMÁNYOS ARP KOMMUNIKÁCIÓ

[19] A Neighbor Cache például egy újdonság. A korábbi Windows verzióknál a címek ún. ARP cache-ben voltak. A váltás nem csak azt jelenti, hogy a Windows 2008-ban átvették az IPv6 terminológiát - átvették azokat a technikai változtatásokat is, melyekkel javítani lehetett valamit az IPv4-es megoldásokon.

Gyors ismétlés miért is jó egy ilyen cache? Arra, hogy ne kelljen állandóan broadcast üzenetekkel tömni a csövet. És miért is rossz? Mert könnyen be lehet vele csapni egy gépet.

A becsapásnak kétféle módja létezik:

- Alattomos informatikusok kicserélik az egyik gépben a hálózati kártyát. Vagy átírják a MAC címét. A cache-ben rossz bejegyzés marad, a csomagok rossz címre lesznek elküldve.
- ARP Poisoning. Rosszfiúk küldenek egy preparált ARP request csomagot a kiszemelt áldozatnak. Az onnantól azt fogja hinni, hogy a cache-ben tárolt IP címnél megváltozott a MAC address (mert az informatikusok vagy kicserélték benne a kártyát, vagy átírták a MAC címét).

Man in the Middle támadás:
http://tudastar.netacademia.net/default.aspx?upid=2836

Láthatod, a helyzet mégsem olyan egyszerű. Ha automatikusan elfogadom a változásokat, akkor törhető a gép. Ha nem, akkor egy változás esetén a fekete lyukba küldöm a csomagokat.
Megoldás?
Egy apró módosítás. A korábbi Windows-ok már akkor módosították a cache-ben lévő bejegyzésükben a feladó címét, amikor megkapták az első, új MAC címet tartalmazó ARP request csomagot (*3.17. ábra A hagyományos ARP kommunikáció*). A Windows 2008 ellenben ekkor még nem: visszakérdez (broadcast) és megvárja a választ[20].

Ha már az eltérő viselkedésekről van szó. Szép nagy pofonba lehet beleszaladni, ha Windows Server 2008-ból rakunk össze multicast NLBS rendszert - és úgy képzeljük, hogy minden ugyanúgy fog működni, mint Windows Server 2003 esetén.
Nos, nem.
Meglepődve fogjuk tapasztalni, hogy a clusterünket más subnetből nem fogják elérni. Pedig ránézésre minden rendben, subneten belül működik is hibátlanul.

A jelenség mögött ARP névfeloldási gubanc rejtőzik.

Windows 2008 NLBS esetén amikor subneten kívülre szeretne csomagot küldeni a cluster, akkor a virtuális kártyájáról küld ARP request-et a router felé. Ekkor az SPA unicast IP cím, míg az SHA multicast MAC address. Nos, ez a kombináció általában

[20] Őszinte leszek: ebben nem vagyok 100%-ig biztos. Az általam használt források teljesen ködösen fogalmaztak. Órákig túrtam a netet, de ott sem találtam egyértelmű infót. Addig oké, hogy a források szerint a Windows Server 2008 még vár egy kört - de szerintem úgy lenne logikus, ha ezt a kört _ő_ kezdeményezné: egyébként a gonosz fiú küldene még egy request-et, oszt jól van.

kiveri a biztosítékot a routereknél. Lenyelik a request-et, nem küldenek vissza választ - enélkül viszont a cluster nem tudja elküldeni a csomagot.
Korábban a Windows 2003 NLBS ilyen esetekben csalt egy kicsit: nem a virtuális kártya IP cím / MAC address párosát küldte el, hanem annak a valóságos kártyának az értékeit, amelyen a virtuális kártya ki lett alakítva. Ezzel persze működött a névfeloldás.

Nos innentől kezdve két dolgot tehetünk. Vagy megvárjuk, amíg a Microsoft kitalál valamit - vagy a kezünkbe vesszük a dolgot. Ez utóbbi abból áll, hogy az NLBS cluster összes node-ján felvesszük a Neighbor Cache-ba a router IP cím - MAC address párosát.

Két lehetőségünk van:

```
Arp –s <IP address> <Mac Address>
netsh interface ipv4 set neighbors "hálózati kártya neve" "<ip-Address>" "<Mac-Address>"
```

A másodiknak megvan az az előnye, hogy kikapcsolás után is megmarad.

Unable to connect to Windows Server 2008 NLB Virtual IP Address from hosts in different subnets when NLB is in Multicast Mode:
http://tinyurl.com/chd56l

3.2.2 DUPLIKÁLT CÍM MEGHATÁROZÁS

Ezzel messze nincs még vége a gubancoknak. Mi van például akkor, ha a gratuitous ARP-ra bármilyen válasz is jön?

Egyike az élet nagy kérdéseinek. Az meg különösen, hogy mi a fene is egyáltalán ez a gratuitous ARP? Ezért nem kell fizetni? A többiért meg igen? Vagy inkább csak felesleges?

Nem egészen. A gratuitous ARP egy olyan ARP request, melyet saját magunknak küldünk ki - azaz a TPA és az SPA megegyezik. Aztán várjuk, hogy jön-e rá válasz. Ha igen, akkor baj van: több példányban létezünk. Jogos a kérdés, miért is tennénk ilyesmit? Hát, például be szeretnénk illeszkedni egy hálózatba és kíváncsiak vagyunk, hogy mennyire egyedi az IP címünk a subneten.

Induljunk ki abból, hogy megtörtént a baj. Kiderült, hogy kettő van belőlünk. Mi az első dolgunk? Természetesen az, hogy tisztázzuk, kit hogy hívnak. Mi magunk - a párbajkódex szerint - a támadó felek vagyunk, a másik pedig értelemszerűen a védekező fél lesz[21]. (Angolban offending node és defending node.)

A következő lépés az lesz, hogy megnézzük, mit is csináltunk eddig. Kiküldtük a gratuitous ARP request-et. A védekező node felvette, hiszen az neki szólt, majd ugyanazzal a mozdulattal be is rakta a saját ARP cache-ébe, miszerint ehhez az IP címhez a mi MAC címünk tartozik.

Hoppá. Ezzel most jól kibabráltunk. Hiszen csak kérdezni akartunk.

Nosza, gyorsan tegyük rendbe a dolgokat. Rakjunk össze egy másik ARP csomagot, ahol az SHA mezőbe a védekező node MAC addressét írjuk, az IP címek meg maradnak ugyanazok, mint a gratuitous ARP csomagban. A védekező node - és persze mindenki más is a subneten - felkapja ezt a csomagot is, majd korrigál a cache-ben... azaz visszaáll az eredeti állapot.
Huhh.

[21] Akit esetleg zavar, hogy a másik fél a védekező, gondoljon arra, hogy arról van szó, hogy mi lépünk be új fiúként abba hálózatba, ahol a többieknek már régóta megvan a kényelmes, jól bejáratott IP címe.

Jó ez nekünk? Nem igazán. A Windows 2008/Vista vonalon már nem is így működik.

- Egyfelől a támadóként kiadott gratuitous ARP már nem tartalmazza a feladó IP címét - azaz az SPA üres. Mit fog csinálni a védekező fél? Felkapja, rakná be az IP címet a cache-be... de hát az üres. Nem is rakja be.
- Ha mi vagyunk a védekezők és kapunk egy hagyományos gratuitous ARP kérést... nos, már vagyunk annyira okosak, hogy fel tudjuk ismerni - és nem rakjuk be az ARP cache-be. Csak azért sem. Így a rögtön utána érkező Bocsesz gratuitous ARP request-tel sem kell foglalkoznunk.

Kiveséztük már a témát? Ádehogy.

Mi is történik velünk, mint támadókkal, az után, hogy visszakaptuk azt a bizonyos duplikációt jelző ARP reply üzenetet? Attól függ. Ha manuálisan konfigurált IP címünk van, akkor az átvált APIPA IP címre, persze egy eventlog bejegyzés és egy figyelmeztető üzenet kíséretében.
Amennyiben DHCP szervertől kapjuk az IP címet - jelen esetben a rossz IP címet - akkor egész egyszerűen küldünk a DHCP szervernek egy DHCPDECLINE üzenetet. Nem kell az IP címed, kűgyél egy másikat. Ha a DHCP szerver egy Windows Server 2008 alapokon futó szerver, akkor meg is jegyzi ezt az IP címet, nehogy később ismét kiküldje. Majd küld egy másikat.

Ezekben az esetekben rendeződött a helyzet. De mi van akkor, ha bekapcsoltak egy számítógépet, mely nincs rajta a hálózaton és van egy manuálisan konfigurált IP címe? Aztán, miután felállt rajta a TCP/IP, rádugják a hálózatra, ahol már van egy másik gép, ugyanezzel az IP címmel. Van ilyenkor gratuitous ARP? Szomorú, de nincs. A két node ugyanazzal az IP címmel fog dolgozni. Természetesen mindkettőnek az eventlogja tele lesz hibaüzenettel, hiszen az ARP keretek alapján észlelni fogják egymás jelenlétét - de a javításra már képtelenek.

3.2.3 FEKETE LYUK DETEKTÁLÁSA

Mi is az a fekete lyuk?

Eddig beszéltünk arról az esetről, amikor egy node IP címe már létezett azon a hálózaton, ahová be akart nyomulni. Ezt le lehetett rendezni.

De mi van akkor, ha egy node-ot lekapcsoltunk? A feladó gép cache-ében még ott van a bejegyzés, adott esetben küldi is neki a csomagokat. Ciki, de ekkor visszajelzés sem jön arról, hogy a csomagok elnyelődtek - egyszerűen eltűntek a fekete lyukban. Amíg a cache ki nem ürült, addig bizony nem is történik változás. (Oké, a TCP/IP magasabb rétegeiben lévő protokollok, szolgáltatások észlelték, hogy nem jött létre a kapcsolat. De ARP szinten a névfeloldás látszólag működött, hiszen megvolt a MAC address, össze lehetett állítani a csomagokat és el lehetett küldeni azokat a nagy semmibe.)

A Windows 2008/Vista oprendszerekben már létezik egy ún. Neighbor Unreachability Detection metódus.

Ehhez viszont meg kell ismerkednünk egy új szereplővel. Úgy hívják, hogy unicast ARP request. Az elv egyszerű: ha meg akarunk győződni arról, hogy egy tetszőleges node él-e még, akkor direktben küldünk neki egy ARP request-et és várjuk a hozzá tartozó ARP reply-t. Vigyázat, a vizsgálati módszer nem megfordítható! Tehát az ARP reply megérkezése után az érdeklődő biztos lehet benne, hogy a másik node működik... de aki a reply-t küldte vissza, az nem lehet biztos abban, hogy az tényleg meg is érkezett, azaz a feladó elérhető a számára.

Ez az egyik módszer. Melyik lehet a másik?

Már céloztam rá. Ott vannak a fentebb lévő protokollok. Például a jó öreg TCP. Az ugye állandóan visszajelzéseket kér. Mi lenne, ha figyelnénk, hogy ezek tényleg megérkeznek-e a túloldalról?

Oké, ezek az eszközök. Hogyan épül fel ebből egy algoritmus?

Úgy, hogy a cache bejegyzések mellé állapotot jelző információk kerülnek. Ezek jelzik, hogy mi is éppen most a helyzet a bejegyzett értékkel.

A lehetőségek:

- INCOMPLETE: Éppen folyik az ellenőrzés egy frissiben létrejött bejegyzésnél.
- REACHABLE: Megjött az ARP reply a node-tól, minden oké. Fontos elem, hogy ez a bejegyzés nem az örökkévalóságig tart. Egy bizonyos időtartamon belül véletlenszerű ideig él a bejegyzés, utána meg kell újítani - feltéve, hogy a távoli node nem adja folyamatosan jelét a létezésének. Ez gyakorlatilag azt jelenti, hogy minden sikeres TCP visszajelzés után a számláló újraindul.
- STALE: Letelt a Reachable Time. Amíg újra meg nem bizonyosodunk ismét a távoli node létezéséről, addig ebben az állapotban marad a bejegyzés.
- DELAY: Amikor a bejegyzés már Stale állapotba került, de még nem küldünk ki neki unicast ARP request csomagot, mert hátha a TCP még aktivizálja magát. (Ez az elmélet. A gyakorlatban a Windows Server 2008/Vista kihagyja ezt a lépést.)
- PROBE: Éppen ellenőrizzük egy korábban Stale/Delay állapotba került bejegyzés valódiságát. Nyilván unicast ARP request csomaggal.

Ha már itt járunk, említsük meg, milyen lehetőségeink vannak manuálisan befolyásolni a cache tartalmát.

Például törölhetjük:

- `arp -d *`
- `netsh interface ipv4 delete neighbors`

Kilistázhatjuk:

- `arp -a`
- `netsh interface ipv4 show neighbors`

Hozzáadhatunk manuálisan értékeket:

- Lásd az NLBS clusteres linket.

3.2.4 INVERSE ARP, REVERSE ARP, PROXY ARP

Definíciókra fogok szorítkozni, minimális körítéssel. Pusztán csak azért, hogy ne nézzünk tágranyílt szemekkel, ha a címben szereplő kifejezésekkel összefutunk valahol.

INVERSE ARP: Szigorúan a Non-Broadcast Multiple Access (NBMA) rendszereknél használják. Ezek WAN technológiák, olyasmik, mint pl. a Frame Relay, ATM vagy az X.25. Itt általában a hardver azonosító az ismert, az IP cím pedig nem. Ebből kifolyólag az ARP alréteg módszereit, az ARP keretszerkezetet átvették ugyan, de pont fordítva használják fel: hardver azonosítóból keresnek IP címet.

REVERSE ARP: Nagyon hasonló az Inverse ARP-hoz, legalábbis olyan szempontból, hogy hardverazonosítóhoz keres IP címet. Csakhogy ezt a metódust már nem csak a fenti WAN technológiák használták, hanem gyakorlatilag mindegyik más is, méghozzá olyan célból, hogy IP címet igényelhessenek maguknak. Aztán jött a BOOTP és elsöpörte a francba az egészet... majd jött a DHCP és elsöpörte a francba a BOOTP-t[22]. Szóval ennyire aktuális dologról van szó.

PROXY ARP: Gondolom, már vártuk. Megszoktuk már, hogy a gonosz routerek mindenhol bekavarnak, emiatt kell DHCP proxy, meg WINS proxy... nyilván lesz valamilyen proxy az ARP-hoz is.
Milyennek képzeljük? Mondjuk, van egy router, a két lábán két subnet. Aztán az egyiken elindul egy ARP Request egy olyan node felé, mely a másik lábán lóg, erre a Proxy ARP felkapja a kérést és átdobja a túloldalra.
Egyszerű, nem?
Lehet... de ez a koncepció ezer sebből vérzik.
Először is, nem nagyon mászkálhat a dróton olyan ARP Request csomag, melyben a feladó IP címe és a címzett IP címe más alhálózatokhoz tartozik. Ilyenkor ugyanis a node nem a másik node MAC címére szokott kíváncsi lenni, hanem sokkal inkább a vele egy alhálózaton lévő default gateway MAC címére.
De még ha van is ilyen csomag, a router akkor sem dobja át a túloldalra, hanem hazudik, azaz amikor az egyik oldalon kérdeznek tőle egy MAC címet, akkor a sajátját adja vissza, mondván, hogy küldd csak el nekem a csomagot, én majd tudom, mit kell kezdenem vele. Még konkrétabban: Data Link Layer szinten a Proxy ARP-ként viselkedő router egyszerűen csak hazudik, majd ha a hazugság révén hozzákerül a csomag, akkor már L3 szinten routolja.

Jó kérdés, hogy most akkor tekerhet-e ilyen speciális csomag a dróton?

[22] Ez egy hatásos bon-mot, de nem teljesen igaz. PXE boot esetén még használják a BOOTP-t.

Egy kedves olvasó hívta fel rá a figyelmemet, hogy bizony igen.

> Sometimes called promiscuous ARP and described in RFCs 925 and 1027, proxy ARP is a method by which routers may make themselves available to hosts. For example, a host `192.168.12.5/24` needs to send a packet to `192.168.20.101/24`, but it is not configured with default gateway information and therefore does not know how to reach a router. It may issue an ARP Request for 192.168.20.101; the local router, receiving the request and knowing how to reach network 192.168.20.0, will issue an ARP Reply with its own data link identifier in the hardware address field. In effect, the router has tricked the local host into thinking that the router's interface is the interface of 192.168.20.101. All packets destined for that address will be sent to the router.
> *Jeff Doyle :Routing TCP/IP Volume I.*

Azaz ha a node-nak nincs beállítva semmilyen default gateway értéke, akkor végső elkeseredésében kiküldhet egy ARP request-et, melyet már kezelni tud a Proxy ARP.

Jól hangzik... de engem teljesen megzavart. Én ugyanis más könyvből tanultam, és abban azt mondja Joseph Davies, hogy Windows rendszerekben két helyen találkozhatunk Proxy ARP funkcióval:

- Windows Server 2008/Vista operációs rendszerekben a Network Connection folderben a Network Bridge képesség pont ezen alapul.
- Windows Server 2008-ban amikor a RAS oszt a subneten belülről egy elkülönített IP zónából címeket, akkor gyakorlatilag ugyanilyen bridzselés játszódik le.

Ami egyértelműen azt jelzi, hogy a Proxy ARP csak akkor működik, ha a proxy egység két oldalán ugyanaz a subnet található.

Nézzük meg a Wikipedia-t.

> For example, suppose a host, say A, wants to contact another host B, where B is on a different subnet/broadcast domain than A. For this, host A will send an ARP request with a Destination IP address of B in its ARP packet. The multi-homed router which is connected to both the subnets, responds to host A's request with its MAC address instead of host B's actual MAC address, thus proxying for host B. In the due course of time, when host A sends a packet to the router which is actually destined to host B, the router just forwards the packet to host B. The communication between host A and B is totally unaware of the router proxying for each other.
> http://en.wikipedia.org/wiki/Proxy_arp

Ez bizony egyértelműen az első verzió. Viszont wikipedia.

De mit ír erről a TCP/IP Guide?

> In contrast to the normal situation, in some networks there might be two physical network segments connected by a router that are in the same IP network or subnetwork. In other words, device A and device B might be on different networks at the data link layer level, but on the same IP network or subnet.
> http://www.tcpipguide.com/free/t_ProxyARP.htm

Akkor most melyik az igaz??

Jó hírem van: mindkettő. A legelső idézet ugyanis Cisco routerekre vonatkozik és a magam részéről biztos vagyok benne, hogy ezek tényleg tudják azt, amit állítanak róluk. Azaz a Proxy ARP komponensük tényleg le tudja kezelni azt a szituációt, amikor az ARP Request-ben más alhálózatba tartozik a két IP cím.

Másfelől viszont Windows rendszerekben nem fordulhat elő ilyen ARP Request.
Nem állítom, hogy minden Windows verziót leteszteltem, de a saját gépemet igen. Leszedtem a hálózati kártyáról minden komponenst, csak a TCP/IPv4 maradt rajta. Abból meg töröltem a default gateway értékét. Újraindítottam a gépet, kiürítettem az ARP cache-t, elindítottam a Wireshark-ot és lelkesen nekiálltam pingelni, telnetelni. Semmi. Habár a parancssorban kaptam rendesen a hibaüzeneteket, de a sniffer nem mutatott semmit.

Azaz lehet, hogy a routerem tényleg tudta volna a Proxy ARP-ot, de a Windows észleli, hogy nem frankó a hálózat beállítása és nem is enged ki a hálózatra semmilyen idióta csomagot.

Mindenféle ARP-ok:
http://www.javvin.com/protocolARP.html
http://www.cisco.com/en/US/tech/tk648/tk361/technologies_tech_note09186a0080094adb.shtml

3.3 WAN TECHNOLÓGIÁK

3.3.1 POINT TO POINT PROTOCOL, PPP

Előre szólok, hogy ez egy kimerítően hosszú fejezet lesz. Már most tegyél fel főni egy kávét. Literes fazékban.

Kezdjük ott, hogy habár protokollnak hívják, de a PPP nem az. Sokkal inkább egy protokollcsalád. Meglehetősen sok alprotokollt tartalmaz, ezek jó részével meg is fogunk ismerkedni.

Mire is jó?

Ha a nevéből indulunk ki, akkor talán nem lesz olyan nagy a meglepetés: pont és pont közötti kapcsolat menedzseléséhez szükséges protokollkupacról lesz szó.

Mivel csak két pontról van szó, egy csomó eddig tanult dologra nincs szükségünk. Nyugodtan kidobhatjuk, a unicast/multicast/broadcast fogalmakat. Ember, csak két node van. Igazából se baráti beszélgetésről, se kisvonatról sem beszélhetünk. Vajon fogunk találkozni Data Link Layer szintű címzéssel, azaz MAC addressel?
Nem.

Csak.
Két.
Pont.
Van.

A PPP elődje - a SLIP - ennek megfelelően egyszerű is volt, mint a faék. Csak TCP/IP-t tudott átpasszírozni a dróton, képtelen volt vezérlőparamétereket egyeztetni és az autentikációhoz is zokni volt. A PPP mindezeket tudja, sőt többet is. Ráadásul nagyjából ugyanolyan sebességet biztosít, mint a jóval egyszerűbb SLIP - így egyáltalán nem csoda, hogy teljesen kiszorította. A Windows Server 2008 és a Vista már nem is támogatja az SLIP-t.

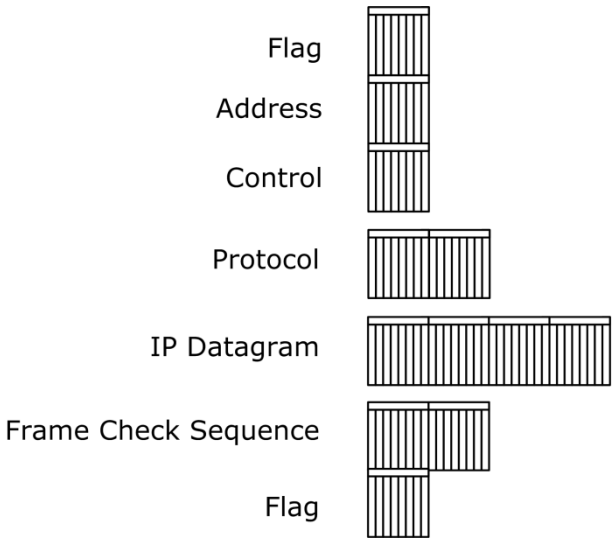
SLIP vs PPP:
http://sunsite.nus.sg/pub/slip/slip-vs-ppp.html

PPP:
http://en.wikipedia.org/wiki/Point_to_Point_Protocol
http://www.cisco.com/en/US/docs/internetworking/technology/handbook/PPP.html
http://www.tcpipguide.com/free/t_PointtoPointProtocolPPP.htm

Mint írtam, egy köteg protokoll. Ezek alapvetően három kategóriákba sorolhatók:

- Data Link Layeren belül csomagolgató protokollok, melyek biztosítják azt, hogy ugyanazon a dróton, akár egyidejűleg is, különböző magasabb szintű protokollokhoz tartozó csomagok tekerjenek.
- Link Control Protocol (LCP): a kapcsolat paramétereinek kölcsönös megtárgyalását, egyeztetését végző protokollcsalád.
- Network Control Protocols (NCPs): Protokollokat kontrolláló protokollok. A PPP-n különböző protokollok húznak át, ezeknek a paramétereit egyeztetik a két végpont között az NCP-k. (Ilyen például az Internet Protocol Control Protocol, azaz IPCP.)

Kezdjük a boncolást a szokásos módon. Nézzük meg, hogyan épül fel egy PPP csomag.
Hát. úgy, ahogyan egy HDLC (High-Level Data Link Control) csomag. Abból származik.



3.18. ÁBRA PPP CSOMAGOLÁS HDLC KERET SEGÍTSÉGÉVEL

FLAG: Az elefánt farka és ormánya. Mint a gyerekkori viccben, ez jelzi, hogy hol kezdődik és hol végződik az elefánt. Az értéke fix: 0x7E.

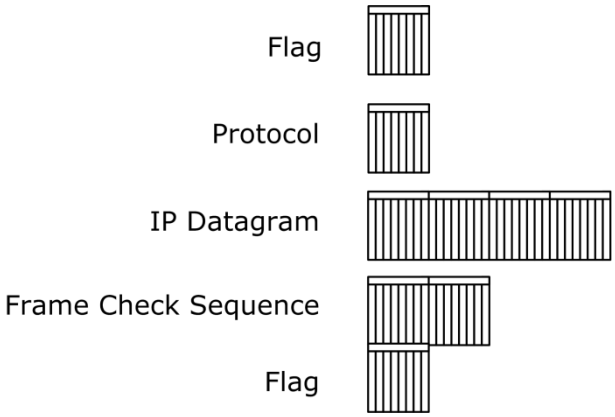
ADDRESS: Az eredeti HDLC keretben ebben az 1 bájtban volt a megcélzott node címe. Mint írtam, itt összesen két pont van, tehát ez a mező meglehetősen felesleges. Az értéke ennek is konstans: 0xFF.

CONTROL: Ez a mező is igazából a HDLC kommunikációban virít. Az értéke PPP csomag esetén konstans, illeszkedve a HDLC keret szabályaihoz: 0x03.

PROTOCOL: Az első nem konstans mező. A szállított protokollra utal. (Az IP például 0x00-21, a Novell IPX meg 0x00-2B.)

FRAME CHECK SEQUENCE: A jó öreg CRC, igaz, kissé rövidebb, mint ahogy megszokhattuk.

Nem tudom, te hogy vagy vele, engem határozottan zavart már első olvasáskor is ez a sávszélesség-pazarlás. Értelmetlen konstansokat küldözgetünk? Ilyen jól állunk a drótokkal?
Nyilván ez nem csak nekem jutott eszembe. Létezik egy másik PPP csomag is.



3.19. ÁBRA LEEGYSZERŰSÍTETT PPP CSOMAG

Megsúgom, ez a gyakoribb.

Az, hogy a konstans mezők elmaradtak, gondolom nem lep meg senkit. De nézzük csak meg, mi történt a Protocol mezővel? Összezuhant. Két bájt helyett egy lett.

Tanulmányozzuk át ezt a táblázatot:
http://www.iana.org/assignments/ppp-numbers.

Láthatjuk, hogy a legsűrűbben használt protokollok esetében a felső bájt üres. Nincs rá szükség.
Persze jogos lehet a kérdés, hogy mi van akkor, ha felváltva jön IP csomag meg mondjuk IPCP (0x80-21)? Nos, itt mutatkozik meg a PPP nagy ereje, a rugalmasság. Ahogy korábban is írtam, a PPP egyik protokollja az LCP, a Link Control Protocol. Ez egy olyan protokoll, melyen keresztül a PPP-re vonatkozó beállításokat egyezteti le a két node.Tehát mind azt, hogy elhagyható-e az ADDRESS mező, vagy a CONTROL mező, vagy olyan protokollról lesz szó, amelyiknél az alsó bájt üres-e... mindezt az LCP-n keresztül beszélik meg a felek. Utána pedig nyilván már tudják értelmezni a csomagokat.

Még néhány kis szines a PPP család életéből.

A PPP csomagoknak is van maximális méretük (MTU, azaz itt inkább Maximum Receive Unit, azaz MRU). Csak éppen nem fix méretűek. Az MRU maximális értéke 65535, a kiindulási értéke 1500 bájt, a tényleges értéke pedig helyzettől függ. A felek LCP-n keresztül állapodnak meg.

Amikor a két node közötti vonalról beszélünk, akkor nem feltétlenül egy madzagra kell gondolnunk. A PPP képes arra is, hogy két node között párhuzamos kiépült csatornákat (tipikus példa ISDN B csatornák) egyként kezeljen. Ezt úgy hívják, hogy PPP Multicast Protocol. Ehhez egy teljesen más keret szerkezet tartozik. Ha nem haragszol, ezt nem rajzoltam meg.
A lényeg, hogy a forgalom fragmentálva van, maguk a töredékek pedig szépen le van kezelve: sorszámok, pozíció értékek jelzik, hogy mi és hol ment át a dróton.

Süllyedjünk.

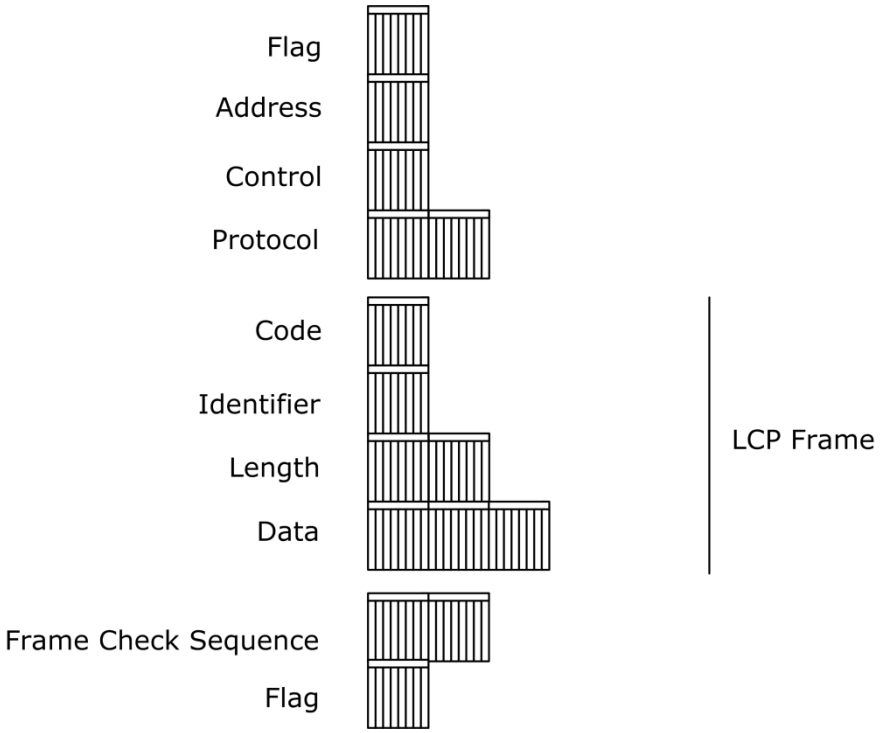
Alapvetően egy PPP kapcsolat felépítése négy lépésből áll:

- PPP konfigurálás LCP-vel.
- Autentikáció.
- Visszahívás.
- Protokoll beállítás NCP-vel.

3.3.1.1 PPP KONFIGURÁLÁS LCP-VEL

RFC 1662

Induljunk ki a teljes HDLC keretből, ezen egyszerűbb megmutatni, hol is jelenik meg az LCP.



3.20. ÁBRA AZ LCP KERET FELÉPÍTÉSE

Ugye, látjuk? Ha összehasonlítjuk a korábbi ábrával (*3.18. ábra PPP csomagolás HDLC keret segítségével*), nem nehéz észrevenni, hogy minden ugyanaz, csak az IP datagram helyett itt LCP keret van.
Nos, igen. Az LCP információk teljesen ugyanolyan csomagban utaznak, mint amilyen a teljes PPP csomag. Hogy miért a teljeshez hasonlít? Jó vicc, az LCP-nek nincs LCP-je, hogy letisztázza, mit lehetne elhagyni.

Nézzük a csomag belsejét.

CODE: 1 bájtos mező, az LCP üzenet tipusát jelzi.

3.2. TÁBLÁZAT

CODE	Kerettípus	Leírás
1	Configure-Request	Vágjunk bele egy PPP kapcsolat felépítésébe.
2	Configure-Ack	Amennyiben elfogadtuk a Configure-Request-et, akkor küldünk egy ACK-ot.
3	Configure-Nak	Amennyibe felismertük a konfigurációs opciókat, de azok elfogadhatatlanok.
4	Configure-Reject	Amennyiben nem ismertük fel a konfigurációs opciókat.
5	Terminate-Request	Zárjuk le a PPP kapcsolatot.
6	Terminate-Ack	Elfogadtuk a lezárást.
7	Code-Reject	Nem ismertük fel a kódot.
8	Protocol-Reject	Nem ismertük fel a protokollt.
9	Echo-Request	A PPP kapcsolat tesztelése
10	Echo-Reply	Válasz a PPP kapcsolat tesztelésére
11	Discard-Request	A linken túli kapcsolat tesztelése

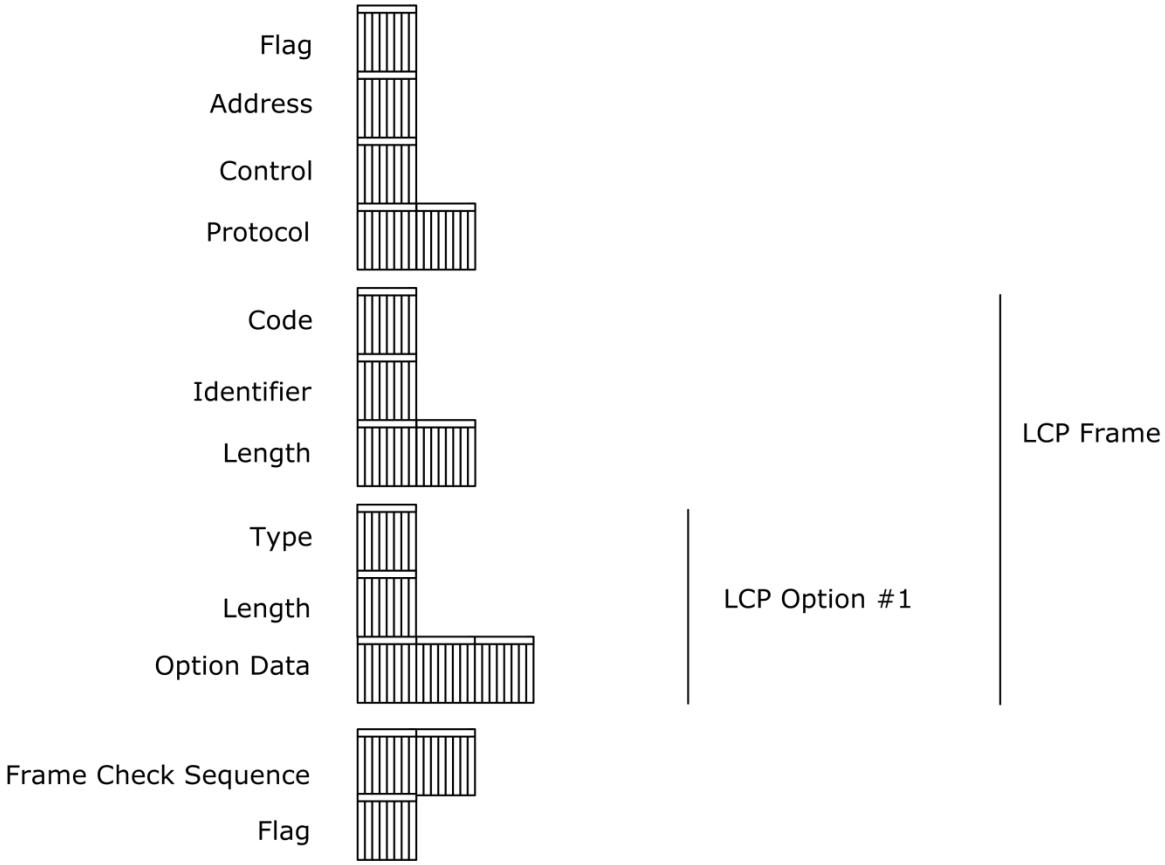
IDENTIFIER: 1 bájtos mező, egy érték, mely összekapcsolja a kéréseket és a válaszokat.

LENGTH: 2 bájtos érték, az LCP üzenet mérete.

DATA: Maga az LCP üzenet. A lényeg. Ezt hamarosan tovább is fogjuk boncolni.

Oké, akkor ahogy igértem, beleássuk magunkat az LCP adatok (DATA) világába. Kezdjük azzal, hogy tippeljük meg: vajon mindegyik kód értékhez van-e értelme LCP adatokat is küldeni? Ugye, nem? Gyakorlatilag csak a 'Configure' tipusú LCP keretekhez tartozik adatmező is.

Metéljük tovább az LCP keretet.



3.21. ÁBRA LCP OPTION EGY LCP KERETBEN

Az LCP DATA mező egy vagy több ún. LCP OPTION blokkból áll. Ezekben a blokkokban utaznak a tényleges konfigurációs paraméterek.

Az LCP OPTION felépítését egy korábbi ábrán (*3.21. ábra LCP Option egy LCP keretben)* láthatjuk.

TYPE: Az opció típusa.

LENGTH: A konkrét Option mérete. Többfajta Option blokk létezhet.

OPTION DATA: Most már tényleg eljutottunk a konfigurációs paraméterekhez.

A lehetséges értékeket a következő táblázat tartalmazza.

3.3. TÁBLÁZAT

Option Name	Type	Length	Leírás
Maximum Receive Unit (MRU)	1	4	A konkrét PPP kapcsolatban szállítható maximális csomagméret.
Asynchronous Control Character Map (ACCM)	2	6	Aszinkron kapcsolatban melyik ASCII vezérlőjel legyen az Escape.
Authentication Protocol	3	5 v. 6	A PPP kapcsolatban melyik authentikációt használjuk. Az értékeket a *3.4. táblázat* tartalmazza.
Magic Number	5	6	Véletlenszám hurokdetektálási célból.
Protocol Compression	7	2	Flag. A feladó ezzel jelzi, hogy egybájtos protokollazonosítót szeretne használni.
Address and Control Field Compression	8	2	Flag. A feladó ezzel jelzi, hogy nem szeretne Address és Control mezőket látni a csomagokban.
Callback	13	3	Valamikor a visszahívás egyeztetésére használták. Ma már erre a CBCP Network Control Protocol szolgál.

3.4. TÁBLÁZAT

Autentikáció kódja	Autentikáció neve
0xC2-27	Extensible Authentication Protocol (EAP)
0xC2-23-81	MS CHAPv2
0xC2-23-05	Message Digest v5 Challange Handsake Authentication Protocol (MD5-CHAP)
0xC0-23	Password Authentication Protocol (PAP)

3.3.1.2 PPP AUTENTIKÁCIÓ

Lám, lám eljutottunk az autentikációkig. Jó ideig itt is fogunk időzni.

Mint eddig is láthattuk, az LCP-n belül a felek letisztázták, milyen autentikációt fognak használni. Windows Server 2008 illetve Vista alatt a következők jöhetnek szóba:

- Password Authentication Protocol (PAP)
- Challenge Handshake Authentication Protocol (CHAP)
- Microsoft Challenge Handshake Authentication Protocol version2 (MS-CHAP v2)
- Extensible Authentication Protocol (EAP)

3.3.1.2.1 PASSWORD AUTHENTICATION PROTOCOL, PAP

RFC 1334

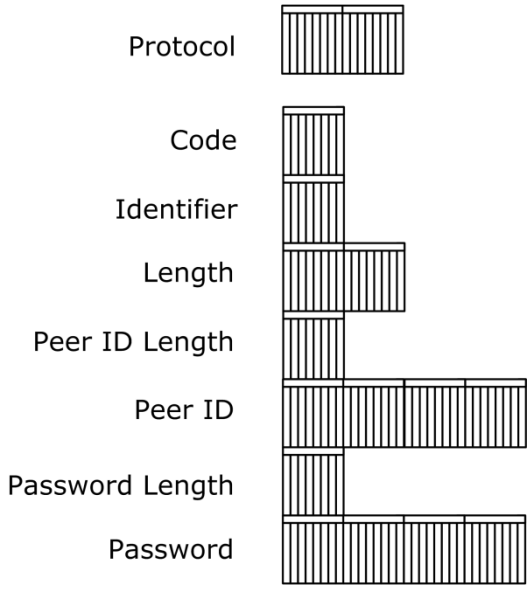
Habár azt írtam, hogy "jöhetnek szóba", de ennél az autentikációnál ezt nem kell szószerint érteni. Ha lehet, akkor maradjunk annyiban, hogy ez inkább szóba sem jöhet.
Miért is nem? Mert plain-text formátumban küldi át a jelszót a dróton. Ami azért elég gáz. Nem is ajánlott éles környezetben használni, maximum csak vonaltesztelési célokra, egyszer használatos felhasználókkal.

Nézzük akkor a koreográfiát:

- Első körben a kapcsolódni kívánó fél küld egy PAP Authentication-Request csomagot az autentikáló félnek. Ebben a csomagban benne van a kapcsolódni vágyó felhasználó neve és jelszava. Plain-text. Naná.
- Az autentikáló fél erejét megfeszítve kikódolja a jelszót, majd összehasonlítja a nála tárolt usernév/password párossal. Ha stimmel, akkor egy PAP Authentication-ACK csomagot küld vissza. Ha nem, akkor egy PAP Authentication NAK csomagot.

Nem sok egyszerűbb dolog létezik a világon.

Ugyanez csomagszinten - pusztán csak azért, hogy ha elkapunk egy ilyen csomagot a hálózaton, akkor ne jöjjünk zavarba a jelszó kiolvasásánál.



3.22. ÁBRA PAP REQUEST

PROTOCOL: Az autentikáció tipusa. PAP esetén 0xC0-23.

CODE: Egy bájtos érték, a csomag tipusára utal. Az Authentication-Request értéke 1.

IDENTIFIER: Egy bájtos érték, mely az összetartozó PAP csomagokat azonosítja.

LENGTH: Vajon? Az egész csomag mérete.

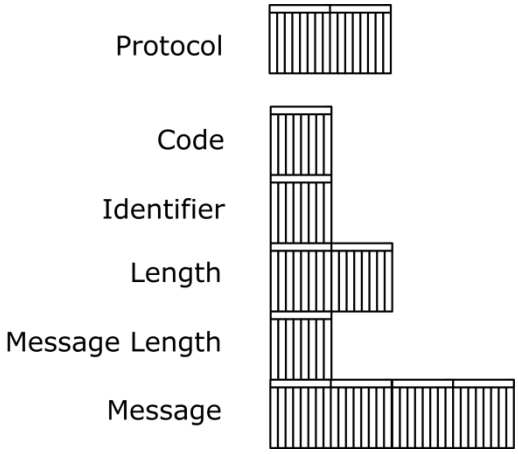
PEER ID LENGTH: Egy bájtos érték, a felhasználó nevét tartalmazó, változó méretű mező mérete.

PEER ID: A felhasználó nevét tartalmazó változó méretű mező.

PASSWORD LENGTH: Egy bájtos érték, a felhasználó jelszavát tartalmazó, változó méretű mező mérete.

PASSWORD: A felhasználó jelszavát tartalmazó változó méretű mező. (Igen, ezt kell kiolvasni a csomagból.)

Nézzük ugyanezeket a válaszcsomagnál.



3.23. ÁBRA PAP AUTHENTICATE

PROTOCOL: Ugyanaz.

CODE: Nagyjából ugyanaz. Az Authenticate-ACK értéke 2, az Authenticate-NAK értéke 3.

IDENTIFIER: Ugyanaz.

LENGTH: Ugyanaz.

MESSAGE LENGTH: Egy bájtos érték, az üzenetet tartalmazó, változó méretű mező mérete.

MESSAGE: Itt lehet tetszőleges szöveget üzenni az autentikálni kívánó félnek. A Windows nem használja ezt a mezőt.

Nos, ez volt a legegyszerűbb autentikáció. Nem biztos, hogy megnyugtató, de ennél csak bonyolultabbak jönnek.

3.3.1.2.2 CHALLENGE HANDSHAKE AUTHENTICATION PROTOCOL (CHAP)

RFC 1994

Egy árnyalattal biztonságosabb, mint a PAP. Igen, tudom, most csaptad földhöz a könyvet. Mit jelent az a biztonságiparban, hogy "egy árnyalattal"? Valami vagy biztonságos vagy nem. Nem?
Elmesélem a kedvenc viccemet. Nem mintha ez lenne a legkacagtatóbb vicc a világon, de tanulság téren ott van az élmezőnyben.

Két turista gyalogol a szavannában. Egyszer csak meglátják messziről, hogy egy oroszlán rohan feléjük. Az egyik gyorsan leül, előkap a hátizsákjából egy edzőcipőt és lecseréli rá a túrabakancsát. A másik értetlenkedve nézi:
- Te komolyan azt hiszed, hogy ebben a cipőben lefutod az oroszlánt?
- Az oroszlánt? Azt nem. De téged igen.

Azaz nem tudunk abszolút biztonságosak lenni... csak biztonságosabbak. Annyira, hogy már ne érje meg a rosszfiúknak velünk vacakolni. Nyilván ez a határ folyamatosan változik. Ami valamikor nagyon sok erőforrást igényelt, az ma már a gyerek otthoni gépén is elfut. Ezért is jönnek ki új, meg új módszerek.

Valamikor a CHAP nem is volt olyan rossz autentikációs eljárás[23]. A koncepció lényege az, hogy a jelszó ne utazzon át a dróton. Ugye, ha nem megy át a jelszó, vitézkedhetnek a capture-huszárok, nem tudják megszerezni.

Persze, valaminek azért át kell mennie. Nincs olyan, hogy varázsszóra előbukkan egy-egy buborékból mindkét helyen a jelszó.

A kulcsszó: OWF[24]. Azaz One Way Function. Magyarul olyan függvény, mely csak az egyik irányba működik. Fóti Marci frappáns hasonlatával élve: a húsdaráló. Egyik oldalon belerakjuk a disznót, a másik oldalon kijön a darált hús. Hiába szeretnénk megfordítani a folyamatot, nem fog sikerülni.

Cryptographic hash function:
http://en.wikipedia.org/wiki/Cryptographic_hash_function

[23] A CHAP és az MS-CHAP v1 nagyjából azonos logikát követő eljárásokat jelent. Egyedül a darálás, azaz az OWF más. Az MS-CHAP v1-et a Windows Server 2008/Vista vonal már nem támogatja.
[24] Nem azonos a hasonlóan hangzó WTF-fel.

Matekban ezt sokféleképpen lehet elérni. Magát a darálási algoritmust hash-nek[25] hívják. De az OWF nem pusztán darálást jelent. Bejön a képbe a só is.

Nyugi, jó helyen jársz, ez nem egy szakácskönyv. Sónak (salt) azt az értéket hívják, amellyel egybedarálják a preparálandó értéket. (Ezt általában azért csinálják, hogy védekezzenek az ún. dictionary attack, azaz a szótáron alapuló próbálkozás ellen. Ez jó is akkor, ha a só nem publikus. Ha az, akkor adtunk egy pofont az exkrementumnak.)

Még egy fontos tulajdonság a hash algoritmusokról: egyértelműek. Azaz ha van egy érték és arra ráküldök egy hash algoritmust, mindig ugyanazt a tömör értéket (darált hús) kapom.

Nézzünk egy általános OWF folyamatot. Fogom a jelszót, beledobom a húsdarálóba. Mellédobok valami sót. Ez általában egy fix érték, melyet a szervertől kapok (challenge). Az algoritmusból kijön egy érték, ezt elküldöm a partnernek (response). Hiába kapja el útközben Hekker Henry, a fasírtból már nem tud disznót gyártani. A szerver ismeri a sót - naná, ő küldte - nála is megvan a jelszó, tehát ugyanúgy össze tudja darálni az egészet. Mivel a hash eredménye egyértelmű, így ha a két érték megegyezik, akkor jó volt a jelszó.

Ha már kezdenéd érteni, megjegyzem, rengeteg chap jellegű autentikáció létezik. Ezek leginkább az OWF folyamatban térnek el egymástól. Egyik erre tesz még két tánclépést, amaz üveggel a feje tetején cifrázza... színes a világ. Mindenesetre a Windows Server 2008 és Vista vonal az egyszerűbbek közül a CHAP autentikációt ismeri, a bonyolultabbak közül meg az MS-CHAPv2 autentikációt. (Habár az utóbbival is fogok bővebben foglalkozni, előljáróban legyen elég annyi, hogy a CHAP csak egyirányú autentikációt ismer, azaz a szerver meg tud győződni arról, hogy a kliens tényleg a kliens-e... de fordítva nem. A kliens nem lehet biztos abban, hogy a szerver tényleg a szerver-e. Ezt majd csak az MS-CHAPv2 fogja tudni.)

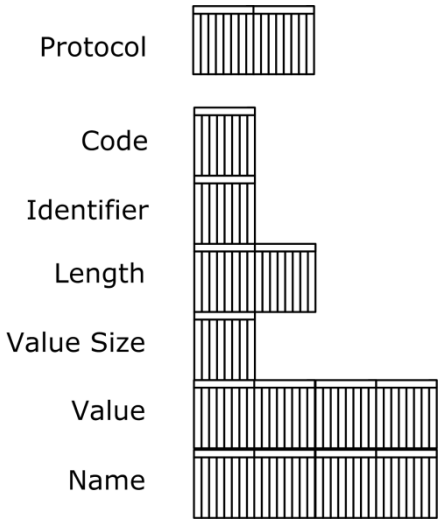
Chap:
http://en.wikipedia.org/wiki/Challenge-handshake_authentication_protocol

[25] SHA-x, MDy, stb...

Akkor most már nézzük meg konkrétan, hogyan is működik az a CHAP.

- Először is, MD5-öt használ. (Ezt becsületesen jelzi is az LCP megfelelő mezőjében.)
- Az autentikáló fél (a szerver) küld egy kihívást (challenge). Ebben van egy session ID, egy karaktersorozat (a só) és a saját neve.
- Az autentikálandó fél elkészít egy választ (response). Ez úgy néz ki, hogy a session ID-t, a sót és a jelszót bedarálja MD5-tel, majd melléteszi a felhasználó nevét. El is küldi.
- Az autentikáló fél legyártja a saját darált husiját: ismeri a session ID-t, a sót és a jelszót. Ha a két hash megegyezik, akkor visszaküld egy CHAP Success üzenetet. Ha nem, akkor egy CHAP Failure üzenet megy vissza.

A sok duma után nézzük meg, hogyan is néz ez ki csomagszinten. Első körben lássuk a Challenge/Response csomagok szerkezetét.



3.24. ÁBRA CHAP CHALLENGE/RESPONSE

PROTOCOL: A CHAP esetében az érték: 0xC2-23.

CODE: Azonosító. Challenge esetében 1, Response esetében 2.

IDENTIFIER: A session ID.

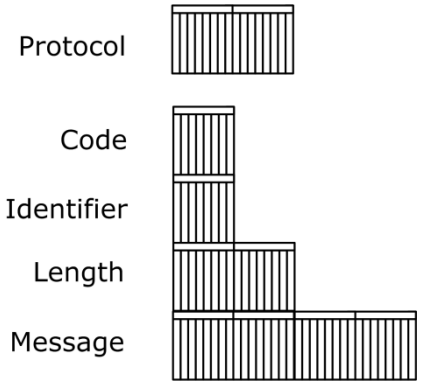
LENGTH: A CHAP csomag mérete.

VALUE SIZE: Egy bájtos érték, a lényeges információt tartalmazó, változó méretű mező (VALUE) mérete.

VALUE: Itt utazik a lényeg.

NAME: Az aktuális feladó nevét tartalmazó mező.

Erre jöhet válaszként a CHAP Success, illetve CHAP Failure.



3.25. ÁBRA CHAP SUCCESS / FAILURE

CODE: Azonosító. Success esetén az értéke 3, Failure esetén 4.

IDENTIFIER: Session ID.

LENGTH: A csomag mérete.

MESSAGE: Itt lehet visszadumálni. CHAP esetén a Windows nem teszi.

Megszüntetett autentikációk a Windows Server 2008-ban:
http://www.windowsnetworking.com/articles_tutorials/Windows-Server-2008-Networking-Services.html

3.3.1.2.3 MICROSOFT CHALLENGE HANDSHAKE AUTHENTICATION PROTOCOL VERSION2 (MS-CHAP V2)

RFC 2759

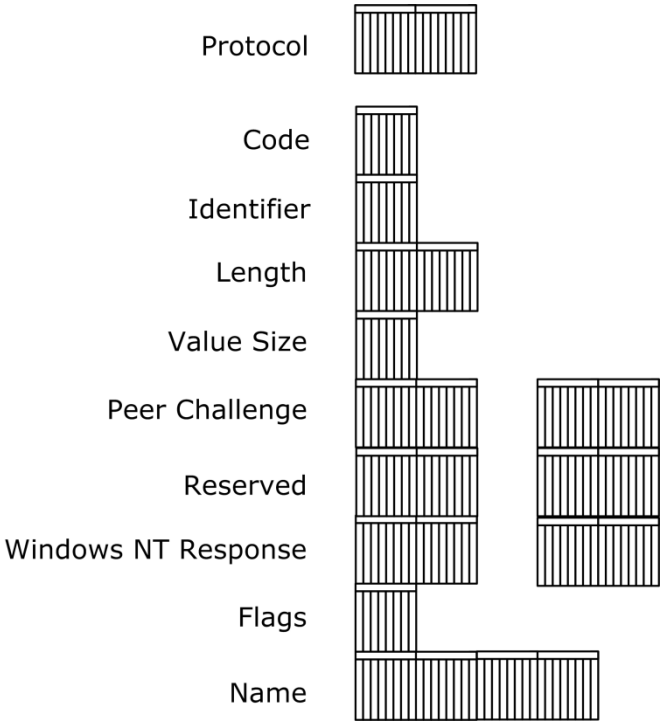
Mint írtam, ez is egyfajta CHAP autentikáció, csak ez kölcsönös azonosításra is képes. Gondolom, nem kell különösebben magyaráznom, mennyivel biztonságosabb ez a módszer. Bolond világban élünk, ma már az autentikáló felet is hamisíthatják.
A közös gyökér ott is látszik, hogy a PROTOCOL mezőben ugyanaz lesz a kód, mint a CHAP estén: 0xC2-23. Viszont az LCP-ben a CHAP verzión belüli kód már más. (Lásd *3.4. táblázat*.)

Akkor nézzük, ebben az esetben milyen a táncrend.

- Az autentikáló fél elküldi a szokásos CHAP(!) csomagot. (Challenge.)
- Az autentikációt kérő fél visszaküld egy MS-CHAPv2 Response csomagot. Ebben benne van a felhasználónév, a só az autentikáló fél számára és a darálthús, azaz az autentikáló fél által küldött só, egybedarálva a felhasználó jelszavával, ez utóbbi ráadásul titkosítva. Jelen esetben MD4 hash algoritmust használunk. (Vegyük észre, megváltozott az OWF: MD5 helyett MD4, és a session ID nincs beledarálva a válaszba.)
- Az autentikáló fél eljárja ugyanezt a pár lépést: só, jelszó, hash, titkosítás. Amennyiben a két számítási eredmény megegyezik, akkor egy CHAP Success csomagot vesz le a polcról. Amennyiben nem, akkor egy CHAP Failure csomagot. Mindkét esetben összerakja, ahogy kell. (Lásd *3.25. ábra CHAP Success / Failure*) Csakhogy ezzel még nincs vége, még neki is autentikálnia kell magát. Ez egy kicsit trükkös, ugyanis nem azt csinálja, hogy elküldi a saját jelszavát - hiszen honnan is tudhatná szegény autentikációra váró fél, hogy mi a túloldali service account user jelszava? Nem, ehelyett egy teljesen másik OWF eljárással bolondítja meg a felhasználó jelszavát, majd ezt a csomagot küldi vissza. Ugyanezzel az OWF eljárással fog nekiugrani a kliens is - és ha az eredmény ugyanaz, akkor megnyugodhat, mert az autentikáló fél tényleg olyan partner, aki már korábban is tudta a jelszavát. Most már csak azt kellene letisztázni, mi is ez az új OWF? Kapaszkodj, nem lesz egyszerű. Fogja a kliens által neki küldött sót, melléteszi a saját maga által korábban elküldött sót, belecsapja a kliens által korábban küldött teljes választ, a kliens felhasználói nevét és a jelszavát, majd ezt az egészet betitkosítja. Mi? Hogy hol van itt az OWF? Jogos a kérdés, nekem is végig kellett túrnom hozzá az RFC2759-et, hogy megtaláljam. Tehát nem a kliens jelszavát teszi direktben bele a csomagba, hanem végigküldi a jelszót egy MD4 darálón, sőt, utána a darált húst ismét ledarálja - majd ez kerül a később titkosítandó csomagba.

- Az autentikációra váró kliens először is örül, ha CHAP Success csomagot kap - de utána mindenképpen le kell ellenőriznie, hogy az annak a MESSAGE mezőjében érkező új katyvasz megegyezik-e az általa ugyanúgy, ugyanolyan bonyolultan előállított lokális katyvasszal. Ha nem, akkor kötelezően el kell dobnia a csomagot.

Nézzük ugyanezt grafikusan is.



3.26. ÁBRA MS-CHAP V2 RESPONSE

CODE: Azonosító. MS-CHAPv2 Response esetén az értéke 2.

IDENTIFIER: Ugyanaz, mint a CHAP esetében.

LENGTH: Az egész csomag mérete.

VALUE SIZE: A CHAP-hoz hasonlóan itt is a VALUE mező mérete - azzal az apró különbséggel, hogy itt a PEER CHALLENGE, RESERVED, WINDOWS NT RESPONSE és FLAG mezők együttesét tekintjük VALUE mezőnek.

PEER CHALLENGE: A só az autentikáló fél számára.

RESERVED: Majd. Egyelőre 0.

WINDOWS NT RESPONSE: Itt megy vissza a titkosított válasz.

FLAGS: Majd. Egyelőre 0.

NAME: Az autentikációt kérő fél neve.

3.3.1.2.4 Extensible Authentication Protocol (EAP)

RFC 3748

Megyünk befelé, mint maci a málnásba.

Eddig ugyanis viszonylag egyszerű volt az élet. LCP, megállapodtunk az autentikációban, eltáncoltuk a méhecske táncot - és már azonosítottuk is magunkat... vagy egymást.
Az EAP viszont egy újabb absztrakciós szintet vezet be.
Az EAP ugyanis egy általános, autentikációs célú hordozó közeg, direkt a PPP számára kitalálva. Egy olyan protokoll, mely tetszőleges autentikációs protokollt képes magába kapszulázni.

Hogy mi az értelme?

Nehéz erre röviden válaszolni.
Az eddig tárgyalt autentikációs módszerek rögzített formátumúak voltak, ráadásul az üzenetváltások száma is limitált volt. (Kettő, vagy három.) Az EAP ezzel szemben ad egy rugalmas szerkezetet, ahol mindenki azt pakol a mezőkbe, amit akar és a két fél annyit beszélgethet egymással, amennyit akar.
Ebből az is következik, hogy létezhetnek az EAP keretein belül a már jól ismert protokollok: semmi akadálya nem lenne, hogy csináljunk egy EAP-PAP protokollt például. Mondjuk, értelme sem. Nem is létezik. Sőt, a Windows Server 2008 és Vista rendszerekben még az EAP-CHAP sem támogatott.

Akkor mégis milyen 'plugin' autentikációkat dughatunk az EAP-ba?

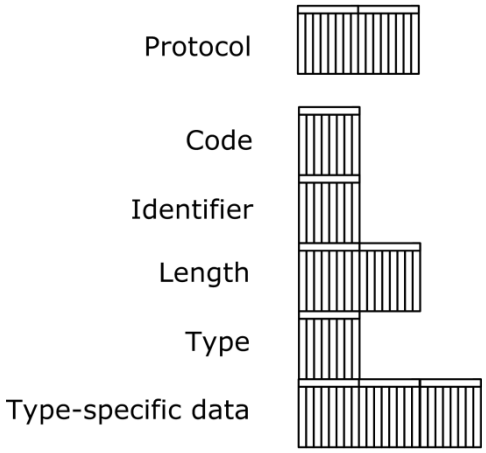
- EAP-TLS (Smart Card Or Other Certificate névre hallgat a Windows GUI-ban)
- PEAP: Protected EAP (EAP az EAP-ban.)

Én mondtam, hogy megyünk a málnásba. Ne gondold, hogy viccelek.

De mielőtt nagyon elszaladnánk, nézzük azokat a dobozokat.

Eleve az EAP protokollon belül négy üzenettípust különböztetünk meg:

- EAP-REQUEST: Az autentikáló fél küldi a behatolni igyekvőnek.
- EAP-RESPONSE: Az autentikálandó fél küldi a cerberusnak. Mind a két fajta üzenetből tetszőleges számú lehet egy autentikációs folyamat során.
- EAP-SUCCESS: Az autentikáló fél küldi, amennyiben sikerült a bejelentkezés.
- EAP-FAILURE: Az autentikáló fél küldi, amennyiben nem sikerült a bejelentkezés.



3.27. ÁBRA EAP-REQUEST ILLETVE EAP-RESPONSE

PROTOCOL: Az EAP kódja 0xC2-27.

CODE: Azonosító. Request:1. Response: 2.

IDENTIFIER: Ez pontozza össze a Request-Response csomagokat.

LENGTH: Az EAP üzenet mérete.

TYPE: Az EAP-on belüli információ tipusa.

TYPE-SPECIFIC DATA: Maga az információ.

Látjátok, ugye? Ez egy teljesen általános doboz, egy univerzális szállítóeszköz. Minden a TYPE és TYPE SPECIFIC DATA mezőkön múlik. Van belőlük néhány.

3.5. TÁBLÁZAT

Type	Típus	Megnevezés
1	Identity	Amennyiben EAP-Request-ről van szó, akkor az autentikáló fél kéri a bekéredzkedő fél nevét. Amennyiben EAP-Response-ról van szó. akkor meg pont fordítva.
2	Notification	Az autentikáló fél szöveges üzenetei a másik félnek. Hátha kiírja a képernyőre.
3	Nak	Az autentikálandó fél segélykiáltása, amennyiben nem ismeri a kért autentikációs módszert. A TYPE-SPECIFIC DATA mezőben ekkor visszaküldi, hogy ő miket is ismer.
13	EAP-TLS	Erről van szó.
25	PEAP	Erről van szó.
29	EAP-MS-CHAP-V2	Erről van szó.

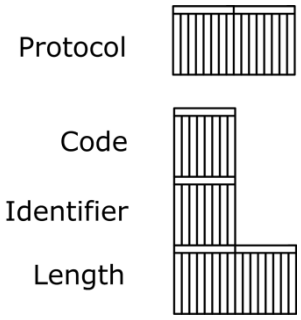
Én mondtam, hogy málnás. Látható, hogy az autentikáció beazonosítása kicsúszott az LCP kezei közül. Az LCP annyit fog tudni, hogy EAP. Aztán majd a Request-Response kommunikáción belül letisztázzák a felek, melyik autentikációról is lesz konkrétan szó.

Hadd kérdezzem meg, miért hajtod le a fejed olyan szégyenlősen és rugdosod azt a láthatatlan kavicsot, látszólag teljes odafigyeléssel? Ha valami nem stimmel, kérdezzél nyugodtan.

Igen, teljesen igazad van. De nekem is.

Azt írtam korábban, hogy az EAP-ba bele lehet dugni a TLS-t és a PEAP-ot. A táblázatban meg ott szerepel az MS-CHAPv2. Akkor most wtf?
Nos, az van, hogy habár elfogadható az MS-CHAPv2 is, de azt igazából úgy csempészik majd be. A PEAP-on keresztül. De erről később részletesen is írok majd.

Nézzük akkor a többi EAP csomagot.



3.28. ÁBRA EAP-SUCCESS ILLETVE EAP-FAILURE

CODE: Azonosító. Success: 3. Failure: 4.

IDENTIFIER: A legutolsó EAP-Response azonosítója.

LENGTH: Fixen 4.

3.3.1.2.5 EAP-MS-CHAP v2

Csak azért, mert ezt már jól ismerjük, így láthatjuk, hogyan 'eaposodik' egy autentikációs protokoll. (Egyébként, mint írtam, sima EAP-pal nem szokás MS-CHAPv2-t használni.)

- Az autentikáló fél küld egy EAP-Request csomagot, melynek a tipusa: Identity.
- A bekéredzkedő küld egy EAP-Response csomagot, ennek tipusa szintén Identity.
- Egyből megy is vissza egy EAP-Request csomag, melynek tipusa MS-CHAP v2, a tartalma pedig teljesen ugyanaz a Challenge, mint amilyenről korábban is írtam.
- Innetől pedig teljesen ugyanúgy történik minden, csak éppen most már az EAP csomagok TYPE SPECIFIC DATA mezőiben utaznak az információk.

3.3.1.2.6 EAP-TLS

RFC 2716, 2246

Nos, itt egy új fiú a blokkban: a TLS. Eddig róla nem volt szó.

Egy kicsit messziről fogok indítani. Amikor titkosítani akarok egy kommunikációt, akkor azt kétféleképpen tehetem meg: vagy egy szimmetrikus kulcsú titkosítást használok, vagy egy asszimetrikusat. Mindkettőről rengeteget lehetne írni, de én most csak két tényre fogok szorítkozni:

- A szimmetrikus kulcsú titkosítás gyors. Piszkosul gyors. Sokkal gyorsabb, mint az asszimetrikus.
- A szimmetrikus kulcsú titkosításnak van egy súlyos hibája: a kulcsot valahogy el kell juttatni mindkét kommunikáló félhez, még akkor is, ha azok többezer kilométer távolságra vannak egymástól, közöttük pedig ott feszül a hekker óceán.

Gondolom, nem kell hozzá zseninek lenni, hogy beugorjon a megoldás: hát küldjük el a szimmetrikus titkosítás kulcsát (session key) asszimetrikus titkosítással, ez lassú ugyan, de csak egyszer, maximum kétszer kell használni - utána pedig hadd szóljon, ami a csövön kifér, szimmetrikus titkosítással.

Ezt hívják úgy, hogy TLS.
Már megint furcsán nézel. Igen, ezt nevezik úgy is, hogy SSL. Az elv ugyanaz, de van egy óriási különbség: a TLS az SSL utódja. SSL-ből volt az 1.0 (mely soha nem lett publikálva), volt a 2.0 (mely bugzott, mint macska éjjel), aztán jött a 3.0, mely már

egészen jó volt - de nem folytatták a sort, mert jött helyette a TLS 1.0, aztán az 1.1 és most éppen az 1.2 az aktuális. Hogy mi minden változott az egyes verziókban, arra most nem térnék ki. De ha például olyat látsz valahol, hogy most implicit vagy explicit titkosítást szeretnél, akkor tudjál róla, hogy az úgynevezett inicializációs vektor a TLS1.1-től explicit. Azaz az implicit az jó eséllyel SSL 3.0 lesz.

Nézzük akkor, mik is a TLS előnyei:

- Biztonságos. Jelenlegi ismereteink, a jelenlegi számítási teljesítmények ismeretében elmondható, hogy nem törhető. (Persze nem mindegy, hány bitesek a kulcsok.)
- Nagyon könnyen kölcsönössé tehető. Menjünk csak végig a folyamaton!
    - A node bekopog B node-hoz.
    - B node elkéri A node tanúsítványát.
    - B node elküldi A node-nak a session key-t, melyet A node publikus kulcsával titkosít.
    - A node a privát kulcsával kicsomagolja a session key-t.
    - A node elkéri B node tanúsítványát.
    - A node becsomagolja a korábbi session key-t B node publikus kulcsával, majd elküldi neki.
    - B node kicsomagolja a session key-t a privát kulcsával.
    - Bájos melléktermékként mindkét félnél ott ragadt egy session key, mellyel már tudják titkosítani a további kommunikációt. Nyilván ha valamelyik hazudott, akkor a kommunikáció sem jön létre.

Vegyük észre, hogy eddig a TLS-ről beszéltem, úgy általánosságban. A fejezet címe viszont az, hogy EAP-TLS.
Megijedni nem kell, habár már nem sok látszik az égből a sok málnabokortól. Tulajdonképpen az történik, hogy a TLS kommunikáció a korábban már megismert EAP csomagokon belül fog utazni. A TYPE értéke 13 lesz, a lényegi információ pedig a TYPE-SPECIFIC DATA mezőkben utazik.

3.3.1.2.7 PROTECTED EXTENSIBLE AUTHENTICATION PROTOCOL (PEAP)

Ahogy a szólás mondja, attól mert paranoiás vagy, még nem biztos, hogy nem üldöznek. Nézzük csak meg az előző két protokollt: mind az EAP-TLS, mind az EAP-MS-CHAPv2 esetében a lényeg, azaz a TYPE-SPECIFIC DATA mező tartalma titkosítva van. De a többi információ nem. És van, amikor ez is számít.

Hogyan tehetjük még biztonságosabbá ezt a kommunikációt? Úgy, hogy beletereljük az egészet egy TLS csatornába.

Vigyázat, málnás, nagyon málnás!

Tudsz követni?
A PPP kapcsolatban résztvevő két node, mielőtt még bármi is történne, egy EAP kapcsolaton keresztül kiépít egy TLS kapcsolatot. Ebben a kiépítésben nem jelenik meg a tényleges felhasználói név és jelszó, még egyszer hangsúlyozom, a node-ok építik ki a csatornát.. Aztán amikor már van mindkét oldalon session key, akkor indul be a tényleges EAP autentikáció, immár a bejelentkezni kívánó user adataival.
Ez a bizonyos második EAP (azaz a PEAP-on belüli EAP) kétféle lehet:

- EAP-MS-CHAP v2[26] (PEAP-on belül Secured Password néven találjuk a GUI-n)
- EAP-TLS (PEAP-on belül Smart Card Or Other Certificate névre hallgat)

TLS-en belüli TLS... szép, mi. És néha még mi nevezzük paranoiásnak magunkat. Hol vagyunk mi egy RFC gyártó biztonsági szakembertől?

[26] Itt már lehet.

3.3.1.3 VISSZAHÍVÁS

Ma már elég kicsi az előfordulási valószínűsége, de része a folyamatnak, így emlékezzünk meg róla is. Magát a folyamatot az ún CallBack Control Protocol, azaz CBCP viszi végig. Nem kell túl bonyolult dolgokra gondolni, az LCP-nél megismert csomagokat használja, a következő eltérésekkel:

- A PROTOCOL ID értéke 0xC0-29
- A CODE értékek az LCP-nél leírtak lehetnek, de csak az első 7 fordulhat elő. Az első 3 esetben (Callback Request [1], Callback Response [2] és Callback ACK [3]) az LCP OPTION mezőkben utaznak a kísérő információk, melyek leginkább telefonszámok.

3.3.1.4 PROTOKOLL BEÁLLÍTÁSOK NCP-VEL

Itt elsősorban IPCP-vel és CCP-vel fogunk foglalkozni. ECP-vel bizony mondom, nem.

Ezek ugyanis mind NCP-k.

Másfél sor. De egy fél templomnyi tömjénfüst.

Kezdjük ott, hogy mi is ez a meglehetősen zavaros NCP, azaz Network Control Protocol? Az, amit a neve is sugall. Egy protokoll, mely segíti, hogy a két pont között a hálózat működjön. Pontosabban, az NCP nem kizárólag egy protokollt jelent, hanem egy családról van szó. Ennek tagjait soroltam fel az első két mondatban:

- IPCP: IP Control Protocol
- CCP: Compression Control Protocol
- ECP: Encryption Control Protocol

3.3.1.4.1 IP CONTROL PROTOCOL

RFC 1332, 1877

Ez tulajdonképpen egy DHCP, csak éppen két pont között. A DHCP szerver funkciót az autentikáló node játssza el, olyan értelemben, hogy ő osztja ki az IP értékeket a betámadó node-nak.

A felépítés teljesen hasonló, mint az előző alfejezetben tárgyalt CallBack Control Protocolnál: LCP csomagokban utaznak az információk, csak éppen egyes mezőknek speciális értékeik lesznek.

Ilyenek.

A PROTOCOL ID értéke: 0x80-21.

A CODE értékek közül szintén csak az első 7 lehetséges. Az első négyhez tartozik LCP OPTION blokk is, ezek tartalmát az alábbi táblázat mutatja:

3.6. TÁBLÁZAT

Option Name	Type	Length	Description
IP Address	3	6	A kiosztott IP cím
Primary DNS Server Address	129	6	Az elsődleges DNS szerver címe
Primary NBNS Server Address	130	6	Az elsődleges WINS szerver címe
Secondary DNS Server Address	131	6	A másodlagos DNS szerver címe
Secondary NBNS Server Address	132	6	A másodlagos WINS szerver címe

A gondos megfigyelőnek feltűnhetett, hogy a kérvényező azért nem kapott meg mindent. Például hol van a subnet mask? Alapértelmezésben. Ne felejtsük el, hogy két pont között építünk hálózatot, tehát subnet mask-nak tökéletesen megfelel a 255.255.255.255. Default gateway? Nyilván az autentikáló fél IP címe. Lehet egynél több Default Gateway egy hoston? Nem igazán. De a számítógép okos, az IPCP után a route táblába írja be az autentikáló fél IP címét, méghozzá úgy, hogy az legyen a legkisebb költségű. Amennyiben létezett korábban 1-es metric, akkor mindegyik korábbi bejegyzés metric értékét megnöveli eggyel, a friss bejegyzését pedig egynek veszi. (De erről le lehet beszélni.)
Mi nincs még? Mondjuk DNS domain név. Na, az nem is lesz. Legalábbis tisztán IPCP segítségével biztosan nem. Ilyenkor jön az, hogy lopunk, csalunk, hazudunk - azaz proxy. Amennyiben az autentikáló fél - aki az IPCP-t is biztosítja - képes DHCP proxyként viselkedni, és van a környéken DHCP szerver, és a betörekvő node Windows Server 2008 vagy Vista, akkor az küld egy DHCPINFORM üzenetet, az autentikáló fél proxyzza a valódi DHCP szerver felé, a választ dettó a másik irányba,

végül az ifjú padawan eldob minden opciót, eltekintve a 15-östől, mely a DNS Domain Name névre hallgat.
Az élet szép. Csak nem egyszerű.

3.3.1.4.2 Compression Control Protocol, CCP

RFC 1962

Semmi meglepetés nem várható. Ez egy olyan protokoll, melynek segítségével a felek megbeszélik, hogy milyen tömörítési technikát használjanak a különböző adatcsomagok küldözgetésére.
Egy közül lehet választani.

RFC 3078

Legalábbis a Microsoft rendszerekben. Ez pedig az MPPC, azaz a Microsoft Point-to-Point Compression. Jelzem, hogy árukapcsolás ténye forog fenn, azaz aki MPPC-t választott, az nagy valószínűséggel [27] kapott vele MPPE-t is, mely a Microsoft Point-to-Point Encryption névre hallgató titkosítási folyamatot takarja. A szakirodalomban úgy is szoktak előfordulni, hogy MPPC/MPPE.

Természetesen ez a protokoll is az LCP csomagformátumot használja.

A PROTOCOL ID értéke: 0x80-FD.

Szintúgy csak az első 7 LCP CODE használatos, ezek közül az első négynél fordulhatnak elő LCP OPTION blokkok.

3.7. táblázat

Option Name	Type	Length	Description
Organization Unique Identifier	0	min. 6	A tömörítési módszer beazonosítása
Microsoft Point-to-Point Compression (MPPC)	18	6	MPPC/MPPE esetén a szükséges paraméterek (leginkább a titkosításhoz használt kulcs hossza, a titkosítás erőssége, meg ilyenek)

[27] Feltéve, hogy a korábbi autentikációs fázisban a felek valamelyik kölcsönösen autentikáló módszerben egyeztek meg. Ellenkező esetben nincs MPPE. Meg az erősebb titkosítást biztosító tunnelling protokollok esetében sem.

3.3.1.4.3 Encryption Control Protocol, ECP

RFC 1968

Itt lehetne külön megbeszélni az adatcsomagok titkosításához használt módszerrel kapcsolatos paramétereket. Ha lehetne mást is használni, mint az MPPE, melyet már a CCP-n belül letisztáztunk.

Windows rendszerekben nem nagyon fogunk találkozni vele. Ergo ebben a könyvben sem.

3.3.1.5 PPP OVER ETHERNET

RFC 2516

Na, ez meg milyen csodabogár?

Oké, tudom, hogy költői a kérdés, hiszen rengeteten használjuk ezt a módszert, de - különösen az eddigiek alapján - nem lepődnék meg, ha sokan bizonytalanságot éreznének az erőben.

Hiszen kitárgyaltuk az Ethernet technológiát, láttuk, hogy a PPP egy teljesen más világ... akkor hogyan jön össze ez a kettő? És legfőképpen, miért?

A választ az ISP-k környékén kell keresgélni. Ezek a cégek ugyanis azt szeretik, ha az előfizetőik PPP kapcsolaton keresztül jönnek be. Hiszen ekkor a kapcsolat remekül autentikálható, minden mérhető, minden kézbentartható, magának a kapcsolatnak a szintjén. Lehet forgalmat szabályozni, lehet forgalom alapján számlázni... kánaán.
De mit szeretne az ügyfél? Hát azt, hogy ne bonyolítsák túl az életét, hadd lógjon ő csak egy mezei Ethernet hálózaton. Az egyszerű a szép.

Ebből született meg ez az öszvér.

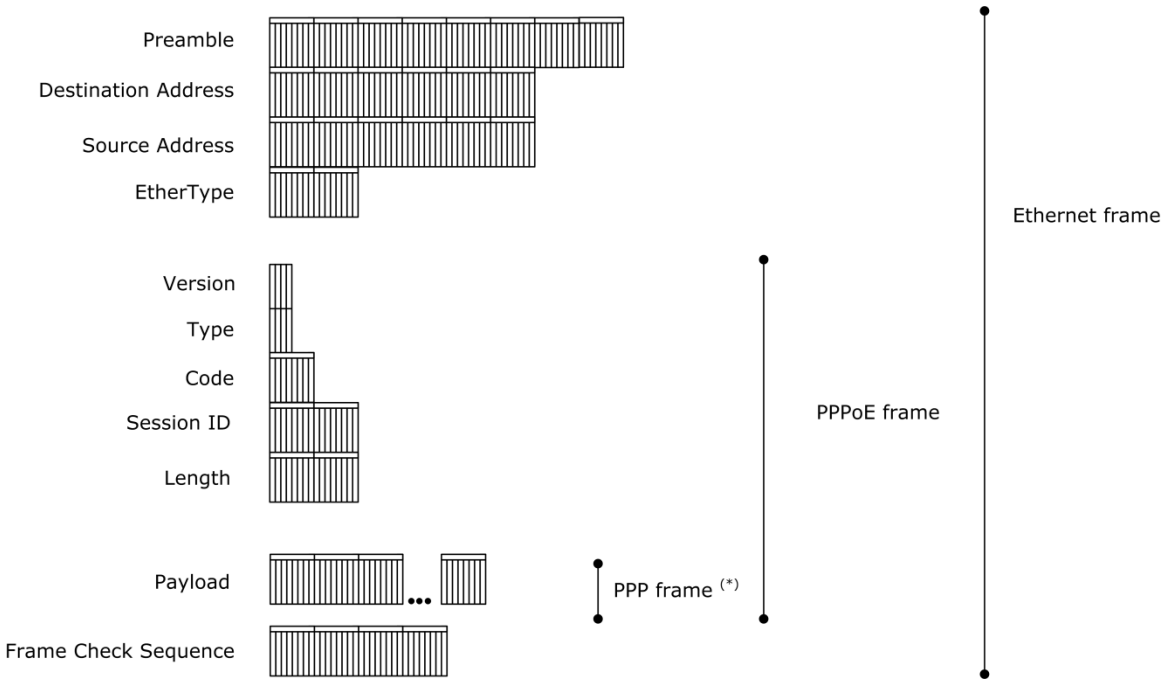
Mint a neve is mutatja, alapvetően Ethernet keretekről lesz szó, csak éppen a Payload-ban PPPoE keretek fognak utazni, azok payloadjában pedig vagy PPPoE információk (Discovery fázis), vagy PPP keretek (PPP Session fázis). Magyarul Ethernetbe kapszulázzuk a PPPoE-t, abba meg a PPP-t.[28]

Ahogy jeleztem, a kommunikációnak két fázisa létezik:

- Discovery Phase: a kliens node felfedezi az Access Concentrator-t, aki egy AAA szerver[29] segítségével autentikálja.
- PPP Session Phase: létrejön a PPP session, beindult a kommunikáció.

Valahogy így néz ki egy keret.

[28] Mielőtt dühös levelet fogalmaznál Dr. Grétsy Lászlónak, megkérdezném: az iróniaérzékelőd működik? Amennyiben nem, akkor javaslom, hogy kapcsold be. Az egész könyv során szükséged lesz rá.
[29] AAA: Authentication, Authorization, Accounting. Azaz kiforgatjuk a gatyájából a delikvenst.



3.29. ÁBRA PPPOE KERET

VERSION: 4 bites mező, az értéke mint a totóban: fix 1-es.

TYPE: Dettó.

CODE: A Discovery fázis során különböző tartalmú csomagok cserélnek gazdát. Ez a mező hivatott arra, hogy beazonosítsa a tipusokat.
A PPP session fázisban az értéke 0.

SESSION ID: A Discovery fázisban az értéke 0. Utána pedig a kialakult session azonosítója.

LENGTH: A PPPoE payload mérete.

PAYLOAD: Meglehetősen rugalmas mező. A Discovery fázisban az adott lépéshez szükséges PPPoE információk utaznak itt. A PPP Session fázisban viszont már PPP keretek. (Ezt jelezné az a csillag ott a képen.)

3.3.1.5.1 PPPoE Discovery - szigorúan kockafejűeknek

Nem mintha más is eljutott volna idáig a könyv olvasásában.

Ahhoz, hogy a kliens és az Access Concentrator egymásra találjanak, egy közepesen bonyolult párostáncot kell lejteniük: PADI, PADO, PADR... PADS. Mint egy barokk pavane.

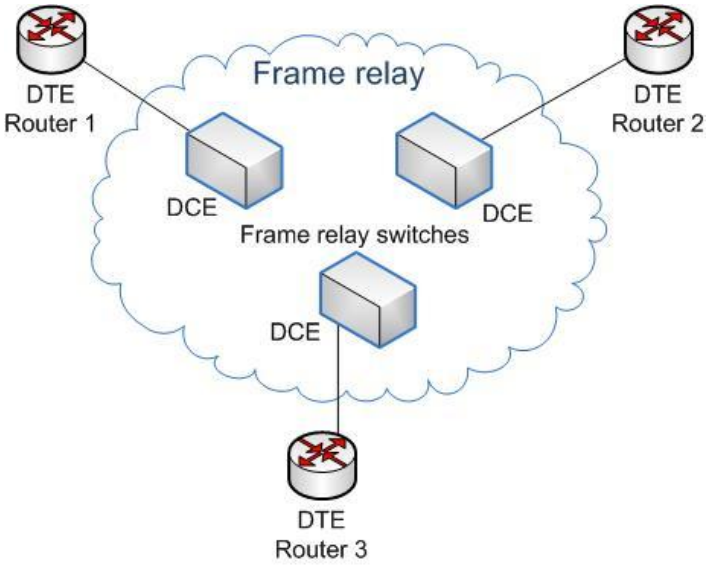
- PPPoE Active Discovery Initiation (PADI): A kliens küldi... bele a nagyvilágba. Ebből következően az üzenet az egy broadcast. (Csupa magas bit.) A CODE értéke 9, a SESSION ID értéke 0.
- PPPoE Active Discovery Offer (PADO): Az AC küldi vissza. Ez már unicast üzenet, a kliens MAC címére. A CODE értéke 7, a SESSION ID értéke 0. A csomag tartalmazza az AC nevét, és a Service-Name értéket.
- PPPoE Active Discovery Request (PADR): A kliens küldi vissza. Unicast üzenet az AC MAC címére. A CODE értéke 25, a SESSION ID értéke 0. A csomag tartalmazza a Service-Name értéket.
- PPPoE Active Discovery Session-confirmation (PADS): Az AC küldi vissza a kliensnek. Unicast üzenet a kliens MAC címére. A CODE értéke 101 - és itt jön a lényeg: a SESSION ID értéke már a végleges session azonosító. A Service-name természetesen itt is benne van a csomagban.

3.3.2 NBMA: FRAME RELAY

Most egy kicsit tágítjuk a látókörünket. A PPP olyan WAN kapcsolatot jelentett, mely két node között jött létre. Mi lenne, ha hasonló WAN technikát használnánk... de több node is részt vehetne a buliban?

Ekkor beszélünk Non-Broadcast Multiple Access (NBMA) linkekről. Mint a neve is mutatja, ebben az esetben egy linken több node is megtalálható, de a linken nem lehet broadcast. Legismertebb képviselői ennek a technológiának az X.25, illetve a Frame Relay.
Az X.25 valamikor igen elterjedt volt, mert szakadozó vonalakon is jól teljesített. Ma már a vonalak sokkal megbízhatóbbak, így már zavaró az X.25-ben a túl nagy kommunikációs vízfej. Gyakorlatilag a jóval kevesebb ellenőrzést tartalmazó, emiatt persze sokkal gyorsabb Frame Relay ki is szorította. A Windows Server 2008 spec már csak az utóbbit támogatja.

Leginkább hálózatok összekapcsolására használják. Fontos jellegzetessége, hogy az egyes node-ok felé a számlázás az átvitt adatmennyiség alapján történik.



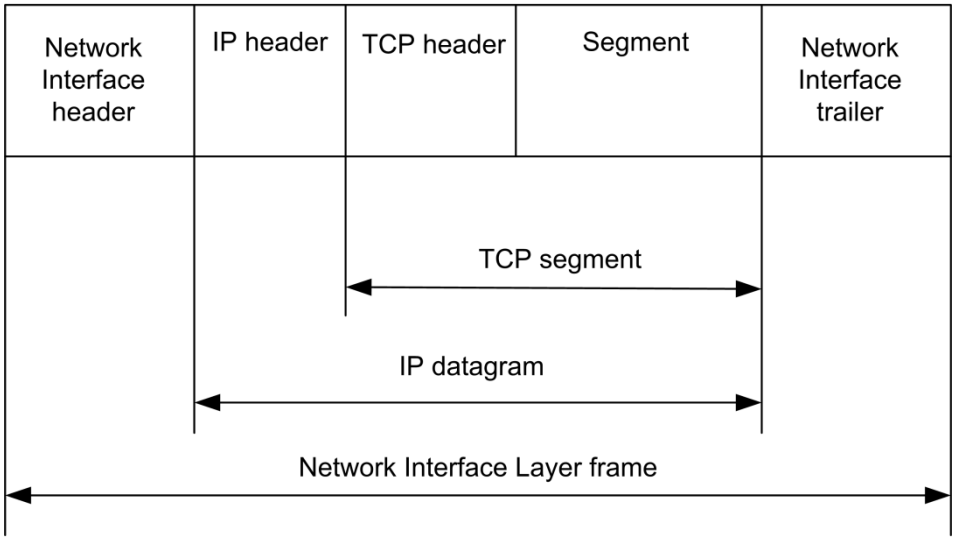
3.30. ÁBRA FRAME RELAY (FORRÁS: WIKIPEDIA)

A Frame Relay működésének kivesézése meghaladja e könyv kereteit.

Az alábbi linkeken kimerítő leírást találhatunk róla:
http://www.protocols.com/pbook/frame.htm
http://en.wikipedia.org/wiki/Frame_Relay
http://www.szabilinux.hu/trendek/trendek422.html

4 AZ INTERNET RÉTEG PROTOKOLLJAI

De még mielőtt továbbmennénk, foglaljuk össze egy ábrán, miről is beszéltünk eddig, illetve mi várható a továbbiakban.



4.1. ÁBRA EGY HÁLÓZATI CSOMAG FELÉPÍTÉSE

Az alfa és az omega. Látjuk, hogy a Network Interface rétegben megismert keret tényleg bekeretezi a csomagot, azon belül pedig a payload maga az IP datagram, mely áll egy IP fejlécből és egy IP payloadból... az utóbbi persze nem más, mint egy TCP szegmens, melynek szintén van fejléce éa hasznos tartalma.

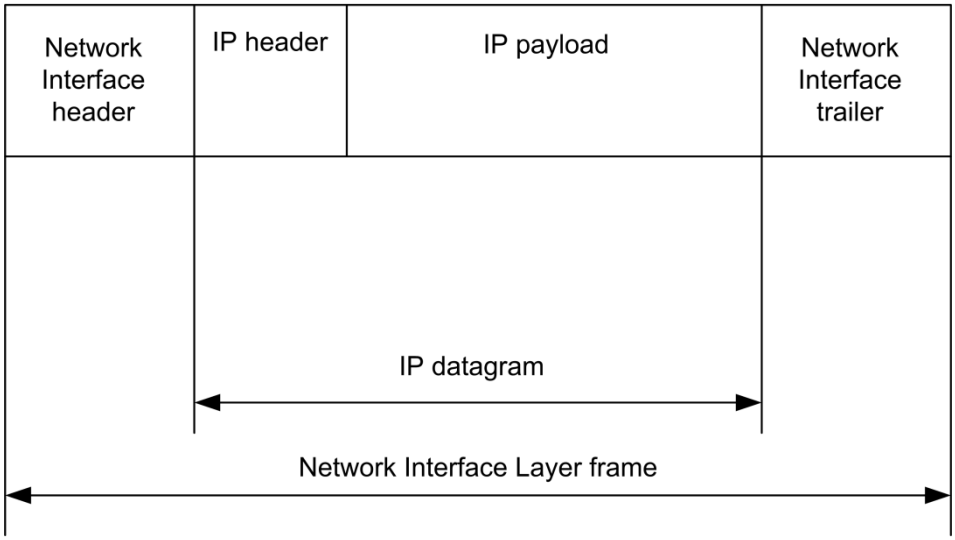
Így néznek ki a hagyma héjai... azaz a hálózati modell rétegei.
Csomagcentrikus megközelítésben.

4.1 IPv4

RFC 791

4.1.1 Az IP Header

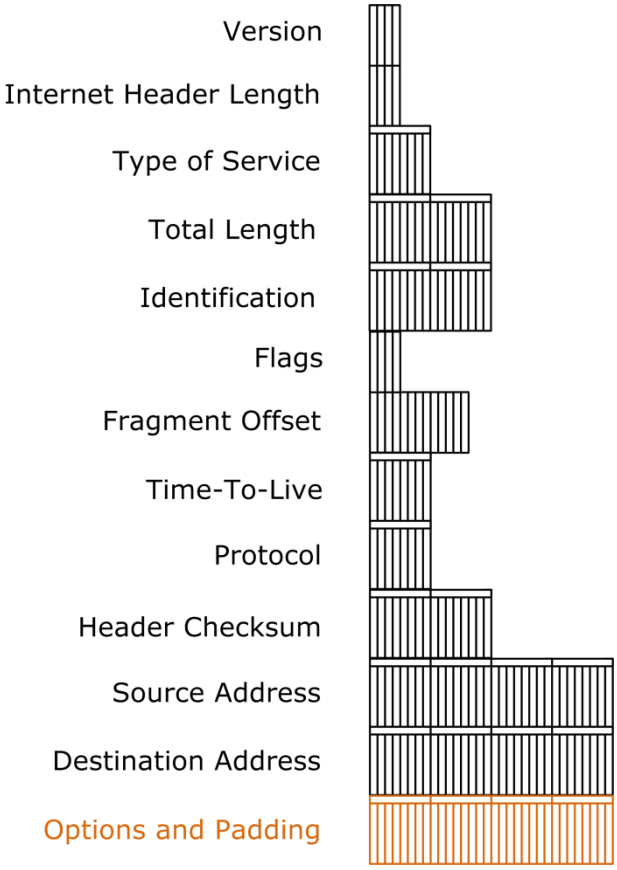
Első körben az IP datagrammal fogunk foglalkozni. Az előző ábrát most leegyszerűsítettem: elfelejtjük, hogy egyáltalán hallottunk olyasmiről, hogy TCP szegmens. Nincs. Nem létezik. Csak egy zárt doboz van (IP payload), melyet az IP csomagba (IP Datagram) raktak, és nekünk - illetve szerencsére a szabad elektronoknak - ezeket kell elszállítaniuk valahová.



4.2. ábra Egy IP csomag felépítése

A címből is látszik, hogy ez a fejezet az IP fejlécről fog szólni... a későbbiekben pedig megnézzük, mi minden lehet még az IP payload - az egyébként ugye nem létező TCP szegmensen kívül. (Súgok: ICMP, IGMP és még sokan mások.)

Csapjunk is egyből a lecsóba.



4.3. ÁBRA AZ IP FEJLÉC

VERSION:Az IP verziószáma. Jelenleg csak a 4-es, illetve a 6-os érték értelmezett.

INTERNET HEADER LENGTH: Mint az ábrán is látható, ez egy négy bites mező. Itt tárolódik, hogy mekkora is az IP fejléc.
Számoljunk egy kicsit. 4 bit, az annyi, mint 15 darab nullától különböző érték. Mit lehet ennyi helyen tárolni? 15 bájtot? Dehát az ábra alapján minimum 20 bájtos a fejléc és akkor még nem is beszéltünk az opcionális OPTIONS AND PADDING mezőkről. Akkor?
Nos, itt az ún szószámláló értékét tárolják. A szó, mint tudjuk, elszáll... másrészt 4 bájt méretű. Mivel az INTERNET HEADER LENGTH maximális értéke 15, így a fejléc maximális mérete 15*4=60 bájt. Minimálisan pedig 20 bájt, azaz a mező értéke nem lehet 5-nél kisebb.

És most akkor engedjünk szabad utat a morgolódásnak. Ha ragaszkodunk a 60-as felső korláthoz, akkor hány biten is tárolhatunk 63 különböző értéket? Igen, hat. Tehát ezzel a kavarással spóroltunk 2, azaz kettő bitet. Viszont minden ki/becsomagolásnál beraktunk egy plusz műveletet. Mindegyiknél. Ráadásul bejött még egy megkötés, az OPTIONS AND PADDING mezőknek, függetlenül attól, hogy milyen opciót, azon belül milyen értékeket tartalmaznak, mindig 4 bájtosnak kell

lenniük. Azaz ha van egy kétbájtos IP opció, akkor a két bit spóroláson buktunk két bájtot, mert üres tölteléket - padding-ot - kell tenni a végére. Ráadásul mindezt egy olyan rendszerben, mely alapból borzalmasan pazarló, gondoljál csak a korábbi fejezetben megismert SNAP átalakításokra, ahol tök feleslegesen tárolunk le konstans értékeket.
Dehát... ez van, ezt kell megtanulnunk.

TYPE OF SERVICE (TOS): Itt kellemetlen választásra kényszerültem. Egyfelől ez egy nagyon jó, szószerint bitre lebontott struktúrával rendelkező 1 bájtos mező. Volt.

RFC 791

Az RFC 791 szerint. Ezt például hálás lenne részletesen is leírni, egyszerű, érthető. Csakhogy ma már a kutya sem használja.

```
⊞ Frame 33 (68 bytes on wire, 68 bytes captured)
⊞ Ethernet II, Src: AsustekC_ab:37:2e (00:1e:8c:ab:37:2e), Dst: Cisco-Li_f1:34:0a (00:18:f8:f1:34:0a)
⊟ Internet Protocol, Src: 192.168.1.100 (192.168.1.100), Dst: 84.2.44.1 (84.2.44.1)
    Version: 4
    Header length: 20 bytes
  ⊟ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
      0000 00.. = Differentiated Services Codepoint: Default (0x00)
      .... ..0. = ECN-Capable Transport (ECT): 0
      .... ...0 = ECN-CE: 0
    Total Length: 54
    Identification: 0x173d (5949)
  ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 128
    Protocol: UDP (0x11)
  ⊞ Header checksum: 0xe16a [correct]
    Source: 192.168.1.100 (192.168.1.100)
    Destination: 84.2.44.1 (84.2.44.1)
⊞ User Datagram Protocol, Src Port: 51684 (51684), Dst Port: domain (53)
⊞ Domain Name System (query)
```

4.4. ÁBRA MIT IS TALÁLUNK A TOS HELYÉN?

Legalábbis sem az XP SP3, sem a Vista, sem a Windows Server 2008 nem a 791-es RFC szerint értelmezi ezt a mezőt... és van egy olyan halvány sejtésem, hogy a Windows 7 sem úgy fogja.

RFC 2474

Ehelyett az RFC 2474 a menő. Mondanom sem kell, bonyolult és érthetetlen. Pedig jól indul: azt mondja, dobjuk ki az utolsó két bitet. Nem köll. A maradék hat bitet pedig elnevezzük DSCP-nek, azaz Differentiated Services Code Point-nak. (Lsd. a fenti ábra.) Utána viszont nagyon ködös lesz. Az RFC-ből annyit ki tudtam hámozni, hogy a hat bitben lévő szervízkódok azt határozzák meg, hogy a csomagok milyen elbánásban részesüljenek, azaz gyakorlatilag egyfajta sávszélesség-kezelésre (QOS) használhatók, méghozzá blokk alapú szervezettségben. Ezeket a blokkokat

Differentiated Services (DS) domain-eknek nevezik. De ott már eltört az agyam, amikor az RFC azt mondta, hogy az ún. DS-compliant hálózatokban a kódok kiosztását és az az alapján történő viselkedést házirendek határozzák meg, melyek ismertetése nem része az RFC-nek.
A Windows annyit tesz ehhez, hogy használj Group Policy-t, vagy ha szereted a kihívásokat, van egy csomó API[30].

Még egy dolgot tisztáznunk kell. Én azt mondtam korábban, hogy az utolsó két bitet kidobtuk - az ábrán viszont az szerepel, miszerint az egyik bit ECN-Capable Transport (ECT), a másik pedig ECN-CE. Ezek bizony nem tűnnek használaton kívüli biteknek.
Nem is azok.

RFC 3168

A szabványalkotók rácsaptak a két üres bitre, mint gyöngytyúk a takonyra és az RFC 3168-ban az úgynevezett Explicit Congestion Notification (ECN) funkció megvalósítására használták fel. Miről is van itt szó? Arról, hogy a routerek, ha elárasztják őket, utolsó segélykiáltással jelezhetik, hogy 'elvtársak, ne lőjetek!'. Ez az ECN. Ha a feladó host olyan, hogy képes venni ezt az adást, akkor leengedi a fegyvert. Ha nem, akkor rezzenéstelen arccal tüzel tovább.
Lehetséges értékek:

- 00 : A feladó kegyetlen és süket.
- 01 v. 10 : A feladó egy érző lélek.
- 11 : A router ólommérgezésközeli állapotban van.

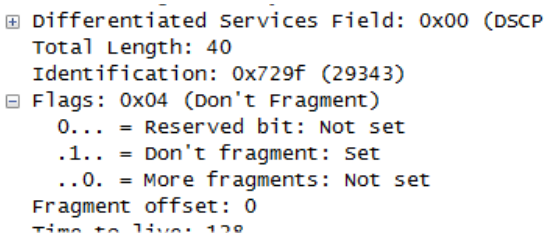
Mint látható (*4.4. ábra Mit is találunk a TOS helyén?*), az én számítógépem jelenleg süket bérgyilkos - ez ugyanis az alapértelmezés. De a netsh paranccsal át lehet állítani a mentalitását, méghozzá hálózati kártyánként.

TOTAL LENGTH: A teljes IP datagram méretét jelzi, bájtban. Mivel a mező kétbájtos, így az IP datagram maximális mérete 65535 byte lesz[31]. Mi van, ha ennél nagyobb a csomag mérete? Tördelünk.

IDENTIFICATION: Ha már tördelnünk kell, akkor valahogy jeleznünk is kell, mely töredékek tartoznak ugyanahhoz az IP datagramhoz. Ezért számozzuk a csomagokat.

FLAGS: Igen, még mindig tördelünk.

[30] Generic QoS (GQoS) and Traffic Control (TC) API, vagy Quality Windows Audio-Video Experience (qWAVE), azaz QoS2 API.
[31] Rémlik valakinek a gyilkos ping?



4.5. ÁBRA FLAGS

1. Az első bit használaton kívüli.
2. A második bit azt jelzi, hogy tördelhető-e a csomag? Ha az értéke egy, akkor nem.
3. A harmadik bit azt mutatja, hogy a mostani IP csomag az utolsó-e, vagy jön még töredék. Ha az értéke 0, akkor ez az utolsó.

Az ábrán 0100 látható, azaz decimálisan 4. (Igen, a mező négy bitet foglal el, de csak három bír értelmezhető értékkel: ez a középső kettő. Nem, nem én vagyok a hülye.)

FRAGMENT OFFSET: Ha már tördeltünk, akkor ez az érték mutatja meg, hogy a konkrét csomag hanyadik töredék éppen.

Jelzem, hogy a tördelésről - fragmentation - később bővebben is lesz szó. Egyelőre legyen elég annyi, hogy ellenezzük.

TIME-TO-LIVE: Azaz van-e élet a halál előtt? És ha van, akkor mennyi? A mezőben lévő érték ugyanis azt mutatja, hogy meddig él a csomag. Ha lejárt, akkor az a hálózati eszköz, amelyiknél az eset bekövetkezett, eldobja a csomagot.
Vajon mi lehet a mértékegysége? Óra? Secundum? Tick?

RFC 1812

Egyik sem. Valamikor tényleg idő dimenziójú volt, de az RFC 1812 óta inkább számláló, melynek értéke minden hopnál - azaz hálózati eszközön történő áthaladásnál - csökken egyet. Logikusan, ha küldünk egy csomagot, amelynek a TTL értéke 0, akkor az nem fog kimenni a subnetről. Ha a TTL értéke 1, akkor a subnetről kimegyünk ugyan, de maximum csak a szomszéd alhálózatra.

PROTOCOL: Azt mondja meg, hogy az IP Payload-ban milyen protokollhoz tartozó csomag utazik. Néhány példa:

4.1. TÁBLÁZAT

Érték	Protokoll
1	ICMP
2	IGMP
6	TCP
17	UDP
41	IPv6
47	GRE
50	ESP
51	AH

HEADER CHECKSUM: Tulajdonképpen egy CRC, de csak a fejléc integritását biztosítja.

Beugratós kérdés: a feladáskori érték vajon megegyezik-e az érkezési értékkel?

Jaj, hát hogyan is egyezne meg? Épp most mondtam, hogy a TTL értékét minden hop lecsökkenti. Nyilván minden fejlécbeli módosításkor újra kell számolni az ellenőrző értéket.

SOURCE ADDRESS: A feladó IP címe... feltéve, hogy a NAT nem rondít bele.

DESTINATION ADDRESS: A célállomás IP címe. Feltéve, hogy a NAT nem rondít bele.

OPTIONS AND PADDING: Jó megfigyelőképességgel rendelkező olvasók észrevehették, hogy a korábbi ábrán (*4.3. ábra Az IP fejléc*) ez a mező narancssárga. Ezzel finoman arra próbáltam célozni, hogy ez a mező teljesen opcionális. Mire is jó? Van egy csomó kisérő információ, melyek bizonyos esetekben jól jöhetnek a hálózati eszközök számára. A kisérő információk tipusokra vannak osztva, mindegyiknek kódja van - és adatstruktúja, természetesen. Ezeket úgy hívják, hogy IP Options, azaz IP opciók. (Később bővebben is fogunk velük foglalkozni.)
Ha nem használjuk ezt a mezőt, akkor az IP fejléc mérete 20 bájt. Ha használjuk, akkor a LENGTH mezőnél tárgyaltak alapján már tudjuk, hogy a fejléc mérete maximum 60 bájt lehet, illetve néggyel oszthatónak kell lennie. Az égegyadta világon semmi sem garantálja, hogy egy konkrét IP opció adatszerkezete pont néggyel osztható számú bájtból álljon - ergo ilyenkor értéktelen törmelékkel (padding) fel kell tölteni a mezőt.

4.1.1.1 TÖREDEZETTSÉG

Habár már esett róla szó, de nem árt pontosan letisztázni: mi is pontosan az a bizonyos MTU? Ha szigorúan nézzük, akkor elegendő kibontani az akronimot: Maximum Transmission Unit, szószerint fordítva: a legnagyobb elszállítható egység. Persze így túl sok értelme nincs, ha értelem szerint fordítunk, akkor azt mondhatjuk, hogy a legnagyobb payload mérete, melyet még egy csomagon belül el tudunk szállítani.

MTU:
http://en.wikipedia.org/wiki/Maximum_transmission_unit

Ez az a pont, ahol nem nevezhetjük csak úgy csomagnak a csomagunkat.
Miért is nem mindegy? Mert láthattuk az ábrán (*4.1. ábra Egy hálózati csomag felépítése*), hogy csak nézőpont kérdése, mit is nevezünk payload-nak. Ami egyik szintről payload, belenézve további header és payload. Ezért fontos tisztázni, hogy az MTU-t a Network Interface Layer viszonylatában értelmezzük - azaz az MTU a NIL payload maximális mérete.[32]

Jó. Akkor mekkora is az MTU?

Attól függ. Lapozz vissza a NIL fejezethez. Azt fogod találni, hogy csomagtovábbítási technológiától függően más és más. Mi fog történni, ha a csomagunk nagyobb MTU-val rendelkező alhálózatról kisebb MTU-val rendelkező alhálózatra kerül? Muszáj átpréselni a nagy szekrényt a kis ajtón: kisebb darabokra vagdossuk a csomagot - azaz fragmentálunk. Kinek fog ez leghamarabb fájni? Úgy van, az Internet rétegnek - ergo ebben a rétegben kell megoldanunk a fragmentálás adminisztrációját, méghozzá értelemszerűen az IP header-ben.

Helyben vagyunk. Erről fog szólni ez a fejezet.

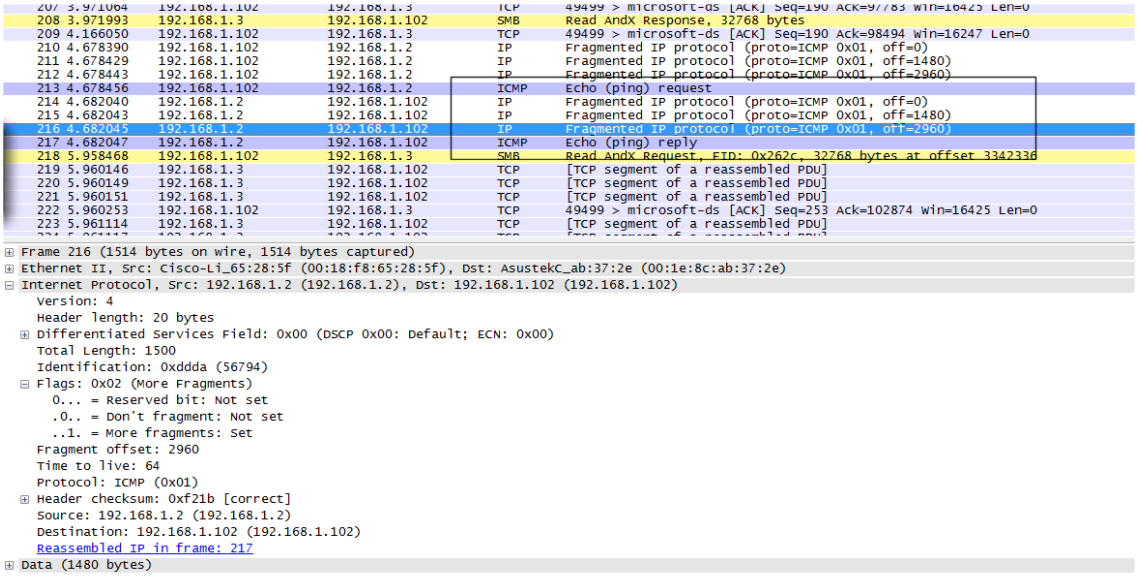
[32] Megjegyzem, ezen a téren nem igazán egységes a szakirodalom. Van ahol azt mondják, hogy a ***konkrét rétegre*** jellemző MTU az a konkrét réteghez tartozó maximális payload mérete. A gyakorlatban viszont inkább az terjedt el, hogy ***a hálózati kártyának*** van MTU-ja - mely felfogás ebben az esetben a fenti, a NIL rétegre vonatkozó definíciót jelenti.

Mit is tudunk eddig? Elég sokat. Ha fragmentálódott a csomagunk, akkor

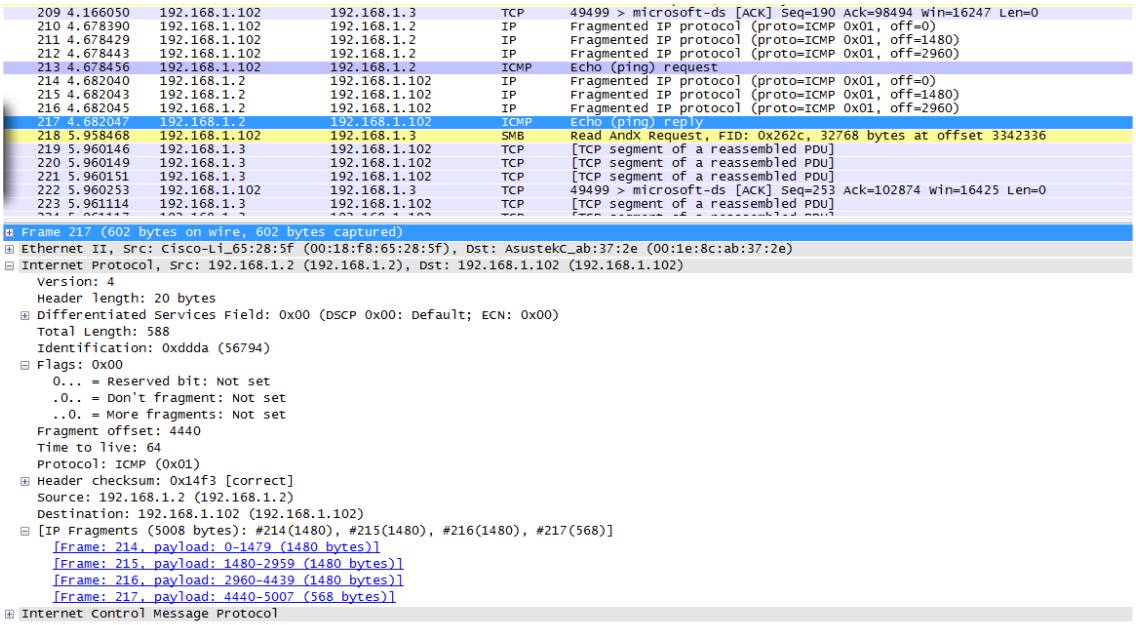
- Az összetartozó, azaz ugyannak a csomagnak a darabjait az IDENTIFICATION mező azonosítja be.
- A darabok sorrendjét a FRAGMENT OFFSET mező határozza meg.
- A szélső töredékeket a MORE FRAGMENTS flag azonosítja.

Akinek ennyi elég is, az nyugodtan átugorhatja a következő pár oldalt. Ugyanis itt fogom kifejteni, hogyan is történnek pontosan ezek a dolgok.

Először is vizsgáljuk át alaposan az alábbi két ábrát.



4.6. ÁBRA FRAGMENTÁLT CSOMAG HARMADIK TÖREDÉKE



4.7. ÁBRA FRAGMENTÁLT CSOMAG UTOLSÓ TÖREDÉKE

Mielőtt darabokra cincálnánk az ábrát, egy gyors kérdés: vajon hogyan lehet a legegyszerűbben töredezett csomagokat produkálni? Úgy van, a varázslatos ping paranccsal. A parancsnak ugyanis van egy olyan paramétere, mely a payload méretét adja meg - egyszerűen nagyobbra kell választanunk az értéket, mint az aktuális MTU értékünk. A fenti ábrák is így keletkeztek: kiadtam a *ping -l 5000 192.168.1.2* parancsot.
Bónusz feladat: aki az ábrák alapján megmondja, mennyi az MTU-m értéke, az kap egy nyalókát. De tessék igyekezni, mert a következő oldalakon elárulom a megoldást.

Akkor kezdjük az első ábrával. Van benne egy ronda téglalap: ez mutatja azt, hogy mely keretekről van egyáltalán szó: a 213-as a kérés, 214-217 között jön vissza a töredezett válasz. Magán az első ábrán a harmadik töredéket láthatjuk, a második ábrán pedig a negyedik, azaz utolsó csomagot.
Akkor most egy ideig nem is szólnék semmit, mindenkit megkérek, hogy az eddigi információk alapján nézze át alaposan a képeket.

.
néz
.
.
még mindig néz
.
.

Gondolom, megvolt. Akkor nézzük át együtt. Mindkét ábrán látható, hogy a csomagban az IDENTIFICATION mező értéke 0xddda. Tehát ugyanahhoz az eredeti csomaghoz tartoznak. Az első képen a FLAGS mező bitjei jelzik, hogy a csomag nem nem (*sic*) töredezett (don't fragment not set), azaz töredezett - és ezen belül még több csomag is van (more fragments set), azaz ez nem az utolsó adag. Az utolsó töredéknél a more fragments értéke not set, azaz jelzi, hogy nincs már több töredék ebből a (0xddda) csomagból.
Nézzük akkor még a FRAGMENT OFFSET értéket: a harmadik csomagnál 2960, a negyedik csomagnál pedig 4440. Itt már azért kellene jócskán matekozni, ha a Wireshark nem lenne olyan kedves és nem adna pontosabb infókat is.
Először például a felső táblán már a keret összefoglaló sorában is láthatjuk az egyes offset értékeket. (Ha esetleg ismeretlen lenne a fogalom: offset alatt mindig valamilyen értékhez képesti elcsúszást szoktunk érteni.) Az első keretnél az offset nulla, azaz a keret összerakáskor a 0. pozícióból fog indulni. A második keret értéke 1480, azaz a keret a képzeletbeli számegyenesen az 1480. pozícióból indul, azaz az előző az 1479. pozícióban végződik. A harmadik keret offset értéke 2960, a negyediknél ugyan nem látjuk a fősorban, de ha ránézünk az alatta lévő ablak aljára, meglehetősen világosan és egyértelműen látjuk, melyik csomag mettől meddig tartó részét képezi az eredeti széttördelt csomagnak.

Oké. Már csak egy kérdés tisztázása maradt hátra: mennyi is akkor az MTU?

Nyilván első körben azt mondanánk, hogy 1480. Hiszen ekkora darabokból áll össze a végső csomag.
De.

Nézzük meg alaposabban az ábrákat. Látunk IP OPTIONS mezőt? Nem. Akkor mekkora lehet a fejléc? 20 bájt. Gyorsan csekkoljuk le, nézzük meg a HEADER LENGTH mező értékét: ott is azt látjuk, hogy '20 bytes'.
Tehát IP csomagon belül a fejléc 20 bájt, a payload pedig 1480 bájt. Ugyanezt összeg formájában is láthatjuk, ha ránézünk a harmadik csomag TOTAL LENGTH mezőjének értékére.
Csakhogy, hogyan is definiáltuk az MTU-t? Úgy, hogy az a ***NIL kereten*** belül a payload - de ez viszont a teljes IP datagram, azaz 1500 bájt. Ő az MTU.

Egy dolgon lehet még elgondolkodni: mi is van akkor a fejlécekkel? Hiszen a végső csomagnak csak egy IP header-e van, ezzel szemben a négy töredéknek bizony négy. Nem vesz el ez értékes biteket a keretekből?
De, elvesz.
Amikor össze kell raknunk a csomagot, akkor kapunk 3 darab 1480 bájtos és 1 darab 568 bájtos payload tartalmat, illetve 4 darab 20 bájtos fejlécet. A négy fejléc alapján legyártjuk az eredeti csomag 20 bájtos fejlécét - 60 bájt információ pedig megy a kukába. Aztán az eredeti payload már csak a töredékek 1480 bájtos payload darabkáiból, illetve az utolsó töredék maradék payload darabkájából lesz összerakva.

Most képzeljük el a következőt: szegény csomagunkat rossz sorsa olyan útvonalra vezeti, ahol az egyik alhálózat MTU értéke 4460, a következő alhálózaton ugyanez 2662, majd a harmadik alhálózaton 1500. Mi történik ilyenkor? Lehet-e a töredezett csomagokat tovább tördelni? Le tudja-e ezt kezelni az IP réteg?

Igen, le tudja.
Csak nem szereti.
Mi sem.

Szerencsére ma már az ilyesmi nagyon ritka, nem is nagyon mennék bele a konkrét megvalósítás boncolgatásába.

Mindenesetre ha teljesen rejtélyesen, kiszámíthatatlanul viselkedő hálózati kommunikációval találkozol, mely hol jó, hol nem - akkor először mindig az MTU környékén érdemes szétnézned.

4.1.1.2 IP OPTIONS

Mint korábban is írtam, ez egy opcionális mező. Ha nincs, úgy is jó. Ha van, úgy is jó. De, mint látni fogjuk, itt főleg olyan információk utaznak, melyeknek túl sok közük nincs az IP payloadhoz - itt inkább a kommunikáció komponensei üzengetnek egymásnak, jórészt tesztelési célzattal.

Ezek az üzenetek 1 és 40 bájt között tetszőleges méretűek lehetnek - de emlékszünk,[33] a méretnek néggyel oszthatónak kell lennie. Hány IP Option lehet egy IP fejlécben? Amennyi belefér. Majd meglátjuk. Hogyan kell szétosztani az egyes IP Option csomagokat, ha fragmentálódott az IP datagramm? Attól függ. Majd meglátjuk.

Akkor lássuk meg.

OPTION CODE: 1 bájt, vezérlőinformációk. Ennek a részei:

COPY: Balról az első bit. Töredezett IP datagramok esetén van szerepe. Amennyiben az értéke 0, akkor az IP Option csomagok csak az első töredékbe kerülnek bele. Ha 1, akkor mindegyikbe.

OPTION CLASS: Balról a második és a harmadik bit.

4.2. TÁBLÁZAT

Option Class binárisan	Option Class decimálisan	Leírás
00	0	Hálózati kontroll
01	1	Majd csak jó lesz valamire
10	2	Nyomozás, mérés
11	3	Majd csak jó lesz valamire

OPTION NUMBER: A maradék öt bit egy azonosító kód. Ez mondja meg, hogy ki is pontosan a konkrét IP Option.

4.3. TÁBLÁZAT

Copy bit	Option Class	Option number	Leírás
0	00 (0)	00000 (0)	End Of Option List
0	00 (0)	00001 (1)	No Operation
1	00 (0)	00011 (3)	Loose Source Routing
0	00 (0)	00111 (7)	Record Route
1	00 (0)	01001 (9)	Strict Source Routing
1	00 (0)	10100 (20)	IP Router Alert
0	10 (2)	00100 (4)	Internet Timestamp

[33] Tulajdonképpen most írom le harmadszor, tehát lassan már muszáj lesz.

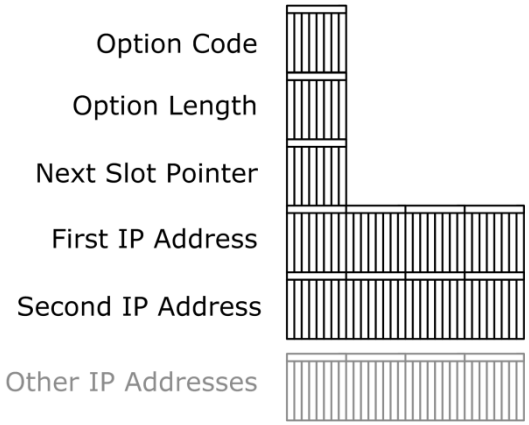
Nem csak a fenti táblázatban lévő IP opciók léteznek.

A teljes lista itt található meg:
http://www.iana.org/assignments/ip-parameters

4.1.1.2.1 END OF OPTION LIST, NO OPERATION

A két IP opciót nyugodtan kezelhetjük együtt. Mind a kettőnek egy bájtnyi az értéke, ez gyakorlatilag az OPTION CODE. Az első értéke 0, a másodiké 1. A NO OPERATION bájt (1) választja el egymástól a különböző IP Option csomagokat, az END OF THE OPTION LIST bájt (0) pedig lezárja az utolsót.

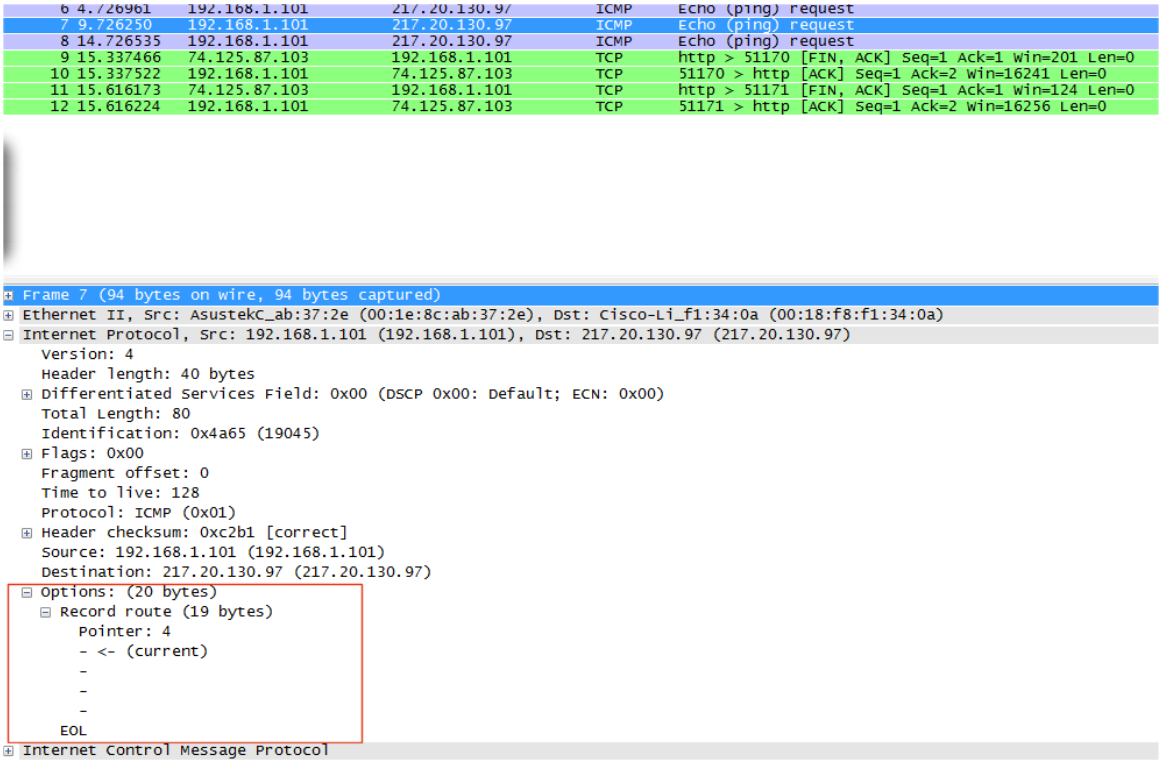
4.1.1.2.2 RECORD ROUTE



4.8. ÁBRA RECORD ROUTE

Itt nincs túl nehéz dolgom: egyszerűen csak le kell fordítanom a nevet.
Record route -> feljegyezzük az útvonalat. Azaz megy a csomag hop-ról hop-ra, Az ugrás sorszáma kerül bele a Next Slot Pointer mezőbe[34], a router IP címe pedig a soron következő IP Address mezőbe.

[34] Pontosabban sorszám*4, mert egy IP cím 4 bájt, és így kapjuk meg azt a méretnövekményt, melyet majd össze kell hasonlítanunk az Option Length mező értékével ahhoz, hogy eldönthessük, egyáltalán rögzíthetjük-e még az IP címeket, vagy már túlcsordultunk.



4.9. ÁBRA RECORD ROUTE IP OPTION

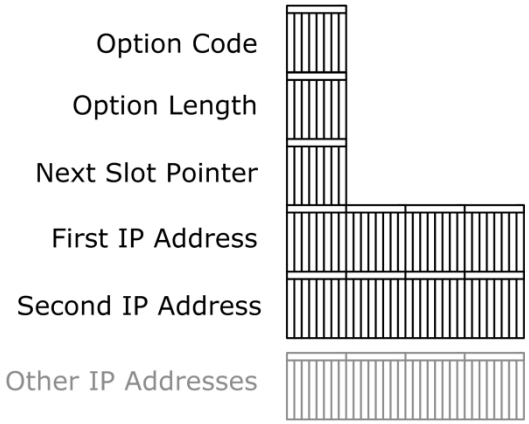
Az ábrán egy kikényszerített Record Route látható. Úgy vettem erőszakot a rendszeren, hogy beírtam a *ping -r 4 index.hu* parancsot. Jelen esetben a 4-es szám az Option Length paraméter értéke - ez jelenti azt, hogy maximum 4 IP címet jegyezhetek fel. Látszik is szépen az IP Option kikalkulált mérete: 4*4 bájt az IP címeknek, 1 bájt az Option Code (az értéke ugye 7), 1 bájt az Option Length és 1 bájt a Next Slot Pointer - azaz összesen 19 bájt. (Nyilván a néggyel oszthatóság miatt lesz 1 bájt padding is, jelen esetben értelemszerűen egy End of Option List mező.)
Azt is tudjuk, hogy az IP fejléc maximális mérete 60 bájt, tehát maximum 9 IP címet tudunk rögzíteni. Ha több esik útba, akkor úgy jártunk. (De legalább megspóroltunk pár bitet az Internet Header Length mezőnél.)

Akkor most már csak az a kérdés, hogy miért nem látunk semmit a csomagban?
Nos, semmi garancia sincs arra, hogy mind az én hálózatomban, mind a vad interneten barátságos routereket fogok találni. Nézzük csak meg még egyszer az ábrát? Egy csomó echo request, de nincs egyetlen echo reply sem. Valójában tényleg ez is történt: a ping timeout-ra futott - azaz a routerek nem voltak hajlandóak együttműködni.

> Egy fontos megjegyzés: ha esetleg arra gondolnál, hogy a tracert is ilyen módon szedi össze egy útvonal állomásait, akkor tévedsz. Az ugyanis mindig eggyel nagyobb TTL értékű pinggel próbálkozik.

4.1.1.2.3 STRICT SOURCE ROUTING / LOOSE SOURCE ROUTING

Az erős emberek opciója. Megmondjuk annak a nyomorult IP csomagnak, hogy konkrétan milyen útvonalon jusson el a célállomáshoz. Méghozzá a STRICT esetben routerről routerre előírjuk az útvonalat, míg a LOOSE esetben csak azokat a routereket adjuk meg, melyeken át kell haladnia a csomagnak - hogy közben merre bóklászik, az az ő dolga.
Jogos lehet a kérdés: ennek meg ugyan mi értelme van? Azért léteznek különböző routolási algoritmusok, azért fecsegnek ezek a routerek olyan sokat egymás között, hogy lehetőleg minden csomag optimális útvonalon közlekedjen. Hogyan jövök én, a feladó host, ahhoz, hogy előírjam az útvonalat?
Emlékezzünk vissza: mire is használjuk általában az IP Options blokkokat? Tesztelésre. Itt is erről van szó. Ha olyan a hálózatunk, hogy van benne egy alacsonyabb költségű útvonal, meg a biztonság kedvéért egy magasabb költségű alternatív útvonal, akkor hogyan tudjuk letesztelni, hogy a második útvonal éppen működik-e? Hiszen minden csomag az első útvonalon megy. Működő hálózatot meg megszaggatni a teszt kedvéért... nem elegáns.
Ilyenkor jönnek be a képbe a SOURCE ROUTING opciók.



4.10. ÁBRA STRICT SOURCE ROUTING

OPTION CODE: STRICT esetben 137, LOOSE esetben 131. (Kibontás: *4.1. táblázat*)

OPTION LENGTH ÉS NEXT SLOT POINTER: Ugyanaz a matek, mint az előző IP opciónál.

FIRST IP ADDRESS, SECOND IP ADDRESS, OTHER IP ADRESSES: Ide kéretik felsorolni az összes érintendő IP címet. Mennyi is lehet belőlük? Az IP OPTIONS max 40 bájt, ebből megettünk hármat, marad 37, az IP cím 4 bájtos, azaz osztunk néggyel - marad kilenc. Ennél hosszabb útvonalat, ha belegebedünk sem tudunk összeállítani.

Szaladjunk át gyorsan a folyamaton. A router/destination host megkapja a csomagot. Ha a NEXT SLOT POINTER értéke nagyobb, mint az OPTION LENGTH értéke, akkor megdöbbenve veszi tudomásul, hogy a csomag neki szólt. Amennyiben nem, akkor a következők történnek:

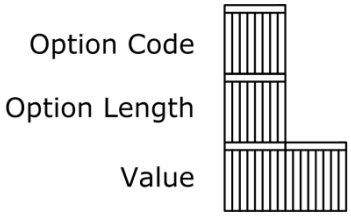
- Megnöveli néggyel a NEXT SLOT POINTER mező értékét.
- A router belenyúl az IP datagram hátizsákjába, előveszi a következő IP címet.
- Ezzel az IP címmel felülírja az IP csomag fejlécében szereplő DESTINATION IP ADDRESS mező értékét.
- Belerakja a saját címét az IP csomag hátizsákjába, méghozzá úgy, hogy az előző hop-nál elhasznált IP címet írja felül.
- Kirúgja a csomagot.

Egy utolsó gondolat: a SOURCE ROUTING nem mindenhol engedélyezett. Az interneten például tipikusan nem.

4.1.1.2.4 IP ROUTER ALERT

Azaz a vészcsengő. Ez az, amikor az óvodás nézők felkiabálják Vitéz Lászlónak, hogy 'Vigyázz, jön a Csokoládépofa!'.

Ugyanez informatikusra fordítva azt jelenti, hogy ez az IP opció figyelmezteti az aktuális hostot, hogy vigyázat, ezzel a csomaggal valamit csinálnia kell a továbbpasszoláson kívül. Tipikusan ilyenek a később részletezendő IGMP csomagok.



4.11. ÁBRA IP ROUTER ALERT

Látható, a csomag szerkezete nincs igazán túlbonyolítva.

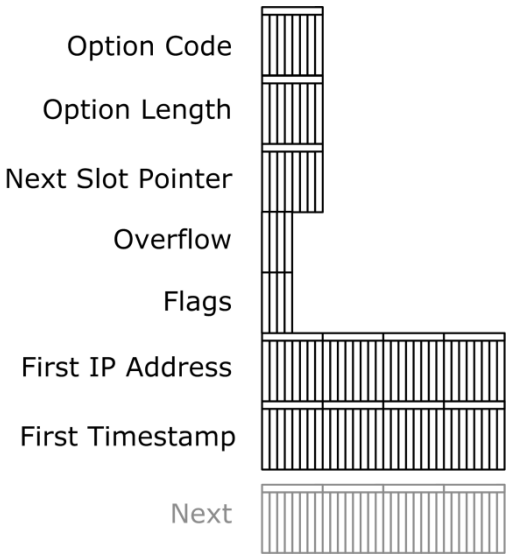
OPTION CODE: Értéke 148.

OPTION LENGTH: Értéke fix 4-es.

VALUE: Jelen esetben az értéke 0.

4.1.1.2.5 INTERNET TIMESTAMP

Némileg hasonlít a RECORD ROUTE opcióhoz - csak itt nem az útvonalat, hanem a csomag megérkezési időpontjait rögzítik a routerek, illetve a célállomás.
Pusztán érdekességként jegyzem meg, hogy az időt az előző éjfél óta eltelt milliszekundumokban méri.



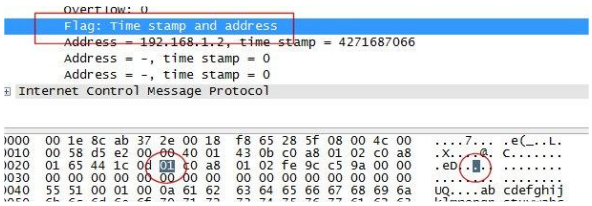
4.12. ÁBRA INTERNET TIMESTAMP

OPTION CODE: 68. Ugye, már nem kell részleteznem?

OPTION LENGTH, NEXT SLOT POINTER: Ugyanazok, mint eddig.

OVERFLOW: Láthattuk, az IP OPTIONS blokk mérete véges. Mi van, ha kilencnél több hoston megy át a csomag? Akkor itt, az OVERFLOW mezőben láthatjuk, hány host maradt ki. (4 bit, tehát max 15.)

FLAGS: Itt tudjuk megadni, hogy tulajdonképpen mit is rögzítsünk: csak az időbélyeget(0), vagy írjuk mellé a host IP címét is(1)? Értelemszerűen az utóbbi esetben feleannyi hop adatait tudjuk rögzíteni, azaz hamarabb lép pályára az OVERFLOW mező is.

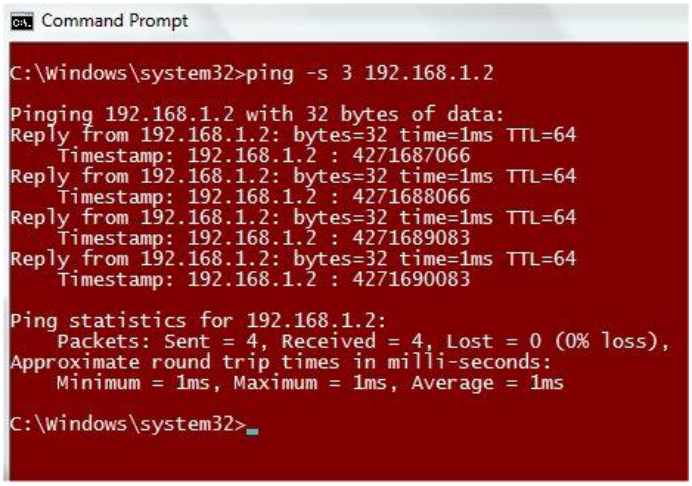


4.13. ÁBRA FLAG=1

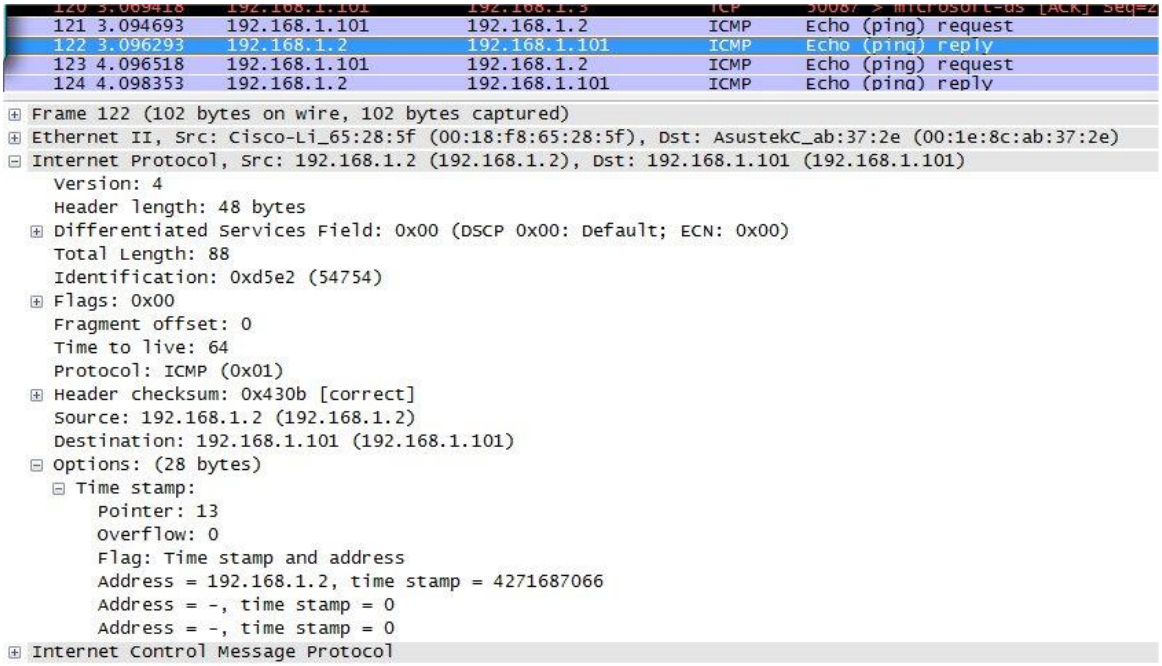
Van még egy meglehetősen különleges értéke a FLAG mezőnek, ez a 3-as. Ekkor megint hátizsákozunk: belerakunk előre IP címeket a batyuba, az időbélyeg pedig csak akkor rögződik, amikor elérjük az előírt IP címmel rendelkező hostot.

IP ADDRESS / TIMESTAMP: A rögzített értékek.

És akkor a demó.



4.14. ÁBRA INTERNET TIMESTAMP MEGADÁSA A PING PARANCSON BELÜL



4.15. ÁBRA INTERNET TIMESTAMP LELEPLEZVE

A fenti ábrából látható, hogy csak a szomszédos hostot pingettem meg. (Egy IP cím és egy időbélyeg bejegyzés van mindösszesen a batyuban.) Én a szívem szerint pingettem volna távolabbi hostokat is, de már a következő routeren sem volt engedélyezve ez az IP opció... az internetről már nem is beszélve.

Matekozzunk egy kicsit. Számoljuk ki például az időbélyegből, mennyire volt kaporszakállú öreg este, amikor ez a capture készült?
Mindkét ábra azt mondja, hogy az időbélyeg: 4271687066 ms, azaz osztva 1000 és osztva 3600, az annyi, mint az előző éjfél után 1186,6 óra.
Ejnye, de megváltozott mostanság az időszámítás.

Valami nem stimmel.

Tovább olvasva az RFC 791-et, azt mondja, hogy amennyiben nem áll a router rendelkezésére UTC formájú idő, vagy az nem millisecundum pontosságú, akkor gyakorlatilag bármi is lehet az időbélyeg. Ezt onnan tudjuk, hogy ilyenkor a legmagasabb helyiértékű bit magasra lett állítva. Hogyan is néz ki a számunk binárisban?
Így: 11111110100111001100010110011010.
Bizony, a bal szélső érték magas... azaz innentől nem köthető össze direktben a szám a valós időértékkel.
De legalábbis nem publikus a képlet.

4.1.2 ÉLET A HATÁRON: ROUTE, NAT, PROXY

Sokáig - kimondva-kimondatlanul - olyan esetekről beszéltünk, amikor a feladó és a címzett egy alhálózaton vannak. Csakhogy az előző fejezetben már kényelmetlenül sokszor hangzott el az a szó, hogy 'router'. Feltételezem, sok emberben horgadt fel a kérdés, hogy mennyiben is változnak meg a csomagok akkor, ha határőrök is vegzálják a küldeményeket?

Akkor nézzük, mi is történik pontosan a határon. Az első, és messze a legkorrektebb megközelítés az, hogy attól függ[35].

Ugyanis egy csomagot lehet routolni meg natolni.

Mi a különbség?

Messziről fogunk közelíteni.

Mindenki tudja, mi az az IP cím?

[35] Minden kérdésre válaszold nyugodtan azt, hogy attól függ. Egyrészt így a kérdező lesz rákényszerítve arra, hogy több információt adjon, másrészt időt nyersz a jó válasz megfogalmazására is. Csak az önmaguktól elvakult emberek, illetve a reménytelenül naívak akarnak mindig egyből válaszolni.

4.1.2.1 Az IP cím

Oké, elismerem, egy hálózatokról és a TCP/IP-ről szóló könyv százvalahanyadik oldalán elég hülyén hangzik ez a kérdés. De úgy tapasztaltam, néha nem árt az evidenciákat is elmagyarázni, hogy szép kerek legyen az anyag.

Tehát az IP cím az egy négybájtos azonosító, mely önmagában nem mond semmit. Minden IP címhez tartozik egy másik négybájtos szám, az ún. alhálózati maszk (subnet mask). Ez definiálja, hogy magából az IP címből hány bit azonosítja a hálózatot és hány bit a hálózaton belül a kérdéses hostot.
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 host 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 host 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 host 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 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 host lehet. (Igazából 254, mert a két szélső cím fenn van tartva a hálózat beazonosítására, illetve a broadcast címre.)

Vizsgáljunk meg egy cifrább esetet.
Hogyan szedjük ketté a következő IP címet: 10.48.247.6/20?
Binárisan:

- IP cím : 00001010 00110000 1111**0111 00000110**
- Alhálózati maszk : 11111111 11111111 1111**0000 00000000**

Innen már csak vissza kell konvertálgatni:

- Hálózat azonosító : 10.48.240
- Host azonosító : 7.6
- A hostok maximális száma pedig: 2^{12}-2, azaz 4094.

Csavarjunk még egyet a példán. Fogja-e egymást közvetlenül látni a következő két host: 10.48.247.6/20 és 10.48.247.5/22?
Miután mindenki megtette a tétjeit, vágjunk bele. Mikor látja egymást két host? (Router nincs.) Ha azonos alhálózaton vannak.
Matek.
Az első cím felbontását fentebb láthattuk. Jöhet a második cím.

- IP cím : 00001010 00110000 11110**111 00000101**
- Alhálózati maszk : 11111111 11111111 11111**100 00000000**
- Hálózat azonosító : 10.48.244
- Host azonosító : 3.5
- A hostok maximális száma : 2^{10}-2, azaz 1022.

Megegyezik a két hálózat azonosítója? Nem. Tehát a két host közvetlenül nem látja egymást - dacára annak, hogy az IP címük szomszédos. Csak éppen a maszk különbözik.

Még egy feladat, ha már így belejöttünk. Fogja-e látni egymást - router nélkül - az alábbi két host?

4.4. TÁBLÁZAT

Név	IP cím	Alhálózati maszk
host1	192.168.154.63	255.255.255.192
host2	192.168.154.64	255.255.255.192

host1:

- IP cím : 11000000 10101000 10011010 00**111111**
- Alhálózati maszk : 11111111 11111111 11111111 11**000000**
- Hálózat azonosító : 192.168.154.0
- Host azonosító : 63
- Hostok maximális száma : 2^6-2, azaz 62.

host2:

- IP cím : 11000000 10101000 10011010 01000000
- Alhálózati maszk : 11111111 11111111 11111111 11000000
- Hálózat azonosító : 192.168.154.64
- Host azonosító : 0
- Hostok maximális száma : 2^6-2, azaz 62.

Nos, látjuk? Megint nem egyeznek meg a hálózati azonosítók. Megint nem fogják látni egymást. Pedig itt ránézésre minden stimmelt, azonosak voltak az alhálózati maszkok, szomszédosak voltak az IP címek... aztán mégsem.[36]
Maradjunk annyiban, hogy gyakorlott szem kell az IP címekkel való kavaráshoz. És ha csak egy picit is bizonytalan vagy, akkor mindenféle szégyenkezés nélkül elő kell kapni a calc.exe programot és kipötyögni a választ.

Vissza a fősodorhoz. Nos, a világ nem volt mindig ilyen rugalmas, mint ahogy én eddig kavartam. Nagyon sokáig ilyen bontás egész egyszerűen nem volt engedélyezve. Az IP címek osztályokba voltak sorolva, mégpedig az alhálózati maszkok szigorú behatárolásával. Valahogy így.

4.5. TÁBLÁZAT

Osztály	Bal oldali bitek	Hálózatokat leíró bitek száma	Alsó határ	Felső határ	Hálózatok száma	Hostokat leíró bitek száma	Hostok száma
A	0	8	1.0.0.1	126.255.255.254	127	24	16777214
B	10	16	128.1.0.1	191 .255.255.254	16384	16	65534
C	110	24	192.0.0.1	223.255.255.254	2097152	8	254
D (multicast)	1110		224.0.0.0	239.255.255.254			
E (foglalt)	1111		240.0.0.0	254.255.255.254			

Ezt valamikor úgy hívták, hogy Classful IP címkiosztás. Szerencsére elröpült felette az idő vasfoga. Ez ugyanis borzasztóan pazarló volt - az IP címek pedig kezdtek rohamosan elfogyni.
A címfogyásra volt az egyik megoldás a CIDR[37], azaz az alhálózati maszkok sokkal rugalmasabb tologatása. (Ezt mutatta be a második/harmadik matekozós példa.)
A másik megoldás pedig a NAT[38].

Az a NAT, ami máshogy működik, mint a routolás.

[36] Tessék észrevenni az ordas csalást. Host1 igazából egy alhálózat legfelső címe, azaz a broadcast cím, Host2 pedig a vele fentről érintkező alhálózat legalsó címe, azaz a hálózat azonosítója. Ergo ilyen IP című hostok nem is létezhetnek. Ettől persze a példa még jó, csak éppen a valóságban host1 utolsó oktettje 62, illetve host2 utolsó oktettje 65 lett volna.
[37] Classless Inter Domain Routing
[38] Network Address Translation

4.1.2.2 ROUTE

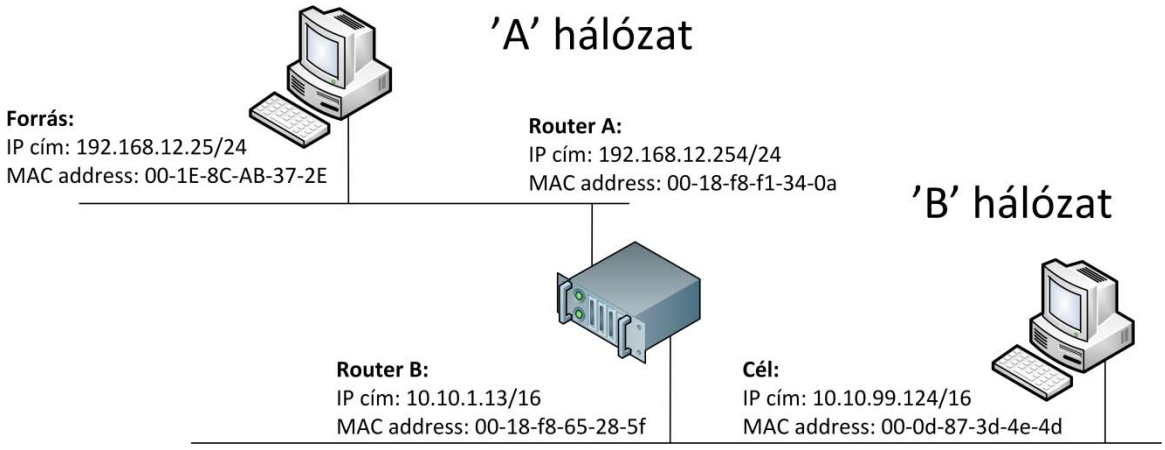
Ehhez persze először el kell mélyednünk a routolás lelkivilágába - és majd csak utána térhetünk vissza a NAT-hoz.

Már az sem egyszerű, mit is nevezünk routolásnak? Routolás az, ha betárcsázunk/bevépéenezünk egy hálózatba, ahol egy IP poolból kapunk címet? Nem, ha az IP pool a belső hálózatból van kihasítva, azaz ugyanaz a hálózati azonosítónk, mint a többi gépé. Ekkor csak bridzselés[39] történik.
Routol például egy switch? Nem. A switchek általánosságban az ISO/OSI L2 rétegében dolgoznak (ebben a könyvben ez a NIL rétegnek felel meg), míg a routerek az L3-ban[40]. (Amely itt az Internet réteg.)

Gondolom, már körvonalazódik: routolás akkor történik, ha egy küldemény az egyik alhálózatból egy másikba szeretne átmenni. (A protokoll természetesen azonos.) A router pedig az az elem, amelyiknek egyik lába az egyik hálózatba, a másik lába pedig a másikba lóg bele[41] - ő pedig képes a kettő közötti forgalom lebonyolítására.

Alakulunk.

Ássunk bele egy kitalált folyamatba.



4.16. ÁBRA ROUTOLÁS HÁLÓZATBAN

1. A **Forrás** névvel jelölt számítógépnek halaszthatatlan közlendője támad, melyet a **Cél** nevű számítógéppel szeretne megosztani. A hálózat egyszerű Ethernet.

[39] 1 szan - 2 treff.
[40] Tudom, hogy van L3 switch is, de ezekkel most nem akarom bonyolítani a könyvet.
[41] Megjegyzem, láttam már százlábú routert is.

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 hostnak 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 address-ét, 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 (*2.1. ábra Rétegek és csomagok (forrás: Wikipedia*), így a Target IP cím hiába a Cél számítógépé, de a Target MAC address miatt a **Router A** magáénak fogja érezni a csomagot. Felszipkázza. A Target IP alapján értelmezi a feladatot, megkeresi, melyik lábán kell továbbítania a csomagot (ne feledjük, léteznek százlábú routerek is), az ARP segítségével begyűjti a **Cél** számítógép MAC address 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 address 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?

Először ismerkedjünk meg a Default Gateway, illetve a route tábla fogalmakkal.
A Default Gateway, mint a neve is mutatja, maga a default, azaz az alapértelmezett. Ez elméletileg azt jelenti, hogy minden esetben, amikor a host a saját alhálózatáról idegen alhálózatra szeretne csomagot küldeni, akkor erre a címre továbbítja azt.

A 4-es pontban innét, ebből a beállításból tudta a **Forrás**, hogy ki lesz a következő hop az útvonalon. Elméletben.

A gyakorlatban viszont röhögve megjelenik a színen a route tábla.

A route tábla ugyanis egy nagyon pontosan, finoman szabályozható útválasztó táblázat, melynek része a Default Gateway koncepció is, de ennél jóval finomabban is lehet szabályozni az egyes megcélzott útvonalakat. Gondoljunk bele például egy olyan esetbe, ahol egy alhálózatból nem csak egy router nyit kijáratot.

Régen matekoztunk már.

```
===========================================================================
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
===========================================================================
```

Így néz ki a számítógépem útválasztási táblázata. Ránézésre vannak sorok, melyek egészen jól értelmezhetőek... de az egész... egy kicsit zűrös.

De csak addig, amíg el nem kezdünk számolgatni.

Vegyünk 3 megcélzott IP címet:

1. 192.168.1.103, binárisan 11000000 10101000 00000001 01100111
2. 192.168.99.17, binárisan 11000000 10101000 01100011 00010001
3. 195.247.14.56, binárisan 11000011 11110111 00001110 00111000

Most jön a neheze. Szorozzuk össze ezeket egyenként binárisan (AND) az útválasztási táblázatban szereplő Netmask értékekkel, melyeket a lenti táblázat 2. és 3. oszlopa is tartalmaz..

1. (192.168.1.103):
11000000 10101000 00000001 01100111 **AND...**

4.6. TÁBLÁZAT

Ssz	NetMask	NetMask binárisan	Eredmény
1	0.0.0.0	00000000.00000000.00000000.00000000	00000000.00000000.00000000.00000000
2	255.0.0.0	11111111. 00000000.00000000.00000000	11000000 00000000 00000000 00000000
3	255.255.255.255	11111111.11111111.11111111.11111111	11000000 10101000 00000001 01100111
4	255.255.255.255	11111111.11111111.11111111.11111111	11000000 10101000 00000001 01100111
5	255.255.255.0	11111111.11111111.11111111.00000000	11000000 10101000 00000001 00000000
6	255.255.255.255	11111111.11111111.11111111.11111111	11000000 10101000 00000001 01100111
7	255.255.255.255	11111111.11111111.11111111.11111111	11000000 10101000 00000001 01100111
8	255.255.255.0	11111111.11111111.11111111.00000000	11000000 10101000 00000001 00000000
9	240.0.0.0	11110000.00000000.00000000.00000000	11000000.00000000.00000000.00000000
10	240.0.0.0.	11110000.00000000.00000000.00000000	11000000.00000000.00000000.00000000
11	255.255.255.255	11111111.11111111.11111111.11111111	11000000 10101000 00000001 01100111
12	255.255.255.255	11111111.11111111.11111111.11111111	11000000 10101000 00000001 01100111

Írjuk fel a Network destination értékeket is binárisan, majd tegyük mellé az előző táblázat Eredmény oszlopát.

4.7. TÁBLÁZAT

Ssz	Network Destination	Network Destination binárisan	Előző eredmény	CT
1	0.0.0.0	00000000.00000000.00000000.00000000	00000000.00000000.00000000.00000000	32
2	127.0.0.0	01111111. 00000000.00000000.00000000	11000000 00000000 00000000 00000000	0
3	127.0.0.1	01111111. 00000000.00000000.00000001	11000000 10101000 00000001 01100111	0
4	127.255.255.255	01111111.11111111.11111111.11111111	11000000 10101000 00000001 01100111	0
5	192.168.1.0	11000000.10101000.00000001.00000000	11000000 10101000 00000001 00000000	32
6	192.168.1.101	11000000.10101000.00000001.01100101	11000000 10101000 00000001 01100111	30
7	192.168.1.255	11000000.10101000.00000001.11111111	11000000 10101000 00000001 01100111	24
8	192.168.99.0	11000000.10101000.01100011.00000000	11000000 10101000 00000001 00000000	17
9	224.0.0.0	11110000.00000000.00000000.00000000	11000000.00000000.00000000.00000000	2
10	224.0.0.0.	11110000.00000000.00000000.00000000	11000000.00000000.00000000.00000000	2
11	255.255.255.255	11111111.11111111.11111111.11111111	11000000 10101000 00000001 01100111	2
12	255.255.255.255	11111111.11111111.11111111.11111111	11000000 10101000 00000001 01100111	2

A táblázathoz hozzáfűztem egy CT, azaz counter oszlopot. Ez az érték mutatja azt, hogy ha a harmadik és a negyedik oszlopot balról jobbra haladva binárisan összehasonlítom, akkor meddig, azaz hanyadik karakterig tart az egyezés.
Az a sor nyer, ahol a legnagyobb ez az érték. Holtverseny esetén a NetMask egyeseinek száma dönt.

Jelen táblázatban mind az első, mind az ötödik sorban a CT értéke 32, de az első sorhoz tartozó Netmask-ban az egyesek száma határozott nulla, míg az ötödik sorban 24. Azaz a 192.168.1.103 cím megtámadása esetén az 5. sor nyert, ezt visszakeresve a route táblából, láthatjuk, hogy a 192.168.1.101 hálózati kártyán kell kimennünk, és onlink-en vagyunk, tehát nincs szükségünk routerre.

2. (192.168.99.17):
11000000 10101000 01100011 00010001 **AND…**

Az előző példában becsületesen végigszámoltunk minden sort. A továbbiakban annyiból egyszerűsítenék, hogy a localhostra (127.0.0.1), a multicast-ra (240.0.0.0) és a broadcast-ra (255.255.255.255) vonatkozó sorokat kivenném. A példáinkban ezek sohasem fognak nyerni.

4.8. TÁBLÁZAT

Ssz	NetMask	NetMask binárisan	Eredmény
1	0.0.0.0	00000000.00000000.00000000.00000000	00000000.00000000.00000000.00000000
2			
3			
4			
5	255.255.255.0	11111111.11111111.11111111.00000000	11000000 10101000 01100011 00000000
6	255.255.255.255	11111111.11111111.11111111.11111111	11000000 10101000 01100011 00010001
7	255.255.255.255	11111111.11111111.11111111.11111111	11000000 10101000 01100011 00010001
8	255.255.255.0	11111111.11111111.11111111.00000000	11000000 10101000 01100011 00000000
9			
10			
11			
12			

Írjuk fel a Network destination értékeket is binárisan, majd tegyük mellé az előző táblázat Eredmény oszlopát.

4.9. TÁBLÁZAT

Ssz	Network Destination	Network Destination binárisan	Előző eredmény	CT
1	0.0.0.0	00000000.00000000.00000000.00000000	00000000.00000000.00000000.00000000	32
2				
3				
4				
5	192.168.1.0	11000000.10101000.00000001.00000000	11000000 10101000 01100011 00000000	17
6	192.168.1.101	11000000.10101000.00000001.01100101	11000000 10101000 01100011 00010001	17
7	192.168.1.255	11000000.10101000.00000001.11111111	11000000 10101000 01100011 00010001	17
8	192.168.99.0	11000000.10101000.01100011.00000000	11000000 10101000 01100011 00000000	32
9				
10				
11				
12				

Az előző algoritmus alapján most a 8-as sor nyert, azaz a 192.168.99.17 cím esetében a 192.168.1.101 hálózati kártyánkon kell kimennünk, és a 192.168.1.2 címen lesz az a gateway, amelyik kienged minket a megfelelő irányba.

3. (195.247.14.56):
11000011 11110111 00001110 00111000 **AND…**

Most már nem kell olyan sok duma.

4.10. TÁBLÁZAT

Ssz	NetMask	NetMask binárisan	Eredmény
1	0.0.0.0	00000000.00000000.00000000.00000000	00000000.00000000.00000000.00000000
2			
3			
4			
5	255.255.255.0	11111111.11111111.11111111.00000000	11000011 11110111 00001110 00000000
6	255.255.255.255	11111111.11111111.11111111.11111111	11000011 11110111 00001110 00111000
7	255.255.255.255	11111111.11111111.11111111.11111111	11000011 11110111 00001110 00111000
8	255.255.255.0	11111111.11111111.11111111.00000000	11000011 11110111 00001110 00000000
9			
10			
11			
12			

Írjuk fel a Network destination értékeket is binárisan, majd tegyük mellé az előző táblázat Eredmény oszlopát.

4.11. TÁBLÁZAT

Ssz	Network Destination	Network Destination binárisan	Előző eredmény	CT
1	0.0.0.0	00000000.00000000.00000000.00000000	00000000.00000000.00000000.00000000	32
2				
3				
4				
5	192.168.1.0	11000000.10101000.00000001.00000000	11000011 11110111 00001110 00000000	6
6	192.168.1.101	11000000.10101000.00000001.01100101	11000011 11110111 00001110 00111000	6
7	192.168.1.255	11000000.10101000.00000001.11111111	11000011 11110111 00001110 00111000	6
8	192.168.99.0	11000000.10101000.01100011.00000000	11000011 11110111 00001110 00000000	6
9				
10				
11				
12				

Az előző algoritmus alapján most az 1-es sor nyert, azaz a 195.247.14.56 cím esetében a 192.168.1.101 hálózati kártyánkon kell kimennünk, és a 192.168.1.1 címen lesz az a gateway, amelyik kienged minket a megfelelő irányba.

```
Windows IP Configuration


Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::21e:8cff:feab:372e%8
   IPv4 Address. . . . . . . . . . . : 192.168.1.101
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1
```

Ez viszont, mint fent is láthatjuk, pont a Default Gateway. Mely minden eddigi példában ott volt holtversenyben az első helyen, csak a NetMask-jában lévő kevés egyes miatt veszített. Itt viszont nem volt holtverseny.

Azaz összességében elmondhatjuk, hogy amikor el kell döntenünk, hogy egy csomagot melyik hálózati kártyánkon és melyik router felé küldjük ki, akkor amennyiben van olyan sor az útválasztó táblában, mely illeszkedik a csomagban lévő target IP address-re (32-es counter), akkor az a sor nyer. Holtverseny esetén a szorosabb illeszkedés a győztes.
(Egy 192.168.105.23 címre való küldésnél illeszkedni fog a 192.168.105.0/24, illetve a 192.168.0.0/16 sor is az útválasztási táblában - de érezzük, hogy az első a szorosabb. A legszorosabb pedig a 192.168.105.23/32 lenne.)
Ha egyik sor sem illeszkedik, akkor a Default gateway lesz a nyerő: hiszen a 0.0.0.0/0 mindenre passzol.

Illetve van még egy tényező, amelyről eddig nem beszéltünk: ez a Metric paraméter. Ez egy költség érték - és a sorokhoz rendeljük. Azonos illeszkedés esetén az alacsonyabb költségű sor nyer.

Térjünk vissza még egy gondolat erejéig a fenti példákhoz. Láthattad, ez nem egy átlagos számítógép route táblája volt - hiszen akkor lenne egy default gateway és kész. Itt viszont közöltük a számítógéppel, hogy a 192.168.99.0/24 hálózat nem a Default gateway mögött van, hanem arra egy másik kijárat - a 192.168.1.2 - vezet. Ezt a következő paranccsal tudtuk elérni:

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

És mivel használtam a *-p* kapcsolót, így ez egy persistent, azaz állandó bejegyzés lett a táblában. Újraindítás során sem veszik el. Nem mellékesen, a route print parancs esetén a persistent route bejegyzések külön sorban is látszanak.

Oké. Megismertük a route táblát. Tudjuk, mi alapján dönt egy host. De hogyan dönti el egy router, hogy melyik hálózati adapterén küldjön ki egy csomagot, ha a célállomás neadjisten sok routerrel arrébb található?

Hát, például megszerkeszthetünk egy univerzális, bazi nagy route táblát, melybe belevesszük az összes alhálózatunkat, majd ezt - megfelelően az egyes routerekre szabva - feltöltögetnénk az útválasztóinkra. Ténylegesen is létezik ilyen, úgy hívják, hogy statikus útválasztás.
De nem ez a hosszú élet titka.

Manapság már sokkal elterjedtebb a dinamikus útválasztás. Ekkor a routerek, különböző algoritmusokat és protokollokat használva lefocizzák egymás között azt, hogy a route tábláik mindig naprakészek legyenek.

Két ilyen útválasztási algoritmust említenék meg:

- RIP
- OSPF

Routing:
http://en.wikipedia.org/wiki/Routing

Implementing IP routing:
http://technet.microsoft.com/en-us/library/cc750576.aspx

IP Routing:
http://technet.microsoft.com/en-us/library/bb727001.aspx

4.1.2.3 RIP

Nyilván amikor elnevezték, jót vigyorogtak a Rest In Peace (Nyugodjék békében) áthalláson - de valójában a betűszó azt jelenti, hogy Routing Information Protocol.

Maga a protokoll az ún. *distance-vector routing* protokollok családjába tartozik. Ez nagyjából azt takarja, hogy az ugrásszámokat (hop count) használja routolási költségként. Ezeket az értékeket tárolja egy routolási táblában, mely táblákat bizonyos időszakonként mindegyik router szétkürtöli a nagyvilágba. A többi router veszi az adást - na meg persze adja is a sajátját - így érve el egyfajta konvergenciát.
Nagy hátránya az algoritmusnak az, hogy nagyon hamar eléri a végtelent: 15 hop felett már kezelhetetlen forgalmat produkálna a hálózaton - így hivatalosan is ez a felső korlátja. Emiatt nagy hálózatokban nem használható, mivel a 16 hop-ra lévő routerek már nem látnák egymást.

A RIPv1 még csak a classful világban tudott dolgozni (nem használt alhálózati maszkot, hiszen ekkor az IP cím egyben meg is határozta az IP osztályt), a RIPv2 már ismeri a CIDR-et. A RIPng (new generation) pedig már támogatja az IPv6-ot.

Adattovábbításra UDP protokollt használ - azaz egy tipikus alkalmazás szintű protokollról van szó.

Routing Information Protocol:
http://en.wikipedia.org/wiki/Routing_Information_Protocol

Distance Vector Routing Protocol:
http://en.wikipedia.org/wiki/Distance-vector_routing_protocols

4.1.2.4 OSPF

Az akronim az Open Shortest Path First kifejezést takarja. A protokoll a *link-state routing* protokollok népes családjába tartozik. (Itt lakik még az IS-IS és a BGP is.)

Akik otthonosan mozognak az Exchange 2000/2003 routolásában, most felsóhajthatnak: ismerős dolgok jönnek.
A rendszerben a routerek között kapcsolatok - linkek - vannak. Mindegyik link egy-egy súlyozott vektor. Ebben a topológiában az OSPF nagyon hamar képes megtalálni az optimális útvonalat. A linkek egy adatbázisban - Link-State Database, LSDB - laknak, ez az adatbázis terjed a routerek között, mint a nátha.

Logikusan következnek az előnyei:

- Csak ha változás történik egy linkben, akkor kezdenek el kommunikálni egymással a routerek. (Eltekintve a belső fecsegésüktől.)
- Elég ha a routerek a szomszédaikkal cserélik ki az LSDB változásokat - előbb-utóbb a változások mindenhová eljutnak.

A fentiekből jön, hogy jóval nagyobb hálózatok is kezelhetők vele, mint a RIP protokollal, és nem mellékes az sem, hogy bármelyik konnektor kiesése viszonylag hamar detektálható.

Az OSPF a hálózati eszköz rétegben dolgozik. Nyilván ennek is van v2, v3 verziója, sőt, az MOSPF a multicastot is támogatja.

OSPF:
http://en.wikipedia.org/wiki/Open_Shortest_Path_First

Rendben.

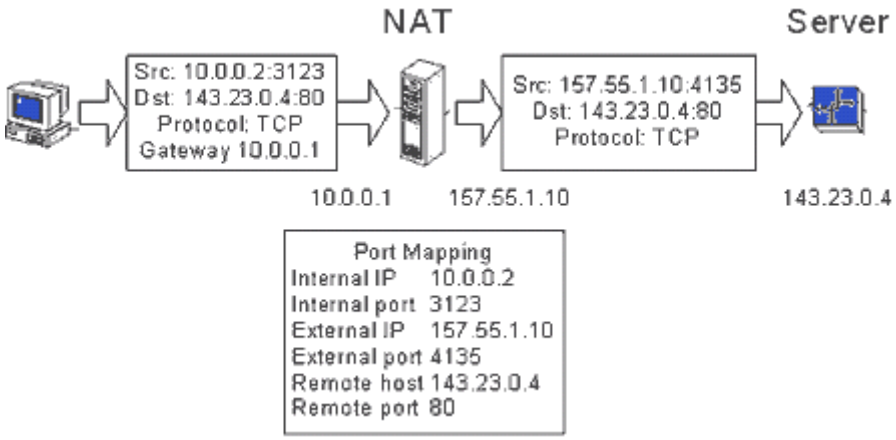
Válaszoltunk mindkét kérdésre. Tudjuk, mi alapján dönt egy router, és tudjuk azt is, hogyan értesülnek az alhálózatok topológiájáról a routerek. Ráadásul tudjuk azt is, mi minden változik egy csomagban, ha az átmegy egy routeren. (Ugye emlékszünk: a MAC address.)

Feltéve, hogy a router routol és nem natol.

De miért is csinálna ilyet?
Hát, például azért, mert rohamosan fogynak a kiadható IP címek az IPv4 világában.

4.1.2.5 NAT

Lassan tényleg nem úszom meg, hogy ne áruljam el, mi is a NAT. Kikódolva annyit tesz, hogy Network Address Translation, magyarul talán hálózati címek fordításának lehetne nevezni.
Az a trükkje, hogy a két hálózat közös határán lebzselő router meg tudja személyesíteni az 'A' hálózatban található bármelyik hostot a 'B' hálózatban, úgy, hogy a saját 'B' hálózatbéli IP címéről nyomul.
Szerintem ábrán sokkal érthetőbb lesz.



4.17. ÁBRA NAT MŰKÖDÉS KÖZBEN

Ja, eddig nem beszéltünk a 'port' fogalmáról. Ez magyarul 'kapu'-t jelent... de még a legelvakultabb nyelvvédő informatikusok sem szokták lefordítani.
Gondoljuk csak el: egy host működés közben rengeteget kommunikál a hálózaton. Nem ritka, hogy éppen jön le a böngészőbe egy oldal, a levelezőkliens pont ránéz a szerverre, hogy van-e új levél, a torrent kliens pedig a Szerzői Jogi törvényeket tölti le ugyanekkor. Ha mindez egy időben megy, akkor nagy gondban lennénk,

amennyiben mi lennénk a hálózati kártya: vajon melyik csomag melyik alkalmazáshoz tartozik?
Erre találták ki a port számot.
Azaz igazából amikor egy alkalmazás (böngésző) kommunikálni akar egy másikkal (weboldal), akkor mind a feladó, mind a címzett *teljes* címe így néz ki: *ip_cím:port*

Példa[42]:

- Feladó : 192.168.11.124 :2874
- Címzett : 217.20.130.97 :80

Ez egész konkrétan azt jelenti, hogy a 192.168.11.124 IP című gép - valószínűleg egy böngészőből - az index.hu nyitólapját kérte le.

Eredetileg szabad volt a vásár portok terén, aztán kialakultak rögzített - de csak ajánlott! - port számok. A 80-as például tipikusan a http szolgáltatásé - de nem kötelezően. A 25-ös az smtp szolgáltatásé... és így tovább. Aztán vannak zónák is, az 1024-65535 tipikusan a kliens portoké, azaz a kliens programok feladó címeié. Egyben a 65535 a legmagasabb szóbajöhető érték is. Fejszámolók ebből kapásból ki is számolhatták, hogy a port szám részére 2 bájt áll rendelkezésre.

Most, hogy már tudjuk, mi az a port, nézzük át alaposan az ábrát: *4.17. ábra NAT működés közben.*

A 10.0.0.2 forrás IP címről (port: 3123, tipikus kliens port) egy webszerverhez szeretnénk kapcsolódni: 143.23.0.4:80. 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 felé a csomagot.

Kicsit pontosítsunk.
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.

[42] Ezt így egyébként hívják socket-nek is.

> 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.

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

Szép, szép. De nyilván az akció számolásigényes művelet. Miért van nekünk erre szükségünk? Miért nem jó a jóval egyszerűbb route művelet?

Mint írtam korábban, durván kezdtek elfogyni a kiadható IPv4 címek. A CIDR sokat segített ugyan, de nem eleget.

Erre találták ki a privát IP tartományokat.

4.12. TÁBLÁZAT

Címtartomány	Alsó határ	Felső határ
A	10.0.0.0	10.255.255.255
B	172.16.0.0	172.31.255.255
C	192.168.0.0	192.168.255.255

Illetve később bővült a klub.

> RFC 3927

4.13. TÁBLÁZAT

Kezdő cím	Végső cím	Címek száma
169.254.0.0	169.254.255.255	65536

Ez utóbbi az IPv4 link-local tartomány. (Leánykori neve APIPA, Automatic Private Internet Protocol Addressing.)

A fenti IP tartományokat csak belső hálózatokon illik használni. Tisztességes edge routerek - azaz az internet határait őrző routerek az ISP-knél - ezeket a tartományokat nem routolják ki. Sőt, a tisztességes céges edge routerek már el sem engedik az internetes edge routerekig. Semmi keresnivalójuk a publikus neten.

Mi következik ebből? Például az, hogy én, mint Nagy Cég, vehetek a szolgáltatómtól 1, azaz egy IP címnyi hozzáférést a nethez. Ide leteszek egy böszme erős routert, mögé pedig simán betehetek több millió gépet, hiszen natoló routerekből akár végtelen méretű hálózatot is ki tudok építeni. A belső IP címe egyiknek sem fog megjelenni a neten.

Aztán lehet mindegyikről tolni lógó nyelvvel az iwiwet.

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.

4.1.2.6 Proxy

Habár ez már nagyon alkalmazás rétegbeli technika, a tisztánlátás kedvéért ejtenék róla pár szót itt is.

A proxy azt jelenti, hogy helyettesítő. Informatikában azt értjük alatta, hogy jön 'A' gép, szeretne elérni valamit 'B' gépen - de ezt közvetlenül nem tudja megtenni. Meg kell rá kérnie 'C' gépet. Ő a proxy.
Mint amikor elmegyek egy újgazdaghoz, az inassal beküldöm a névjegyemet, aki cserébe kihoz nekem egymillió eurót. Itt az inas volt a proxy, én nem is találkoztam a gazdag emberrel. Ez jó lehet neki, mert mi van, ha nálam revolver van és több pénzt akarok - illetve jó lehet nekem, mert lehet, hogy az illető ebolás és már a pillantása is fertőz. Az inas pedig biztosan fegyvertelen és oltva is van.

Nem ismerős? Nem ugyanezt csinálta a NAT?

Nos, nem. Nem véletlenül tettem ide ezt a fejezetet.

Kezdjük azzal, hogy a két technológia más szinten működik. A proxy az alkalmazás rétegben, a NAT az internet rétegben. De óriási különbség van a mögérakott intelligenciában is. A proxy konkrét alkalmazás-rétegbeli protokollra van beállítva. (Külön proxy-t kell írni a http-re, külön proxy-t az smtp-re, és így tovább.) A proxy pontosan ismeri a protokollkészlet utasításait. Ha névjegy helyett nyelesfésűt adok az inasnak, akkor elveszi, majd odabent bedobja a kukába. Ha névjegy helyett revolvert adok neki, akkor kihívja a rendőrséget. A milliót csak a konkrét névjegyre hozza ki válaszul. Ha időközben bővül a protokoll utasításkészlete, akkor természetesen a proxy-t is utána kell okosítani - nehogymár a pendrive-ot is kidobja a kukába, ha egyébként a rajta lévő információ egyenértékű a névjeggyel.

Mit csinál eközben a NAT? Ész nélkül forgatja a címeket. Eszébe sem jut, hogy értelmezze, a megcélzott port vajon melyik protokollhoz tartozik.
Fel se néz az asztaltól, csak veri a billentyűket.

4.1.3 ICMP, AZAZ A PING ÉS A HAVEROK

RFC 792, 950, 1812, 1122, 1191, 1256

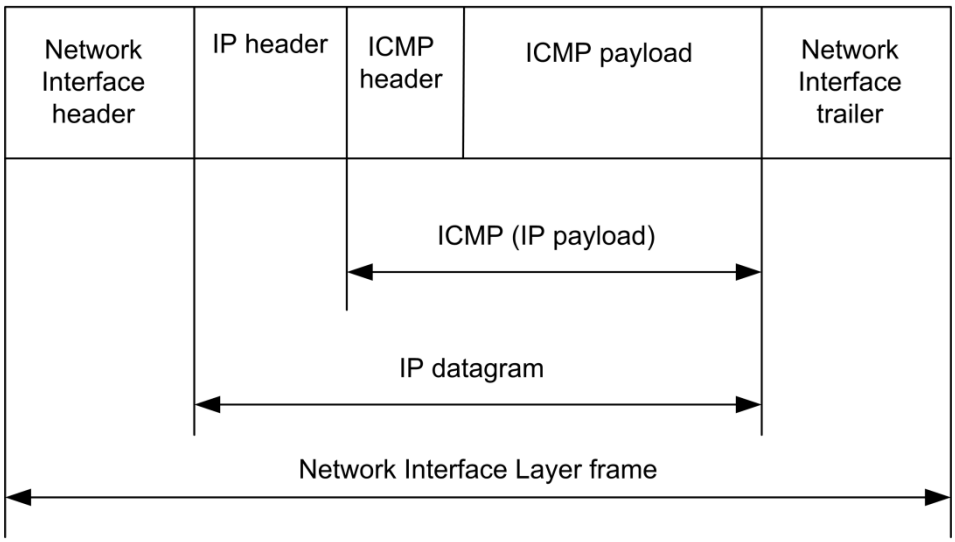
Később fogjuk látni, hogy a szállítási protokollok egész jól meg lettek szervezve: az egyik gyors, de nincs benne semmi hibavizsgálat, a másik pedig lassú, kicsit fecsegőbb - cserébe beépített hibaellenőrzést és javítást is tartalmaz. Miénk a választás, melyiket használjuk.
Az IP nem ilyen. Az IP headerben nincs semmilyen hibajelzési lehetőség.
Akkor mi van? ICMP - azaz az Internet Control Message Protocol.

Hol is helyezkedik el a csomagok hierarchiájában az ICMP? Vajon az IP alatt... mellette... vagy fölötte? Segítek: *4.2. ábra Egy IP csomag felépítése*.

Az IP datagramm áll IP header-ből és IP payload-ból. (Eddig az IP header-t boncolgattuk.)
Az IP payload lehet:

- TCP szegmens vagy UDP csomag,
- Egyéb IP protokoll datagram (GRE, AH, ESP, stb...)
- ICMP csomag,
- IGMP csomag.

Az ICMP csomag pedig - micsoda meglepetés - állni fog egy ICMP header-ből és egy ICMP payload-ból. Tiszta Matrjoska baba.



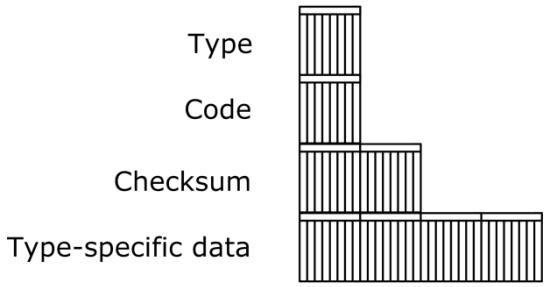
4.18. ÁBRA AZ ICMP ELHELYEZKEDÉSE A DOBOZOKBAN

Az ICMP csomagok fognak nekünk visszajelzéseket adni mind a routolási hibákról, érdekességekről, mind a csomagszállítási hibákról. Olyan az ICMP, mint a Négy páncélos és a kutya című filmben a kutya: normális esetben a négylábú csak utazott

a tankban, de ha Guszlikék bajba kerültek, akkor a kutyát küldték el egy üzenettel segítségért.

Persze ezt nem úgy kell elképzelni, hogy amikor visszament egy ICMP üzenet a feladónak, akkor az megismételt volna egy elveszett csomagot. Az a TCP. Jelen esetben az ICMP hibaüzenetre vagy a router/host értékel át bizonyos helyzeteket, vagy a host beletolja a rendszergazda képébe az üzenetet, mondván, hogy 'csinálj babám valamit, mert baj van'.

Merüljünk el a konkrétumokban.



4.19. ÁBRA EGY ÁLTALÁNOS ICMP CSOMAG

Az ábrán láthatjuk, hogyan néz ki egy teljesen általános ICMP csomag.

TYPE: Az üzenet típusa. A későbbiekben a következőkkel fogunk megismerkedni:

4.14. TÁBLÁZAT

ICMP Type	Közismert neve
0	Echo reply
3	Destination unreachable
4	Source Quench
5	Redirect
8	Echo Request
9	Router Advertisement
10	Router Solicitation
11	Time Exceeded
12	Parameter Problem
17	Address Mask Request
18	Address Mask Reply

Ennél van több is:
http://www.iana.org/assignments/icmp-parameters

CODE: A típuson belül az altípus.

CHECKSUM: CRC.

Eddig volt az ICMP header.

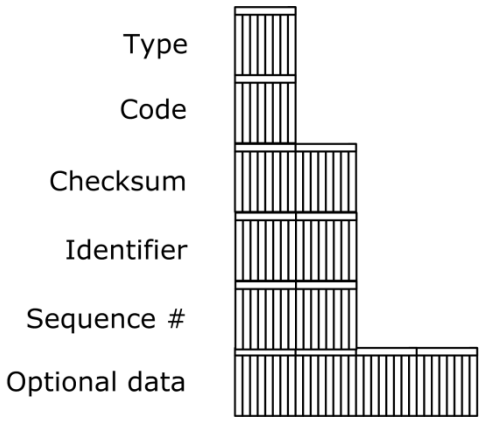
TYPE SPECIFIC DATA: Típustól függő adatok. Tulajdonképpen ez a payload.

4.1.3.1 ICMP ECHO REQUEST & REPLY

Ez az a parancs, melyet még gyerekeim nagymamája is ismer. A világ legegyszerűbb tesztje: *ping index.hu* -> és már tudjuk is, hogy vajon működik-e a netünk. Vagy ledosolták-e éppen az indexet.

Mint a tesztek legtöbbje, ez is két részből áll:

- Először belerúgunk a szőrkupacba. (ICMP Echo Request)
- Ha visszamorog, akkor még él a kutya. (ICMP Echo Reply)



4.20. ÁBRA AZ ICMP ECHO CSOMAGOK SZERKEZETE

Mind a két ICMP ECHO csomag hasonlóan strukturálja a TYPE SPECIFIC DATA mező szerkezetét.

TYPE: A pontos értékeket a *4.14. táblázat* tartalmazza.

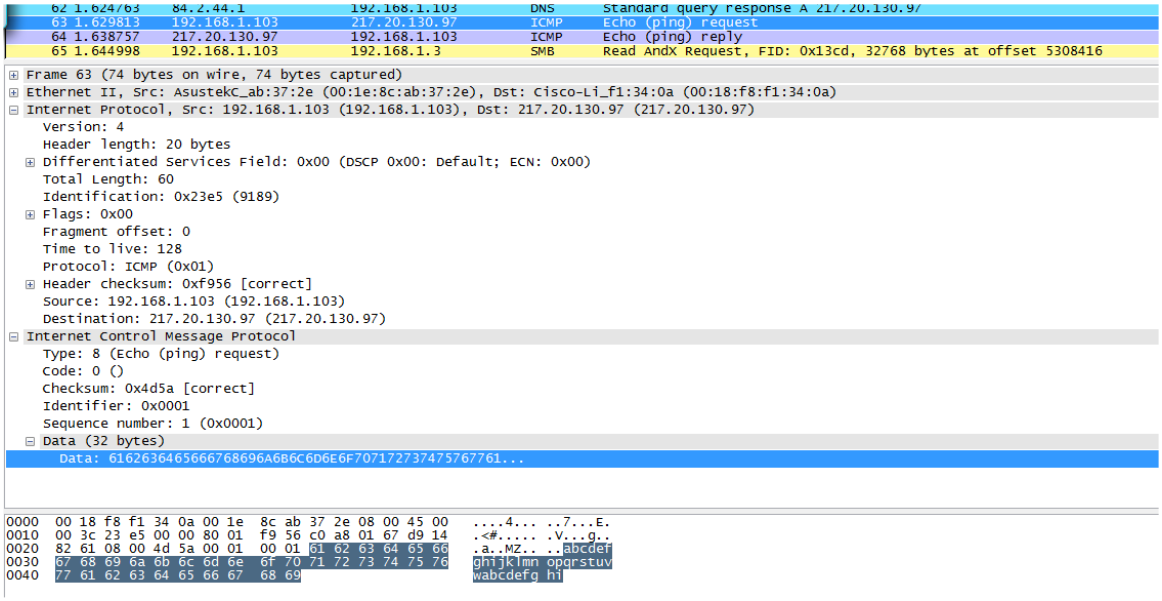
CODE: Nagy büdös nulla.

CHECKSUM: CRC.

IDENTIFIER, SEQUENCE NR: Ezek az azonosítók jelölik az összetartozó REQUEST-REPLY csomagokat.

OPTIONAL DATA: Valami ballaszt. Az életszerű teszthez kell adat is, amit át kell vinni.

Nézzük, mit mond a sztetoszkópunk.



4.21. ÁBRA ICMP ECHO REQUEST

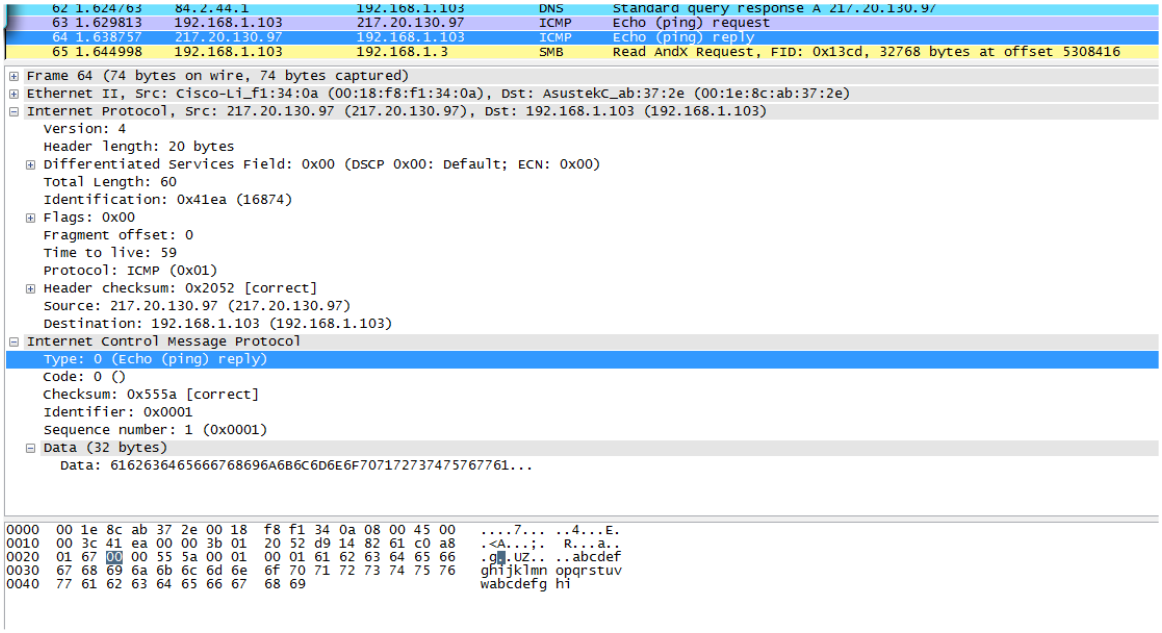
A képernyőkivágáson jól látható, amit az ábrán (*4.18. ábra Az ICMP elhelyezkedése a dobozokban*) is próbáltam érzékeltetni: az ICMP csomag az IP csomag payload-jában van - azaz megelőzi az egészet egy IP header.

Az ICMP csomagban a TYPE=8 jelzi a REQUEST-et, az azonosító szám értéke 1 - ugyanennyinek kell lennie a REPLY csomagban is. Végül a ballaszt, a DATA... na, mi is ez? Legalul látszik, hogy a mező az angol ABC betűivel sorfolytonosan töltődik fel, méghozzá úgy, hogy ha elfogynak a betűk, akkor kezdődik az ABC elölről.
Mekkora lehet a DATA mérete? Az alapértelmezett érték 32 bájt, a maximális érték pedig 65500. (Mint korábban is írtam, az IP csomag maximális mérete 65536 bájt[43].)

Ping of Death:
http://en.wikipedia.org/wiki/Ping_of_death

[43] A korai operációs rendszerek nagy részét remekül meg lehetett fektetni egy 65500 bájtnál nagyobb ping kiküldésével. Ez volt a ping of death. A felesleg ugyanis túlcsordult, bele a verembe. Aztán bof. De ezeket a lyukakat már a múlt évezredben befoltozták.

Végül így néz ki a fogadjisten.



4.22. ÁBRA ICMP ECHO REPLY

Ehhez az ábrához nincs sok hozzáfűznivalóm. Elégedett sóhajtással vehetjük tudomásul, hogy az IDENTIFIER és a SEQUENCE NUMBER mezők értéke megegyezik a korábbi REQUEST-ben lévő értékekkel, a TYPE jelenleg 0, a CODE masszívan tartja a zérót, a DATA pedig továbbra is az angol ABC-ből képződik.

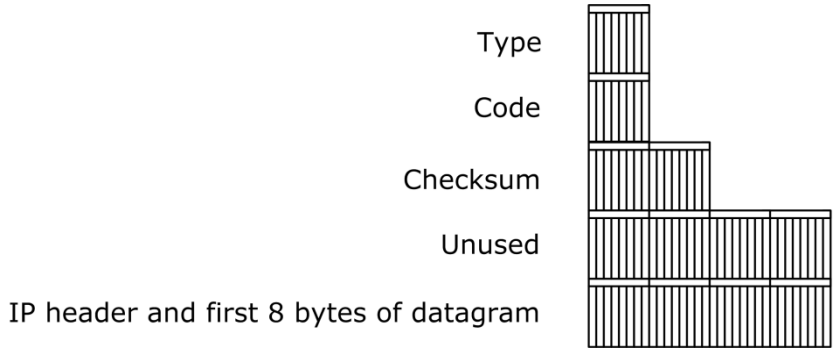
Gyerünk tovább, lesz ez még sokkal érdekesebb is.

4.1.3.2 ICMP DESTINATION UNREACHABLE

Az előbb láttuk, mi történik egy sikeres ping esetén. Most megnézzük, mi történik sikertelen próbálkozások esetén.
Nyilván visszamegy egy üzenet a sikertelenségről.

Hogyan is? Hiszen nem értük el a címzettet. Akkor ki küldi vissza az üzenetet? Természetesen az utolsó még elért hálózati elem. Illetve, a legutolsó elem, aki információval bír. (Elképzelhető ugyanis, hogy a megcélzott host elérhető ugyan, de például a megadott portja nincs nyitva.)

Az üzenet pedig egy ICMP csomag, méghozzá a 3-as tipusú, az ICMP Destination Unreachable.



4.23. ÁBRA ICMP DESTINATION UNREACHABLE

TYPE: Jelen esetben az értéke: 3.

CODE: Van néhány értéke. Lásd *4.15. táblázat*.

CHECKSUM: CRC.

UNUSED: Sorminta.

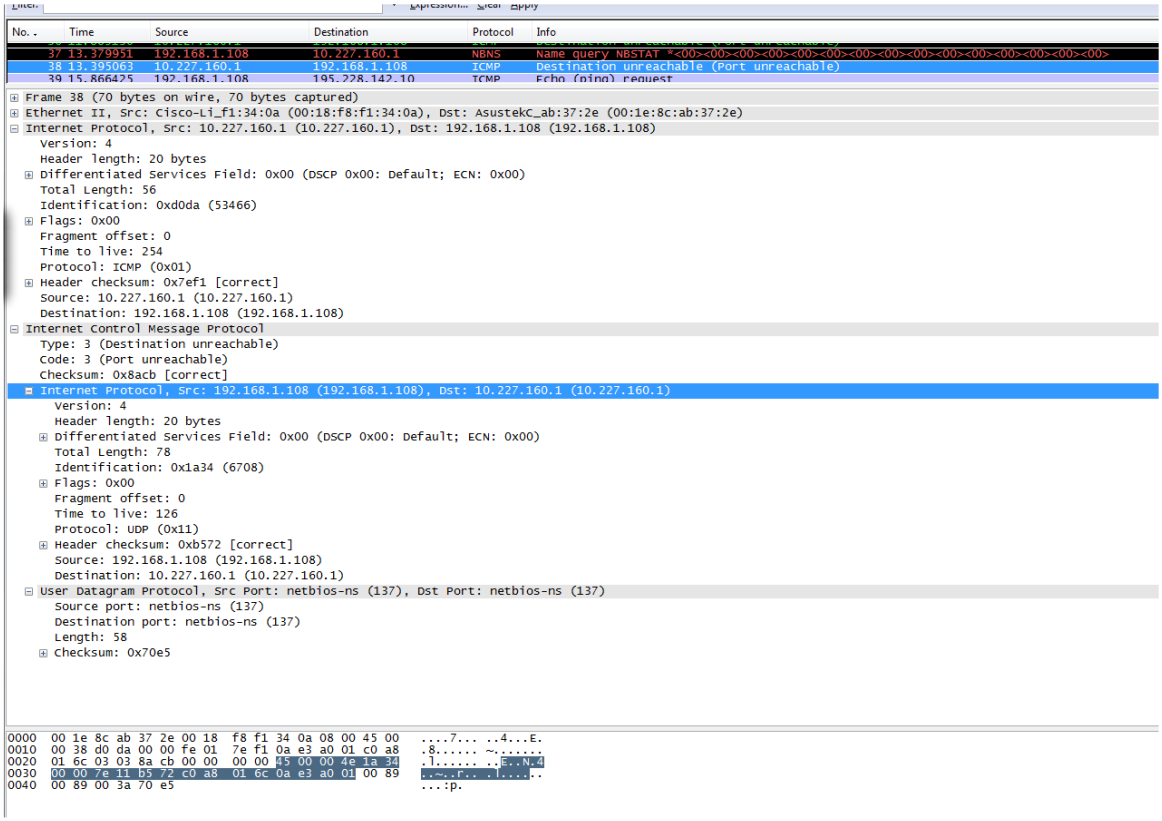
IP HEADER + FIRST 8 BYTE: Hogy valami legyen is abban a csomagban. Persze itt már van információdúsabb ballaszt is, mint az angol ABC. De ezt egy konkrét példán hamarosan láthatjuk.

4.15. TÁBLÁZAT

Code	Elnevezés	Megjegyzés
0	Network Unreachable	A visszafeleselő router route táblájában nincs a csomag címzettjének IP címére vonatkozó bejegyzés.
1	Host Unreachable	A visszafeleselő router route táblájában nincs a csomag címzettjének hálózatára vonatkozó bejegyzés.
2	Protocol Unreachable	A címzett host küldi vissza, ha a csomagban szereplő protokollazonosítót nem tudja értelmezni.
3	Port Unreachable	A címzett host küldi vissza, ha a csomagban szereplő porton nem figyel semmilyen szolgáltatás. Apró trükk, hogy ez az üzenet csak UDP szállítási protokoll mellett generálódik, TCP esetén - a később tárgyalandó Reset, azaz RST - elnyomja.
4	Fragmentation Needed and DF Set	A router küldi vissza, ha az MTU eltérések miatt tördelnie kellene a csomagot, de a DF (Don't Fragment) flag nem engedi.
5	Source Route Failed	Ha az IP Options rovatban előírtuk a kötelezően követendő útvonalat a routerek között, de az nem teljesíthető.
6	Destination Network Unknown	Ha a megcélzott host hálózatára vonatkozóan van ugyan bejegyzés a router route táblájában, de az valótlan. A gyakorlatban nem használják, a 0 kód megy vissza helyette.
7	Destination Host Unknown	Elméletileg ez az üzenet akkor keletkezik, ha a megcímzett host a Network Interface Layer címfelderítő mechanizmusai szerint nem létezik. Gyakorlatilag csak akkor használják, ha a router-hez PPP kapcsolaton keresztül kapcsolódó host/router érhetetlen el.
8	Source Host Isolated	Ha a feladó host el van szigetelve hálózatának többi részétől. Egyáltalán nem használják.
9	Communication with Destination Network Administratively Prohibited	Ha úgy egyébként létezik route bejegyzés a megcélzott munkaállomás hálózatával kapcsolatban, csak network policy tiltja a csomag továbbítását. A hétköznapokban nem használják, egyedül katonai célokra engedélyezett.
10	Communication with Destination Host Administratively Prohibited	Ha network policy tiltja a csomag továbbítását a megcélzott host számára. A hétköznapokban nem használják, egyedül katonai célokra engedélyezett.
11	Network Unreachable for the Type Of Service	Amennyiben a router úgy van beállítva, hogy használja a TOS mezőt (emlékszünk, *4.1.1* fejezet, leginkább sávszélesség-szabályozás.), de a csomagban nem talál ehhez szükséges infót és emiatt nem éri el a címzett hálózatot.
12	Host Unreachable for Type Of Service	Amennyiben a router úgy van beállítva, hogy használja a TOS mezőt, de a csomagban nem talál ehhez szükséges infót és emiatt nem éri el a címzett hostot.
13	Communication Administratively Prohibited	A routeren packet filter tiltja a csomag továbbítását. Elméletileg használják, a gyakorlatban inkább nem - túl direkt válasz lenne a rosszindulatú portszkennelésekre.
14	Host Precedence Violation	Amennyiben a TOS mezőben megadott Precedence érték nem engedélyezett.
15	Precedence Cutoff in Effect	Amennyiben a TOS mezőben megadott Precedence érték alacsonyabb a minimálisan megengedettnél.

Destination Unreachable Messages:
http://www.tcpipguide.com/free/t_ICMPv4DestinationUnreachableMessages-3.htm
http://www.networkcomputing.com/netdesign/1107icmp3.html?ls=NCJS_1107bt

Nézzünk meg konkrétan is egy üzenetet.



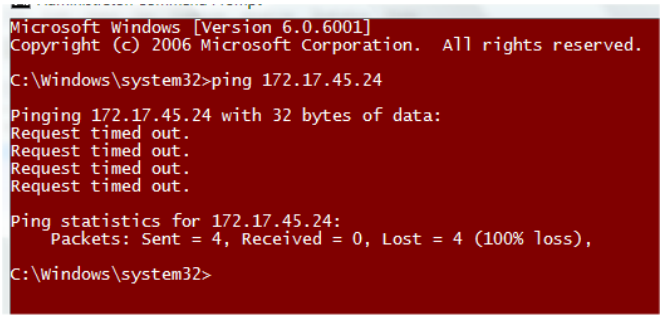
4.24. ÁBRA ICMP DESTINATION UNREACHABLE - PORT UNREACHABLE

Itt a következő történt: a 192.168.1.108 host meg szerette volna tudni a 10.227.160.1 host netbios nevét egy *netbt -A 10.227.160.1* parancs kiadásával. Erre válaszolta azt a megcélzott host, hogy bocs haver, de nekem a 137-es UDP portomon nem figyel semmilyen szolgáltatás sem. (Valószínűleg 'X'-re végződik a host operációs rendszerének a neve.)

Látható, hogy az ICMP payload gyakorlatilag mindent elmond. Ott van benne a _*kérés*_ teljes IP header-e, illetve az IP payload-jának első nyolc bájtja. A transport rétegnél majd látni fogjuk, melyek is ezek a bájtok - de remélem, most sem kell a szomszédba menni logikáért. Elhangzott már a könyvben, hogy egy port értéke 0-65535 közötti érték lehet, ez egész pontosan két bájton ábrázolható. A length és a checksum szintén 2-2 bájtos mezők - és már meg is vagyunk. Ezek kerülnek bele - az IP header mellett - az ICMP payload-ba. (A capture fájlban az eredeti kérés(37) közvetlenül az ICMP csomag(38) előtt látható, tehát itt most nem sokat segített az ICMP payload. De nincs ez mindig így.)

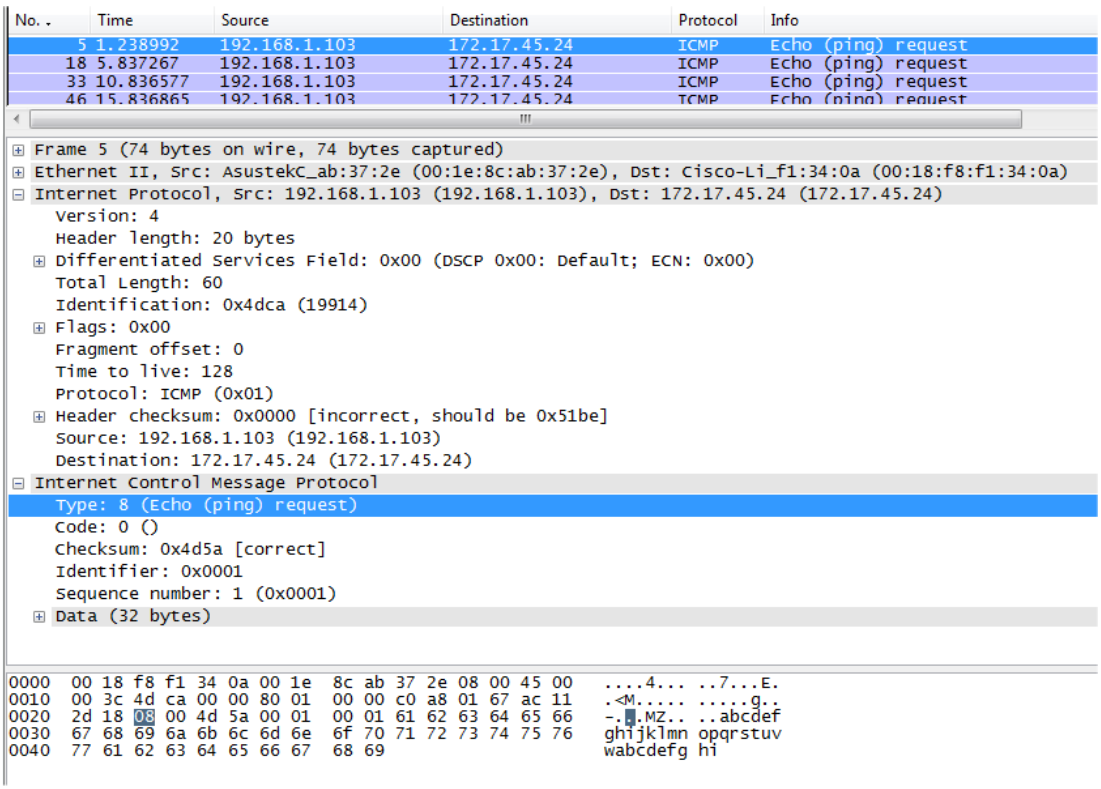
Végül, oldva a túl száraz leírást, elmesélem, hogyan sikerült rögzítenem ezt a capture fájlt.

Előljáróban megjegyezném, hogy az ICMP Destination Unreachable üzenet egy átlagos hálózatban ritka, mint a fehér holló. Legtöbbször ugyanis a hálózati hibák vagy timeout, vagy RST formájában jelentkeznek.
Nos, elgondolkodtam. A Destination Host Unreachable üzenet ismerős volt, gondoltam, nem lesz nehéz reprodukálnom. Nosza, pingeljünk meg egy nem létező hostot!



4.25. ÁBRA REQUEST TIMED OUT

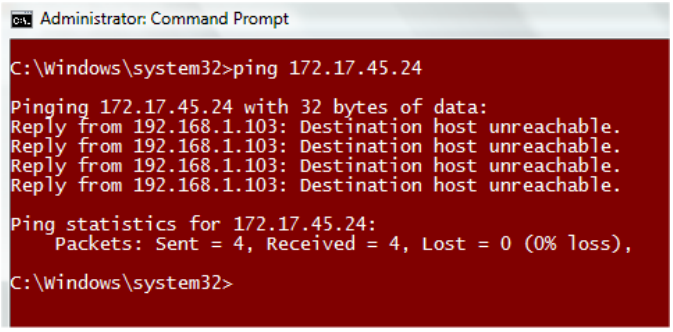
Nem túl biztató. Mi is történik közben?



4.26. ÁBRA REQUEST TIMED OUT - A HÁTTÉRBEN

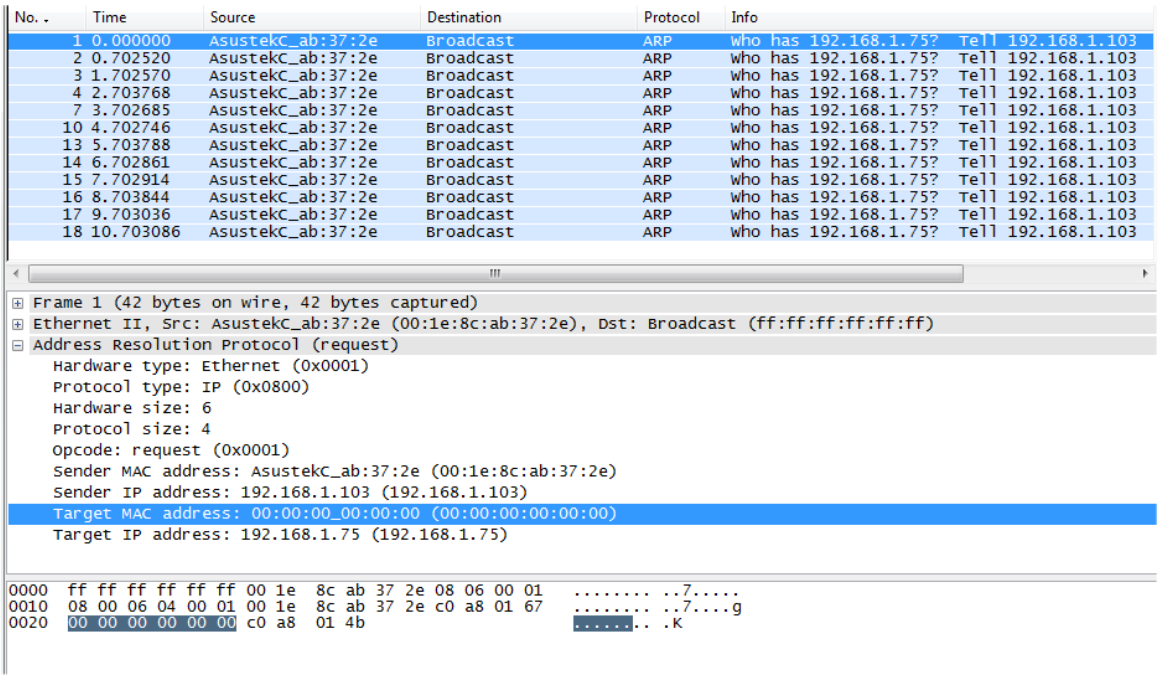
Semmi. Kiadunk egy ICMP Request kérést, várakozunk, majd tudomásul vesszük, hogy még válaszra sem méltatnak. De azért bepróbálkozunk még háromszor.

Pedig istenbizony, szoktam látni munka közben ilyen üzeneteket. Mikor is volt utoljára ilyen...? Megvan: amikor elírtam egy szerveren a default gateway értékét! Nosza, át is írtam a gépemen lévőt 192.168.1.75-re.



4.27. ÁBRA DESTINATION HOST UNREACHABLE

És igen, itt van. Ezt kerestük. Capture fájl:



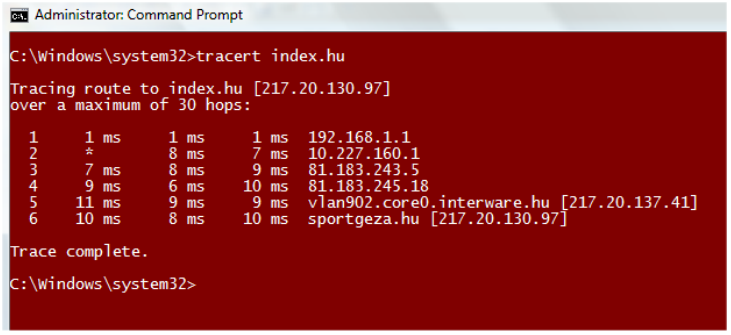
4.28. ÁBRA DESTINATION HOST UNREACHABLE - A HÁTTÉRBEN

Vagy mégsem? Egy nyomorult ICMP üzenet sincs benne, a számítógépem vad ARP broadcast viharokkal próbálja megtudni a kamu default gateway MAC Address-ét - majd egy idő után elunja.
Ez sem az igazi.

Nyilván, ha létező IP címet írok DGW-nek, mely persze nem egy router címe, akkor megint Request Timed Out jön vissza.

Oké. Játsszunk el a router packet filterével. Gondoltam, vagy a 2-es, 3-as... vagy a 13-as kód (*4.15. táblázat*) csak meglesz.
Itt már nem fárasztalak képekkel, nem lett meg. Gyakorlatilag minden kísérletre RST-t kaptam vissza. Megpróbáltam félrevezetni a csomagot a Strict Source Routing segítségével (*4.1.1.2 IP Options*) de a routerem nem támogatta ezt az opciót.

Végül teljesen váratlanul mosolygott rám a szerencse. Megtrészeltem az index.hu-t.



4.29. ÁBRA TRACERT INDEX.HU

Ránézésre semmi különös, a folyamat sikeresen lezajlott. Azért ránéztem a capture fájlra... és ebben találtam meg a korábbi ábrán (*4.24. ábra ICMP DESTINATION UNREACHABLE - PORT UNREACHABLE*) bemutatott ICMP üzenetet.

Szemöldököm nyilván egyből a plafonig szökött: ez most akkor hogyan? Mi is ment itt félre? Amikor minden jó?

Ehhez ismerni kell egy kicsit a *tracert* lelkivilágát.

E mögött a parancs mögött tulajdonképpen *ping* parancsok bújnak meg. Mint korábban is írtam, a tracert egyáltalán nem a Record Route nevű IP opciót használja egy útvonal állomásainak rögzítésére. Pingel. Csak éppen mindig eggyel nagyobb TTL értékkel - azaz minden ping a következő hop címével jön vissza.

De a tracert nem elégszik meg ennyivel. Minden egyes hop esetén igyekszik begyűjteni az IP cím mellé a host nevét is. (Lsd. *4.29. ábra Tracert index.hu*.) Kedves tőle. Név viszont kétfajta van: egyrészt találhatók nevek DNS zónákban, másrészt van magának a hostnak is neve, melyet Windows-t futtató gépen rövid, azaz Netbios névnek hívunk. A tracert bepróbálkozik mindkét névfeloldással... és erre a próbálkozásra kapta azt a választ a 10.227.160.1 hosttól, hogy azon bizony nem fut Netbios Name Service szolgáltatás, azaz az UDP 137-es portja mögött nem dolgozik senki. Gyors ellenteszt: kiadtam a *netbt -A 10.227.160.1* parancsot - és ugyanúgy megjöttek a várt ICMP üzenetek a capture fájlban.
Vedd észre a szituáció szépségét. Nem tudtam elérni egy portot, erre kaptam vissza az ICMP hibaüzenetet. De éppen nemrég írtam, hogy eljátszottam a router packet filterével - és mégsem jöttek a várt üzenetek. Akkor most mi van?

A kulcs a szállítási protokollnál van. Amikor packet filterrel játszottam, akkor a *telnet 192.168.1.1 port_number* parancsokkal próbálkoztam - a telnet viszont TCP. Mit is ír a *4.15. táblázat* a 3-as kódnál? TCP esetében az RST elnyomja az ICMP-t. Mivel is próbálkozik a Netbios névfeloldás? Úgy van, UDP-vel.

Tanulság:
Az ICMP Destination Unreachable üzenetek nem is annyira gyakoriak, mint ahogy azt az ember első ránézésre gondolná.

4.1.3.2.1 PMTU

Ebben az alfejezetben érdemes pár szót ejtenünk a kicsit misztikus PMTU-ról is. A betűszó konkrétan a Path MTU, azaz Path Maximum Transmission Unit kifejezés kezdőbetűiből áll össze. Az MTU-ról esett már szó a könyvben, hasonlóképpen a fontosságáról is. Tudjuk, hogy az MTU az adott hálózatban átvitt csomagokban a maximális hasznos teher (payload) méretét jelenti, NIL szinten. Értékének ismerete azért fontos, mert ha nagyobb payload-dal rendelkező csomag érkezik, akkor azt bizony tördelni kell - és mi már azt is tudjuk, hogy a tördelés nagyon egészségtelen művelet: zabálja az erőforrásokat, bonyolítja az életet és molesztálja a nyugdíjasokat.
Értelemszerűen az igazi az lenne, ha a forrásként viselkedő hostunk már a csomag összerakásakor tudná, hogy a célpontig vezető útvonalon mennyi az egyes alhálózatok MTU értéke. Ekkor ugyanis ő maga is a legkisebb MTU-ra méretezné a csomagot, így nem kellene egyszer sem tördelni azt az út során.

RFC 1191

Ezt a konkrét útvonalra vonatkozó MTU értéket nevezzük PMTU-nak. Meghatározásához pedig az ICMP Destination Unreachable üzenetet használjuk, egész konkrétan a 4-es kóddal bírót. Persze ehhez egy kicsit át kell alakítanunk a csomag szerkezetét. (A PMTU detektálással a 1191-es RFC foglalkozik, az ezt ismerő eszközök az ún. RFC 1191 compliant eszközök.)

Ne gondolj túl nagy átalakításra. A korábbi ábrán (*4.23. ábra ICMP Destination Unreachable*) látható egy négy bájtos sorminta. A módosított csomagban ebből elveszünk két bájtot és ide tároljuk le a NEXT HOP MTU értékét - azaz azt, hogy a következő alhálózatban mekkora lesz az MTU érték.

Nem túl bonyolult, ugye? Már csak arról kell gondoskodnunk, hogy egész biztosan jöjjön is ICMP visszajelzés. (Ne feledjük, az ICMP Destination Unreachable üzenet alapvetően azért hibaüzenet.) Ezt úgy érhetjük el, hogy beállítjuk a kezdeti MTU értékét a lokális hálózatunkon használt értékre, a DF flag-et magasra billentjük - és elküldjük a csomagot. (Ezt hívjuk PMTU detektálási kezdeményezésnek.) Ha a csomag így is végigment visszajelzés nélkül, akkor nagyon jó: az alhálózatunk MTU értéke egyben a PMTU is. (Vagy nem... de erről később.) Ha valahol tördelni kellett volna, de a DF-fel ezt ugye letiltottuk, akkor 4-es kódú ICMP Destination Unreachable üzenet jön vissza. Szerencsés esetben már az újabb szerkezettel - amelyikben már létezik mező a megkívánt MTU érték számára. Újracsomagolunk, immár az új MTU értékkel.

Ügyes.

De mi van akkor, ha - akár csak - egy router is buta az útvonalon és nem ismeri a 1191-es RFC-t?

Két eset lehetséges:

- A router visszaküldi a 4-es ICMP Unreachable Destination üzenetet, de még a régi fajtát. Ekkor a feladótól függ a további stratégia. Besaccolhat egy értéket, aztán kisérletezgethet vele tovább, de mehet egyből tutira is, a legkisebb, még értelmes értékre (576 bájt) állítva az MTU-t.
- A router még ostobább, még a rég ICMP üzenetet sem küldi vissza. Ezt hívják feketelyuk routernek. Ekkor egy kicsit cifrább az eljárás. A feladónak azért egy idő után fel fog tűnni, hogy nem érkeznek meg a hibafelderítési felhanggal küldött csomagjai. A TCP csomagoknál például kapnia kellene egy visszaigazolást. Ha ez nem történik meg, akkor a feladó megpróbálkozik a DF nullára állításával. Amennyiben ez után már visszajön a visszaigazolás, akkor érzékeli, hogy MTU problémával és feketelyuk routerrel áll szemben. Innentől a stratégia már ugyanaz, mint az előbbi pontban, azzal az apró eltéréssel, hogy a sikertelen MTU próbálkozást nem az ICMP üzenet fogja jelezni a feladó számára, hanem a TCP visszajelzés elmaradása.

A Windows Server 2008 és a Vista routerként viselkedve RFC 1191 kompatibilisek, feladó hostként viselkedve pedig helyből PMTU detektálósak. Ezeket az attitüdöket registry-ből módosíthatjuk - de minek?

PMTU:
http://en.wikipedia.org/wiki/Path_MTU_discovery
http://www.netheaven.com/pmtu.html

4.1.3.3 ICMP Source Quench

Köszönöm, elegem van. Böff.

Mondja ezt a router, miután minden testnyílásán keresztül telenyomták csomagokkal, ő pedig nem tudja ebben a tempóban feldolgozni és továbbküldeni azokat.

Az ICMP Source Quench egy olyan üzenet, melyet egy túlterhelt router küld vissza a csomagok feladóinak. Az üzenet lényege: lassítsatok, fiúk!

Igenám, de van ezzel egy kis baj:

- Vannak olyan fiúk, akik képtelenek lassítani. Az ICMP üzenet ugyanis IP szintű - viszont az internet layer-ben nincs forgalomszabályozás. Egyedül a TCP-ben van, mely viszont már a transport réteg. Egyedül a konkrét implementációtól függ, hogy figyel-e az egyik réteg a másik réteg problémáira.
- Az ICMP Source Quench üzeneteket el is kell küldeni. Pont annak a routernek, amely egyébként is erőforrás problémákkal küszködik. Olyan ez, mint amikor veszettül dolgozol egy projekten, látod, hogy nem leszel kész határidőre, segítséget kérsz a főnöködtől - aki jelentést irat veled arról, miért haladsz lassan. Oké, hogy a legtöbb implementációban a túlterhelt router nem küld *minden* csomagra ICMP Source Quench választ - de ez csak hajszárító a Niagara ellen.
- Eddig csak jóindulatú hozzáállásról beszéltünk. De mi van a rosszfiúkkal? Ha én azzal keresném a kenyérrevalót, hogy weboldalakat dosolok[44] le, akkor az első ICMP Source Quench csomag láttán kéjes vigyor öntené el az arcomat és fülig érő szájjal fokoznám tovább a tempót - hiszen már karnyújtásnyira vagyok a célomtól.

Szóval, szép, szép ez a visszajelzős móka... de nem egyértelműen hasznos. A WIndows Server 2008 / Vista implementációkból teljesen ki is maradt: a routerként viselkedő szerver nem küldözget ilyesmit vissza, a kliensként működő operációs rendszerben pedig az IP réteg nem veszi a fáradtságot, hogy zavarja vele a TCP csatornát.

A csomag felépítése egyébként teljesen megegyezik az ICMP Destination Unreachable csomag felépítésével. (*4.23. ábra ICMP Destination Unreachable*)

[44] DOS: Denial of Services. Azaz túlterheléses támadás.

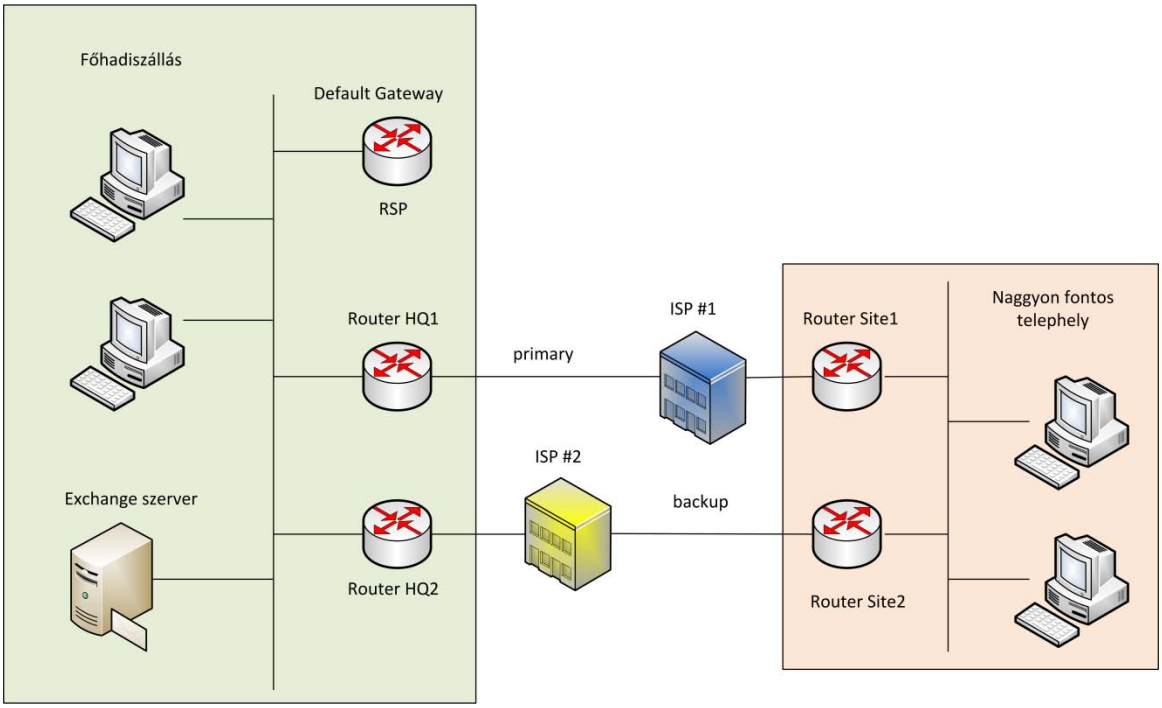
4.1.3.4 ICMP REDIRECT

Személyes vallomással kezdem: utálok mindent, ami megpróbál okosabb lenni nálam. Nem azért, mert ehhez még csak meg sem kell erőltetnie magát - hanem azért, mert ha valami nem tudja biztosan, hogy mit akarok, akkor minden variálás nélkül egyszerűen csak csinálja azt, amit mondok neki. Ifjú informatikusként viszolyogtam a PnP-től, az automatikus hálókártya konfigurálástól - és a Word telepítése után is az az első teendőm, hogy a tools/options alatt kikapcsolok mindent, ami úgy kezdődik, hogy auto.

Nos, az ICMP Redirect pont ilyen kellemetlen funkciót lát el.

Elmesélek egy sztorit.

Nagy ügyfél. Magas rendelkezésre állás. A főhadiszállás és a nagyon fontos telephely földrajzilag is távol vannak egymástól. A kettő között bérelt vonal feszül, nem is egy: van egy elsődleges és van egy backup vonal. Eltérő szolgáltatóktól, természetesen.



4.30. ÁBRA A JÁTSZÓTÉR

A főhadiszálláson mind a kliensgépek, mind az Exchange szerver route táblája olyan pőre volt, amilyen egy route tábla csak lehetett. Egyedül a Default Gateway címe volt benne mindegyikben. Ez egy Route Switch Processor (RSP) jellegű masina volt, azaz ő ismert minden útvonalat, ő mondta meg mindenkinek, hová menjen. Ha például

egy főhadiszállásbéli kliens el akart érni egy klienst a telephelyen, akkor elküldte a csomagjait az RSP-hez, az átlökte azokat az elsődleges routerhez, onnan pedig már mentek is tovább. Ha üzemzavar miatt ledőlt az éles vonal, akkor az RSP a backup routerhez küldte tovább a csomagokat. Ment minden szépen... amíg ki nem lett cserélve a vas az RSP alatt. Az újon ugyanis elfelejtették letiltani az ICMP Redirect funkciót.

Ekkor indult be a rock'nroll.

Első üzemzavar. Ügyfél reklamál, hogy a naggyon fontos telephelyről nem érik el az Exchange szervert. Hálózatos ember átvizsgálja a rendszert - minden rendben. Exchange rendszergazda átvizsgálja a szervert - minden rendben. Akkor most mi van?

Hát az ICMP Redirect betette a lábát a rendszerbe. Ez a szolgáltatás ugyanis a hálózati forgalom optimalizálására hivatott. Úgy működik, hogy ha egy kliens küld egy csomagot egy routernek és a router érzékeli, hogy a következő hop is még ugyanezen az alhálózaton van, akkor küld vissza a kliensnek egy ICMP Redirect üzenetet, mondván, hogy 'barátom, legközelebb küldheted direktben is oda a csomagot, ne terheljél már, ha van egyszerűbb út is'.
A kliens fogja az ICMP Redirect csomagot, kiveszi ~~a vízből az oxigént~~ belőle a másik router címét, és bejegyzi a saját route táblájába, hogy ha legközelebb ehhez a szerverhez kell mennie, akkor már ne a Default Gateway-t zaklassa.

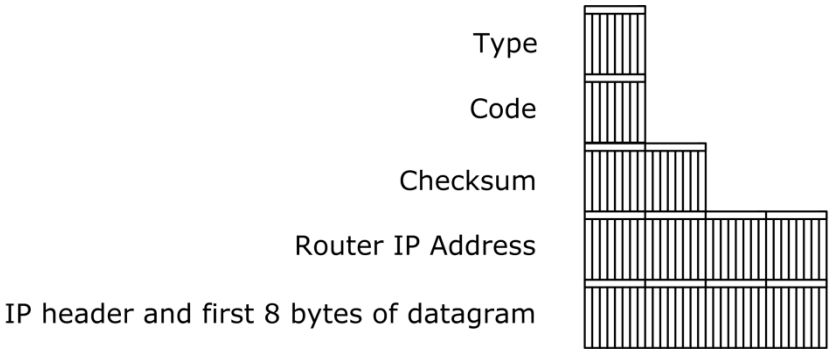
Nézzük most akkor a példát. A telephelyről egy kliens rácsatlakozott az elsődleges vonalon az Exchange szerverre. Küldött neki egy csomagot, a szerver pedig válaszolt. Hogyan? Mivel a Default Gateway címe az RSP volt, elküldte annak a válaszcsomagot. Az meg visszaszólt, hogy van ám rövidebb út, a Router HQ1-en keresztül. A szerver boldogan jegyezte be a route táblájába a telephelyi kliens IP címét az elsődleges router - mint következő hop - IP címével egyetemben.. Innentől a csomagok az RSP kikerülésével mentek a Router HQ1 felé. (Ismétlés: ugye egy hostra vonatkozó route bejegyzésnél a subnet mask 255.255.255.255, míg a Default Gateway subnet mask bejegyzése 0.0.0.0. Habár mindkettő 32 pontot fog elérni az illesztési vizsgán, de a subnet mask egyeseinek száma miatt a hostra vonatkozó - azaz az ICMP Redirect indikálta - bejegyzés fog nyerni.)
Ezzel nem is volt addig baj, amíg az éles vonal működött. De amikor ledőlt, az RSP hiába tudta, hová kellene menni helyette, az Exchange szerver már le se tojta. A saját route táblája alapján nyomult lógó nyelvvel az elérhetetlen Router HQ1 routerre.

Külön szépsége volt az esetnek, hogy a bejegyzések nem magába a route táblába történtek, hanem a route cache-be - azaz a bejegyzések nem voltak perzisztensek -

így az újraindítás mindig megoldotta a hibát. Mi meg csak álltunk ott, a fejünket vakarva.

A történetnek nem az a tanulsága, hogy hány helyen hibáztunk. Átgondoltabb tervezéssel nyilván el lehetett volna kerülni ezt a hibát.
Az igazi tanulság az, hogy tudjál erről a lehetőségről. Amikor megtervezel egy bonyolult hálózatot, benne tömérdek routerrel, akár RIP, akár OSPF dinamikus routolással... ne felejdkezz el róla, hogy létezik még egy mechanizmus, mellyel foglakoznod kell: vagy le kell tiltanod, vagy be kell építened az elképzeléseidbe.

Vizsgáljuk meg akkor most már ezt a csomagot.



4.31. ÁBRA ICMP REDIRECT

TYPE: Jelen esetben 5.

CODE: Több értéke is lehet.

4.16. TÁBLÁZAT

CODE	Jelentése
0	Egy egész hálózatra vonatkozó átirányítás (Elsorvadt.)
1	Egy hostra vonatkozó átirányítás
2	Egy egész hálózatra vonatkozó átirányítás, amennyiben TOS-t is használunk.
3	Egy hostra vonatkozó átirányítás, amennyiben TOS-t is használunk.

ROUTER IP ADDRESS: Az a cím, ahová az érintett csomagokat mostantól küldeni kell.

Egy kérdés maradt már csak hátra: mit szól ehhez a Windows Server 2008 / Vista?

Nos, alaphelyzetben be van kapcsolva ez a viselkedési mód, de kikapcsolható:

- *netsh interface ipv4 set global icmpredirects=enabled* v. *disabled*

ICMP Redirect:
http://en.wikipedia.org/wiki/ICMP_Redirect_Message

4.1.3.5 ICMP Router Discovery

Egy újabb mechanizmus, mely belenyúlkál az IP routolásba. Ha röviden össze szeretném foglalni, miről is van szó, azt kellene mondanom, hogy olyan mechanizmusról van szó, melynek segítségével a hostok megtalálják a számukra ideális default gateway-t. És még csak véletlenül sem ejtettem ki azt a szót a számon, hogy DHCP (Dynamic Host Configuration Protocol), vagy ES-IS (End System to Intermediate System Routing Protocol). (Amilyen hosszú kifejezésekről van szó, véletlenül nem is lehetne a nevüket kiejteni.) De még csak azt sem mondtam, hogy routing protokoll. Nem, kérem szépen, a történet csak DGW keresésről szól.

ES-IS:
http://www.javvin.com/protocolESIS.html

Azt írtam volna, hogy 'mechanizmusról'? Akkor hibáztam. Ugyanis mechanizmus*ok*ról van szó.

- ICMP Router Advertisement: A router ezerrel reklámozza magát.
- ICMP Router Solicitation: A hostok nyaggatják a routereket.

Vad egy kicsit, nem? Beállítasz egy default gateway értéket, aztán egy fél óra múlva vagy a router dumálja le a hostról és állít be egy másikat - vagy a host hívja ki maga ellen a sorsot, eldobva a beállított default gateway értéket, egy szebb és csinosabb értékért.
És még csak nem is figyelmeztettek előre, hogy ilyenek történhetnek.

Nyilván nem erről van szó. Lehetőségekről beszélünk, melyek alapértelmezésben a Windows Server 2008 / Vista rendszerekben igencsak takarékra vannak kapcsolva. De attól még érdemes megismerkedni velük.

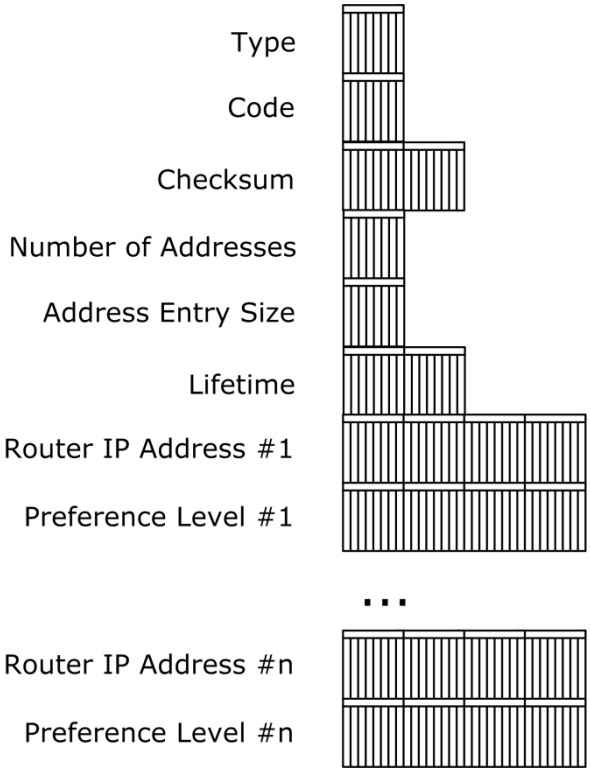
ICMP Router Discovery Protocol:
http://www.networkdictionary.com/protocols/irdp.php

4.1.3.5.1 ICMP Router Advertisement

Ezt az üzenetet a routerek küldik. Ha senki sem ingerli őket, akkor kvázi véletlenszerű időzítéssel (7-10 perc), ha bejelentkezik egy host az ICMP Router Solicitation üzenettel, akkor viszont kapásból. (Meg, hát!)

Hová küldik? Ez is változó. Van olyan router, aki az all host multicast IP címre (224.0.0.1) és van olyan router, aki a subnet broadcast címére. (Az RRAS-ba épített ICMP Router Discovery az all host multicast IP címet preferálja.)

Mit küldenek? Ezt.



4.32. ábra ICMP Router Advertisement

Type: Jelen esetben 9.

Code: Zéró. Nincs aleset.

NUMBER OF ADDRESSES: Ez egy kicsit furcsa lehet, hiszen a Default Gateway azért Default, hogy minden arra menjen, amire nincs konkrét szabály. Hogyan lehet akkor mégis több belőle? Úgy, hogy biztosra akarunk menni. A Default Gateway-k sem élnek örökké. Simán lehet olyan hálózatunk, amelyben az első hop két különböző router is lehet. Vagy mondjuk ugyanazon router több lábbal is áll az alhálózaton.
Száz szónak is egy a vége, ez az érték az a bizonyos 'n'.

ADDRESS ENTRY SIZE: Hány szóból állnak az egy-egy routerre vonatkozó bejegyzések. Mivel mind az IP cím, mind a preferenciaszint 4-4 bájtos értékek, így 4*2*8(bit)/32(bit) = 2 lesz a konstans érték.

LIFETIME: Hány másodperccel korábban volt a legutóbbi ICMP Router Advertisement?

ROUTER IP ADDRESS #N: Szépen sorban jönnek a kiajánlott Default Gateway IP címek.

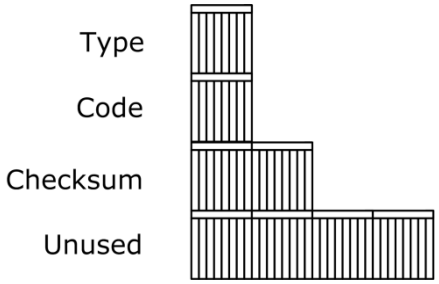
PREFERENCE #N: Az adott IP cím preferenciája... magyarul mennyire szeresse a host az illetékes DGW-t. Minél nagyobb a szám, annál jóképűbb a router.

4.1.3.5.2 ICMP ROUTER DISCOVERY

Gondolom, nem lepődik meg senki, amikor elárulom, hogy ezt az üzenetet a hostok küldik, méghozzá vagy az all router multicast IP címre (214.0.0.2) vagy a subnet broadcast címére. (A Windows Server 2008 /Vista megint a multicast-ot szeretik.)

Mikor? Amikor egy routertől ICMP Router Advertisement csomagot kapnak. (Amennyiben az ICMP Router Discovery engedélyezve van egy hoston és nem adunk meg konfiguráláskor Default Gateway-t, akkor a host szorgalmas lesz és magától is küld ICMP Router Solicitation csomagokat.)

Mit küldenek? Ezt:



4.33. ÁBRA ICMP ROUTER SOLICITATION

Látható, hogy abszolút nincs túlvariálva. Ez szimplán egy szemérmetlen fenékriszálás, mely a hajlandóságot jelzi - semmi több.

TYPE: Jelenleg 10.

CODE: Nulla.

Windows Server 2008 / Vista alatt a következő paranccsal kapcsolgathatjuk:

```
netsh interface ipv4 set interface <interfaceNameOrIndex> ↵
routerdiscovery=enabled v. disabled v. dhcp
```

Az alapértelmezett kapcsoló a *dhcp*. Ez a DHCP szervertől teszi függővé az ICMP Router Discovery engedélyezését: ha a szerveren meg van adva a Perform Router Discovery opció (kódszám=31), akkor mehet - ha nincs, akkor nem.

4.1.3.6 ICMP TIME EXCEEDED

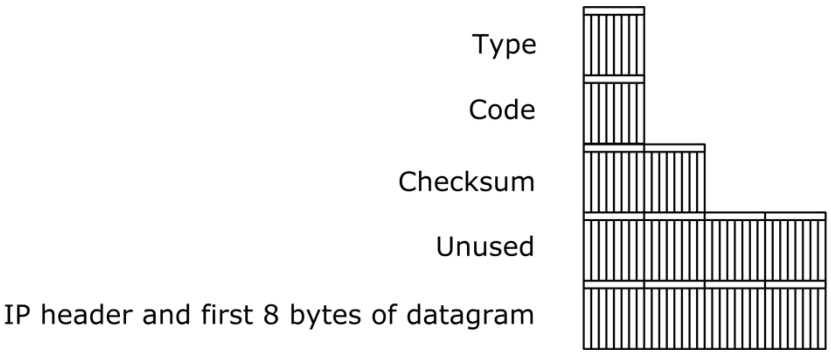
Idő van.
Pontosabban, nincs. Elfogyott.

Alapvetően rengeteg olyan pont van, ahol elfogyhat az idő, de ne feledjük, most az IP rétegben vagyunk. Két időérzékeny feladatunk van:

- Csomagokat összerakni.
- Csomagokat routolni.

Az első tényleg idő dimenziójú. Ha a fogadó host egy bizonyos idő alatt képtelen összerakni egy szanaszét töredezett csomagot - mert mondjuk nem érkezett meg egy darab - akkor egy *ICMP Time Exceeded - Fragment Reassembly Time Exceeded* üzenettel reagálja le a helyzetet.

A routernél kicsit más a helyzet. Minden csomagnak van egy TTL értéke, mely tulajdonképpen 'alkalom' dimenziójú: induláskor a csomag kap egy értéket és minden router, amelyiken átmegy, levesz belőle egyet. Aztán az a router, amelyiknél nullára fut a számláló, visszaküld egy *ICMP Time Exceeded - TTL Exceeded* üzenetet.



4.34. ÁBRA ICMP TIME EXCEEDED

Látható, hogy nagy csodák nincsenek, a csomag felépítése teljesen megegyezik az ICMP Destination Unreachable csomagéval.

TYPE: 11.

CODE:

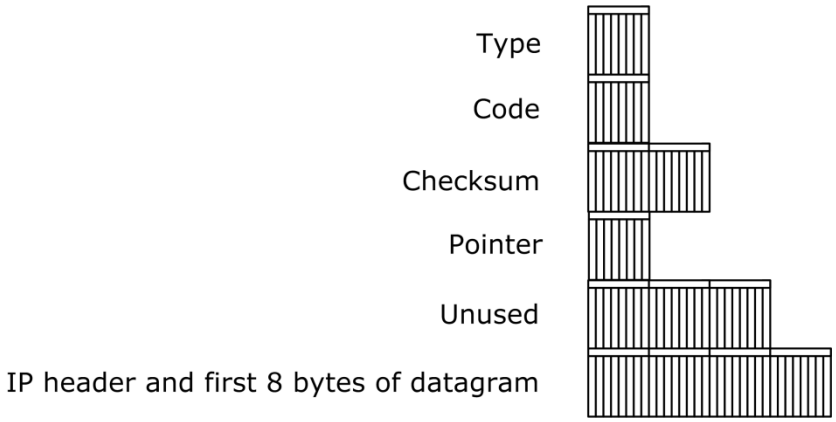
- 0: ICMP Time Exceeded - TTL Exceeded
- 1: ICMP Time Exceeded - Fragment Reassembly Time Exceeded

4.1.3.7 ICMP PARAMETER PROBLEM

Több feltételnek kell együtt bekövetkeznie, hogy ezt az ICMP példányt levadászhassuk[45]:

- A host ne tudja értelmezni a csomagban lévő IP fejlécet.
- Az ICMP üzenetek széles skálájából egyet se lehessen ráhúzni az esetre.

Azaz ez a 'valami baj van, de fogalmam sincs, hogy mi' üzenet.



4.35. ÁBRA ICMP PARAMETER PROBLEM

A szokásos ICMP csomagtól annyiban tér el, hogy az Unused sormintából elvettünk egy bájtot és értelmet adtunk neki.

TYPE: 12

CODE:

4.17. TÁBLÁZAT

CODE	Jelentés
0	A hiba oka a POINTER bájtban
1	Hiányzik egy kötelező érték
2	Elbaltázott hossz

POINTER: Egy szám, mely azt mutatja, hogy az IP Header-en belül melyik bájton kezdődik a hibás paraméter.

IP HEADER AND FIRST 8 BYTES OF DATAGRAM: Hogy ne kelljen sokat keresgélni, hol van a hibás IP Header.

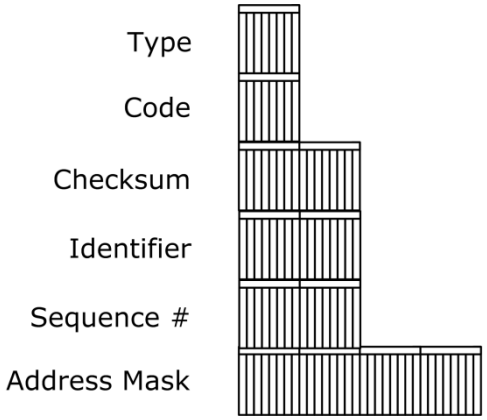
[45] Kivéve CRC error. Akkor ugyanis visszajelzés nélkül dobják el a hostok a csomagokat.

4.1.3.8 ICMP Address Mask Request & Reply

Arról már beszéltünk, mi van akkor, ha egy szenilis host elfelejti a Default Gateway értékét - illetve egy szemfüles router ajánl neki egy rövidebb utat az erdőn keresztül.

Jelen ICMP üzenetet arra használjuk, ha a host a subnet mask értékét felejti el.

A mechanizmus úgy néz ki, hogy a megzavarodott host küld egy ICMP Address Mask Request kérést: vagy egy direktet egy kipécézett routernek, vagy egy broadcastot az alhálózatnak. Ha valaki fogja az adást és ismeri az alhálózatra vonatkozó helyes subnet mask értéket, akkor visszaküldi azt egy ICMP Address Mask Reply csomagban. Ha nem jön semmilyen visszajelzés, akkor a host kínjában a classful kategóriák alapján választ magának subnet mask értéket. (*4.5. táblázat*)



4.36. ábra ICMP Address Mask Request & Reply

TYPE: Request: 17, Reply: 18

CODE: Fix nulla.

IDENTIFIER, SEQUENCE NUMBER: A pinghez hasonló azonosító számok, az összetartozó csomagokat jelölik.

ADDRESS MASK: A Request csomagban egy nulla, a Reply csomagban pedig a helyes subnet mask érték utazik.

Állítgatás:

```
netsh interface ipv4 set global addressmaskreply=enabled v. disabled
```

Alapértelmezésben Windows Server 2008 / Vista esetében tiltott.

4.1.4 IGMP, AZAZ EGY KIS MULTICAST

RFC 1112, 2236, 3376

Egzotikus területre megyünk. Biztos vagyok benne, hogy mindenki hallott már a multicast-ról, de abban is biztos vagyok, hogy nem sokan használjuk.
Pedig nem tűnik rossz dolognak.

Egy gyors emlékeztető. Milyen cast-ok vannak?

- UNICAST : A forgalom két hálózati pont között történik, a hostok közvetlenül egymást címezik.
- BROADCAST : Egy host ad, mindenki más fogja az adást.
- MULTICAST : Egy host ad, több host fogad - de nem mindegyik.

Még mindig az ismétlésnél maradva: mit is jelent az, hogy fogad? A Network Interface Layer fejezetben volt szó arról, hogy a frame-ek akár kisvasúti kocsin, akár a drótba beleordítva eljutnak minden node-hoz a linken. Mindegyik hálózati kártya megvizsgálja, hogy érintett-e a csomaggal kapcsolatban.

- Ha neki szól, akkor a konkrét hoston felküldi egy réteggel feljebb. -> UNICAST.
- Ha mindenkinek szól, akkor mindegyik host felküldi egy réteggel feljebb. -> BROADCAST.
- Ha sok hostnak - de nem mindegyiknek - szól, akkor a konkrét host megnézi, hogy ő közte van-e a kiválasztottaknak. Ha igen, akkor felküldi egy réteggel feljebb. -> MULTICAST.

Gondoljuk végig, mi lehet a multicast nagy előnye? El akarok küldeni egy nagy méretű fájlt több hostnak. Unicast módon a nagy fájl többször is átmenne a dróton. A multicastnál a feladó csak egyszer küldi el - és mindenki, aki érintett benne, felkapkodja a csomagokat. Tipikusan ilyen szituáció lehet egy videó stream, vagy egy csoportos telepítés.

Oké, most már nagyjából tudjuk, mi az a multicast. De mi az az IGMP?
Internet Group Management Protocol.

Mit tudunk a broadcastról? Sok mindent, többek között azt is, hogy a routerek nagyon szigorúan blokkolják. Gondolj bele, ha nem tennék, akkor az egész világ értesülne arról, hogy pl. DHCP szervert keres a számítógéped.

Ezzel szemben a multicast átmegy a routeren.

A trükk az, hogy amikor egy host kap egy multicast címet, akkor odaszól a routerének, hogy 'te, figyi, ezen a subneten van valaki, aki az alábbi multicast címen figyel'. Mi több, a routerek egymással is kommunikálnak, megbeszélve, hogy melyik router mögött milyen multicast címek tartózkodnak.
Ez a multicast témájú nagy fecsegés hostok és routerek, illetve routerek és routerek között, ez az IGMP.

De mielőtt beleásnánk magunkat a routereket érintő kommunikációba, foglalkozzunk egy kicsit magával a multicast-tal.

Amikor megemlítettem a classful IP osztályokat, akkor szó volt egy ún. multicast tartományról is. Ez a 224.0.0.0 - 239.255.255.255 címtartomány. Nem mondtam, de a CIDR erre a tartományra már nem vonatkozik: itt egy kicsit mások a szabályok és ezek nem is változtak az osztályok eltörlésével. (A subnet mask-ot például elfelejtheted.)

Ebben a címtartományban vannak úgynevezett well-known (lefoglalt) címek és vannak egyedi címek. Nézzünk néhányat az első kategóriából:

4.18. TÁBLÁZAT

IP cím	Jelentése
224.0.0.1	Az összes multicast képes host a hálózaton
224.0.0.2	Az összes multicast képes router a hálózaton
224.0.0.4	Az összes DVRMP[46] router a hálózaton
224.0.0.5	Az összes MOSPF[47] router a hálózaton
224.0.0.6	Az összes PIM[48] router a hálózaton
224.0.0.0-224.0.0.255	Lokális címek, nem mennek át a routeren
239.0.0.0-239.255.255.255	Adminisztratív célokra fenntartott tartomány (administrative scoping[49])

Multicast címek:
http://en.wikipedia.org/wiki/Multicast_address

Az egyedi cím definíciója még egyszerűbb: minden lehet egyedi cím, amely nem lefoglalt.

Mit értünk multicast group néven? Minden olyan hostot, mely egy konkrét multicast címen figyel. Például az 'összes multicast képes host a hálózaton' az egy multicast group, mert a 224.0.0.1 címen mindegyik host figyel.

És akkor az mi, hogy multicast képes host? Most akkor minden gép ismeri a multicast-ot, vagy csak a kiválaszottak? És vajon mindegyik ugyanúgy?

[46] Distance Vector Multicast Routing Protocol
[47] Multicast Extension to Open Shortest Path First routing protocol
[48] Protocol Independent Multicast routing protocol
[49] RFC 2365

Route protokollok:
http://www.cisco.com/en/US/tech/tk828/tech_brief09186a00800a4415.html
http://en.wikipedia.org/wiki/Distance_Vector_Multicast_Routing_Protocol
http://www.javvin.com/protocolMOSPF.html
http://www.dataconnection.com/multicast/pimprotocol.htm

Az RFC 1112 szerint az ismerésnek több szintje van:

- Level0: Se küldés, se fogadás szintjén nem ismeri a host a multicaast-ot.
- Level1: Tud multicast csomagot küldeni.
- Level2: Tud multicast csomagot küldeni és fogadni is.

A Windows Server 2008 / Vista alatt ezeket a szinteket netsh-val (naná!) tudjuk állítgatni:

```
netsh interface ipv4 set global mldlevel=none v. sendonly v. all.
```

Na, még egy kis alapozás. Hogyan is kell ezt az egészet elképzelnünk? Akkor az Ethernet kártyás hostnak van egy becsületes IP címe, mondjuk a 192.168.100.2 - aztán van mellette egy multicast címe is?
Bizony. Amennyiben például engedélyezzük rajta a multicast-ot, akkor rögtön kap egy 224.0.0.1 címet[50]. És ehhez adhatjuk hozzá az egyedi címeinket, attól függően, milyen csoportokat szeretnénk kialakítani.

Igen, igen, látom a zavarodat. Eddig arról volt szó, hogy Ethernet, tehát NIL szinten MAC address alapján történik az azonosítás. Milyen MAC address alapján fogják megtalálni a multicast csomagok a gépünket? Neadjisten át fog menni az ARP broadcast a routereken?
Nem, ettől nem kell félni.

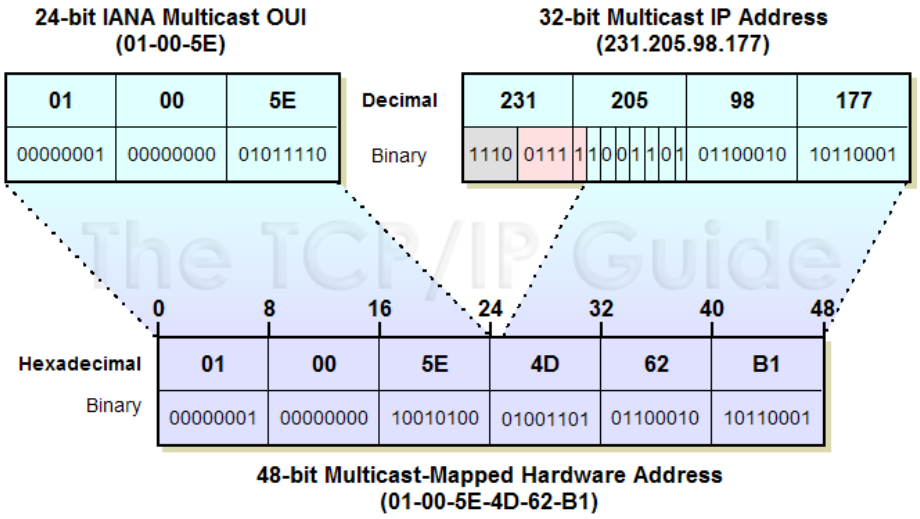
Nem lesz ARP broadcast[51].

A multicast címekhez automatikusan rendelődik MAC address a cím alapján, egy viszonylag egyszerű algoritmus segítségével:
A 48 bites MAC address felső 25 bitje egy fix érték, méghozzá a következő:
00000001 00000000 01011110 0.
Az alsó 23 bitje pedig az IP cím jobb szélső 23 bitje.

[50] Ne keresd, az ipconfig -all nem mutatja.
[51] Lassan mondom, hogy Orbán Viktor is megértse.

Konkrét példa:

Létrehozok egy egyedi multicast group address-t, legyen ez a 224.61.0.4. Hogyan fog kinézni az ehhez a címhez rendelt MAC address?
Az első fele egyszerű: 00000001 00000000 01011110 0.
A második feléhez kell először az IP cím:11100000 00111101 00000000 00000100.
Ebből elvesszük a jobb szélső 23 bitet, majd hozzácsapjuk az első félhez:
00000001 00000000 01011110 00111101 00000000 00000100
Innentől csak át kell váltani hexadecimálisba: 01-00-5E-3D-00-04
Értem?[52]



4.37. ÁBRA HOGYAN KELETKEZIK A MULTICAST MAC ADDRESS (FORRÁS: WWW.TCPGUIDE.COM)

Ha azt mondod, hogy még nem minden világos, akkor azt mondom, hogy bizony igazad van. Hány MAC címe lehet egy hálózati kártyának? Lehet-e egyszerre valódi, beégetett MAC címe és lehet-e mellette multicast MAC címe? Ha lehet, akkor mennyi? És hogyan jelenik majd ez meg?

Itt most vissza kell nyúlnunk az előző nagy fejezethez, azon belül is egészen eddig az ábráig: *3.6. ábra Destination MAC Address*. Hogy ne lapozz annyit, idemásolom a vonatkozó részt:

> A MAC Address alapvetően egy hatbájtos érték, ahol az első három bájt a hardver gyártójára utal, a második három pedig a konkrét eszköz azonosítója.
> Az első bájt legalsó két bitje speciális szerepkörrel bír, mind a feladó, mind a címzett címében.

[52] A fenti MAC Address képzés csak Ethernet hálózatokra vonatkozik. Token Ring esetén például az RFC1469 írja le a módszert.

DESTINATION ADDRESS, I/G BIT (INDIVIDUAL/GROUP)
Azt határozza meg, hogy a megcélzott cím egyedi (unicast, azaz individual), vagy csoportos (multicast, azaz group). Unicast esetben a bit értéke 0, multicast esetben 1. A broadcast (FFFFF) ebben az esetben a multicast alesete.

DESTINATION ADDRESS, U/L BIT (UNIVERSAL/LOCALLY ADMINISTERED)
Azt határozza meg, hogy a megcélzott cím eredeti, beégetett (universal) vagy lokálisan felülírt (locally administered). Alaphelyzetben egy hálózati eszköznek a világon egyedi MAC címe van - bizonyos esetekben viszont felülírhatjuk ezeket az értékeket.

Ebből a bizonyos I/G bitből fogja tudni a hálózati kártya drivere, hogy éppen multicast MAC címről van-e szó, vagy valódiról. Amennyiben az előző, akkor tudni fogja, hogy nem a beégetett MAC címmel kell összehasonlítania, hanem a kártyához rendelt multicast MAC címekkel. Majd amikor felkapta a csomagot, a MAC-IP összerendelésből fogja tudni, hogy a kártya melyik IP címére továbbítsa azt - innentől pedig már jöhet az UDP, utána pedig az alkalmazás.

Jó. Akkor már csak egy kérdés van hátra: hol állítom be a kliensen a multicast IP címet?
Öööö...
Bő félnapos turkálás után arra jutottam, hogy ilyen általános beállítóablak nincs. Ha van multicast-képes alkalmazásod, akkor abban vagy van multicast IP beállítóablak (pl. Message Queue) vagy az alkalmazás maga választ ki egy multicast címet egy bedrótozott multicast tartományból - és ilyenkor elég csak engedélyezned.

RFC 2730

Ezenkívül szóbajöhet még a bolondsipka is: Multicast Address Dynamic Client Allocation Protocol, azaz MADCAP. Ez teljesen analóg a DHCP-vel, olyannyira, hogy a Windows 2000 óta része a DHCP szolgáltatásnak is. Természetesen a MADCAP sem osztogat boldog-boldogtalannak multicast címeket, kell hozzá egy kliens is, aki kér.

Multicast how-to:
http://tldp.org/HOWTO/Multicast-HOWTO-2.html

Internet Protocol IP Multicast Technology:
http://www.cisco.com/en/US/docs/internetworking/technology/handbook/IP-Multi.html

TCP/IP Address Resolution for Multicast IP Addresses:
http://www.tcpipguide.com/free/t_TCPIPAddressResolutionForIPMulticastAddresses.htm

Introduction to Multicast
http://www.firewall.cx/multicast-intro.php

Most, hogy már nagyjából ismerjük az építőkockákat, nézzük végig, mi is történik egy multicast kommunikáció során.

4.1.4.1 EGY SZERVER KÜLDENI AKAR

Itt a nyilam, mibe lőjem?

Nyilván először kell egy csoportcím, ahová el akarom küldeni a csomagokat. Ez lehet egy speciális célra lefoglalt (ez az egyszerűbb, ezt legalább helyből tudjuk) és lehet egy teljesen egyedi - mely esetben a küldő alkalmazásnak kell valahonnan ismernie a csoportcímet.

Mondjuk megvan a megcélzott multicast IP cím. Küldünk ARP broadcast-ot? Ádehogy. Hiszen nemrég írtam, hogy az IP címből egyértelműen meg tudjuk határozni a címzett MAC Address-ét. És nem csak mi, feladók, hanem a címzett is magának. Azaz biztosak lehetünk benne, hogy ha be van állítva nála az a bizonyos multicast IP Address, akkor fel fogja kapni a drótból a csomagot.

Mielőtt útjára indítanánk a csomagot, beszéljünk még a TTL-ről. Mekkorára válasszuk? Nyilván ha megcélzott IP cím a 224.0.0.0-224.0.0.255 között van, akkor a TTL az 1 lesz. Mert ez szigorúan helyi cím kell legyen, azaz már az első router le kell blokkolja. Ha viszont 224.0.1.0 feletti címről van szó, akkor valami magasabb TTL-t kell kiokoskodnunk. (Létezik rá egy ökölszabályszerű táblázat.)

Jó. Legyen a TTL mondjuk 10. Ekkor átmehetünk 9 routeren. De hogyan találjuk meg ezt a bonyolult útvonalat?

4.1.4.2 EGY KLIENS FOGADNI AKAR

Nem, nem lóversenyen.

Akár arról van szó, hogy a kliensnek van valamilyen multicast IP címe, melyen figyel a well-known csomagokra... akár arról, hogy egy alkalmazással egyeztettek egy egyedi IP címet és azon figyel - nos, maga az IP cím megléte még kevés abban az esetben, ha a feladó és a címzett között van egy router. Azaz a multicast IP cím nagyobb, mint 224.0.1.0.

Ebben az esetben a kliens nem üldögélhet tétlenül a vezetéken, hanem akcióba kell lépnie. Fogja, és értesíti a routerét, hogy van ám a hálózatban egy olyan host, aki ezen a bizonyos multicast IP címen figyel. (IGMP Host Membership Report.) A router pedig fogja és tudomásul veszi.
Mi történik, ha bekapcsolódik egy másik host is ebbe a multicast kommunikációba az illető subnet-en - azaz ő is megkapja ugyanazt a multicast IP címet? Ő is szól a routernek. Mit csinál a router? Nagy ívben letojja. A routernek csak arra az egy információra van szüksége, hogy _*minimum egy*_ host figyel-e a címre.

Ugye, pont ezért multicast.

4.1.4.3 A ROUTER, AKI ÖSSZEHOZZA A FELEKET

Feltéve, hogy a router IGMP kompatibilis.

Hogyan is néz ki egy ilyen router feladatlistája?

- Először is, nyilván meredten figyel az IGMP Host Membership Report csomagokra.
- Másodszor ő is intenzíven érdeklődik, mi is a helyzet a subnet-en: azaz rendszeresen IGMP HOST Membership Query kéréseket küld ki.
- Akár így, akár úgy, beesnek az adatok. A router ezeket szépen csoportosítgatja, pátyolgatja, táblázatban tartja.
- De ezzel a kinccsel nem bánik fukarul: ha az alhálózaton van másik IGMP-képes router is, akkor vele is megosztja az információt. (Erre szolgálnak a korábban már említett DVRMP, MOSPF és PIM routing protokollok.)
- És persze a lényeg: figyel a host helyett is. Azaz ha a router a külső lábán észleli, hogy valaki multicast címre küldött egy csomagot, méghozzá olyanra, mely csoportos címhez tartozik a belső lábán lévő subneten is host, akkor átemeli a multicast forgalmat a belső lábára is.[53]
- Mindezt persze kaszkádolva is tudja - köszönhetően a routerek egymás közötti kommunikációjának. Azaz ha van a router, alatta egy downstream router, az alatt meg egy másik downstream router, a multicast csomag képes lekeveregni a legalsó szintre is - feltéve, hogy a láncban az összes router IGMP-képes, a csomag TTL-je pedig enged annyi hopot.

IGMP linkek:
http://en.wikipedia.org/wiki/Internet_Group_Management_Protocol
http://www.networksorcery.com/enp/protocol/igmp.htm
http://www.javvin.com/protocolIGMP.html

[53] Ez azért nem ennyire egyszerű. Mit is értünk azalatt, hogy egy router figyel a multicast forgalomra? Ugye nem lehet minden létező multicast címhez külön MAC Address-e.
Akkor felránt megvizsgálni olyan csomagokat is, melyeket nem is neki küldtek?
Pontosan. Ezt nevezik promiszkusz üzemmódnak. Két verziója is van:

- Teljes: a host minden csomagot felránt.
- Multicast: a host csak a multicast csomagokat - ugye az Ethernet MAC Address esetén az utolsó bit magas az első oktetből - rántja fel, de abból mindegyiket.

És igen, az nyert, aki arra tippelt, hogy mindkettő borzasztó erőforrásigényes üzemmód. Ráadásul ütik is egymást.

4.1.4.4 IGMP v1

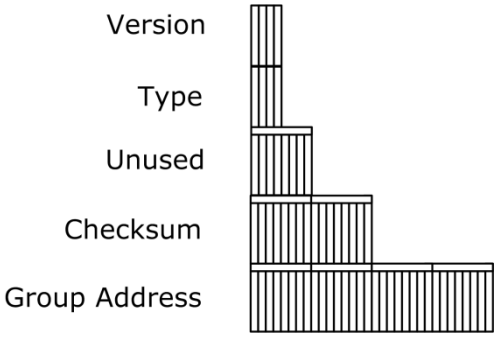
RFC 1112

Röviden összefoglalva az eddigieket:

- A hostok, amennyiben ismerik a multicast üzemmódot, és van multicast címük, Host Membership Report üzeneteket küldözgetnek rendszeresen a hálózaton. Az üzenetek feladója a saját, nem multicast IP címük, címzettje az aktuális multicast cím, a TTL pedig 1.
- A routerek, feltéve, hogy IGMP-képesek, rendszeresen küldözgetnek Host Membership Query üzeneteket a LAN-on. (Feladó a router nem multicast IP címe, címzett a 224.0.0.1, a TTL szintén 1.) Ezekre a hostok Host Membership Report üzenetekkel válaszolnak.

Gondoljuk végig, hogyan is van ez? A router kiküld egy HMQ üzenetet, majd minden host eszeveszett módon elküld válaszként egy HMR üzenetet? Ekkora vihar, jórészt feleslegesen? Hiszen a routert istenigazából nem az érdekli, hogy konkrétan mely hostok tartoznak egy-egy multicast csoportba, neki bőven elég az is, hogy van-e legalább egy tagja egy csoportnak?
Ebből kifolyólag nem minden host válaszol egyszerre, hanem véletlenszerű ideig fontolgatják a megszólalást, majd válaszol... az egyik. Mivel a HMR esetén a címzett a multicast csoport, így a csoportbéli tovább fontolgatók már értesülnek róla, hogy valaki válaszolt, ergo nekik már nem kell.
Amennyiben méltányolandó időtartamon belül nem jön válasz egy konkrét multicast csoportból, akkor a router halk sóhajtással törli azt a táblázatából és értesíti a haver routereket.

Akkor nézzük most már bit szinten is a konkrétumokat.



4.38. ÁBRA IGMPV1 CSOMAG FELÉPÍTÉSE

VERSION: Az IGMP verziószáma, értelemszerűen jelen esetben 1.

TYPE: Az üzenet tipusa.

- 1: Host Membership Query
- 2. Host Membership Report

UNUSED: Vajon?

CHECKSUM: A jó öreg CRC.

GROUP ADDRESS: HMR esetén a csoporttagságot jelentő multicast cím. HMQ esetén nem használt, az értéke 0.0.0.0.

IGMPv1:
http://technet.microsoft.com/en-us/library/cc957911.aspx

4.1.4.5 IGMP v2

Az IGMPv1 egyszerű volt, valamennyire hatékony is - de akadt rajta tökéletesíteni való.

RFC 2236

Az IGMPv2-ben a következő újdonságok debütáltak:

- 'Elhagytam A ~~Várost~~ Csoportot' üzenet, azaz Leave Group Message.
- Csoportspecifikus lekérdezések, azaz Group Specific Query.
- Kérdezőbiztos (querier) megválasztása.

4.1.4.5.1 ELHAGYTAM A CSOPORTOT

Az IGMPv1-ben ha kiürült egy multicast csoport, a router egészen addig nem értesült róla, amíg körbe nem küldött egy HMQ lekérdezést és arra végül nem kapott HMR választ az érintett csoportból. Azaz egészen addig nyomta át erre a hálózatra is a multicast adást. Most mondhatnánk, hogy 'jajjaj, szegény router, na bumm, egy kicsit feleslegesen dolgozott' - de ne felejtsük el, a multicast kommunikációnak éppen az a lényege, hogy nagy sávszélességet igénylő adások mennek rajta, célzottan. És akkor még nem is beszéltünk arról, mennyi idő, amíg egy alhálózaton a 'csoport kiürülése' információ végighullámzik a többi IGMP-képes routeren.

Az IGMPv2-ben eleinte minden ugyanúgy megy. A router küld egy HMQ lekérdezést, amelyre minden csoportból válaszol az egyik host. Emlékszünk, a hostok véletlenszerű merengési idővel reagálnak, és az első válaszoló nyer. Nos, az IGMPv2-ben ennek az első válaszolónak annyira megnő az önbizalma, hogy azt feltételezi, ő az egyetlen, ergo az utolsó csoporttag a csoportban. Emiatt ha valamilyen oknál fogva elhagyja a csoportot, akkor küld egy 'legyetek jók, ha tudtok' üzenetet, azaz egy Leave Group Message csomagot. A címzett ilyenkor a 224.0.0.2, azaz az all router csoport, a feladó a host nem multicast címe, a Group Address pedig az elhagyott multicast csoport címe. Persze az IGMPv2 routerek sem most jöttek le a falvédőről, ellenőrzésként kiküldenek egy csoportspecifikus query-t, a később tárgyalandó Group Specific Query csomagot. Ha erre valaki válaszol, akkor a routerek nem csinálnak semmit. Ha csak néma csend követi a kérdést, akkor tényleg a kilépő host volt az utolsó, beindul a multicast csoport törlési procedúrája.

4.1.4.5.2 Csoportspecifikus lekérdezés

A Group Specific Query (GSQ) üzenet teljesen analóg az IGMPv1 Host Message Query üzenetével. Az egyetlen különbség az, hogy amíg az általános esetben használt HMQ üzenet címzettje a 224.0.0.1 (all host), illetve a csomagban lévő GROUP ADDRESS értéke 0.0.0.0, addig a GSQ üzenetben mindkét helyen a lekérdezendő csoport multicast címe található. Ebből következően nem minden multicast host kapja meg, hanem csak az illető csoporttagok - ergo csak ők reagálnak egy HMR üzenettel.

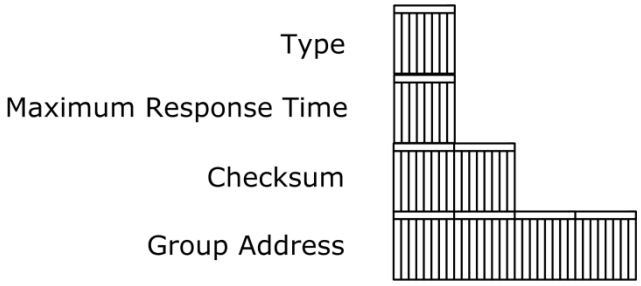
4.1.4.5.3 A kérdezőbiztos

Akkor jön be a képbe, ha több IGMP-képes router is van egy alhálózaton. Nyilván annak nincs semmi értelme, hogy mindkettő HMQ üzenetekkel árassza el a hálózatot - éppen elég, ha az egyik kérdezősködik, aztán a változásokat már valamelyik routing protokoll segítségével átküldni a másiknak. Vagy többinek.

Mi a kiválasztás módszere? Melyik lesz az erősebb kutya?

Amelyiknek nagyobb a ... IP címe.

Nézzük, hogyan változott meg a csomagszerkezet.



4.39. ÁBRA IGMP V2 CSOMAG FELÉPÍTÉSE

Örvendetes változásokat láthatunk. Például a korábbi két félbájtot összevonták egybe, illetve találtak funkciót annak a kihasználatlan bájtnak is.

TYPE:

4.19. TÁBLÁZAT

TYPE	Értelmezése
17 (0x11)	Host Membership Query (ide tartozik a GSQ is)
18 (0x12)	IGMPv1 Host Membership Report
22 (0x16)	IGMPv2 Host Membership Report
23 (0x17)	Leave Group Message

MAXIMUM RESPONSE TIME: Lekérdezések esetén (HMQ/GSQ) maximum ennyi ideig vár a router a válaszra.

GROUP ADDRESS: A hagyományos HMQ esetén 0.0.0.0, minden más esetben az érintett multicast cím.

IGMPv2:
http://technet.microsoft.com/en-us/library/cc957916.aspx

4.1.4.6 IGMP v3

RFC 3376

Nehéz ügy. Az IGMPv2-vel már tökéletesre faragtuk a multicast forgalom kezelését - mi jöhet még? Mit tudunk még javítani rajta?

A hagyományos multicast kommunikáción már semmit. Csakhogy ma már a multicast sem a régi.

Mi is a multicast? Van egy forrás és az úgy juttat el egy adást néhány kiválasztott gépre, hogy a forgalom csak egyszer megy át a dróton.

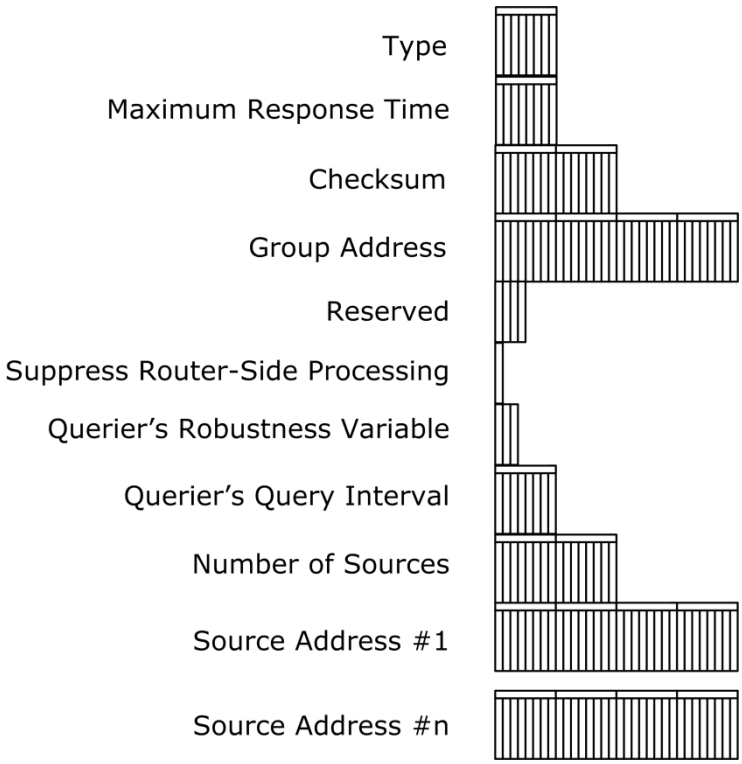
Tippelj: mennyi esélye van annak, hogy egy általad internetre küldött adás eléri mindazokat multicast hostokat, melyek különböző kontinenseken vannak rácsatlakozva a netre? Ugye, nem nehéz: nulla. Ma az interneten elhanyagolhatóan kevés a multicast routerek száma[54].
De lépjünk túl ezen a fázison. Legyen egy ideális enterprise méretű hálózatod, ahol minden router egyben IGMP-képes is. Gyönyörű. Mehet oda-vissza mindenhol a multicast. Elkezdünk nyomni egy adást mondjuk San Francisco-ban, aztán aszerint, hogy a távolabbi hálózatokban, illetve az azok mögötti hálózatokban akad-e olyan host, mely ehhez az adáshoz tartozó multicast címmel rendelkezik, a routerek teszik a dolgukat. Így az adás előbb-utóbb eljut Kazincbarcikára is.
De mi van akkor, ha valahol vékony a sávszélesség? Mondjuk Budapest és London között? Ekkor az összes magyar helyszínen borzasztóan fog szaggatni az adás. Logikusan célszerűbb lenne valami hordozható médián már előre eljuttatni az anyagot Budapestre - és még néhány helyi központba a világban - majd amikor indul a műsor, mindegyik résztvevő host maga dönti el, melyik adóra figyel a csoporton belül. Ezt hívják úgy, hogy Multiple Source For Multicast Traffic, az IGMPv3 pedig azért jött létre, hogy ezt a kommunikációt ki tudja szolgálni.

MBONE:
http://www.mbone.net/
http://en.wikipedia.org/wiki/Mbone
http://acs.lbl.gov/OldMisc/mbone/

[54] Viszont létezik az interneten egy MBONE, azaz Multicast Backbone. Ezt leginkább dedikált szervezetek (IETF, NASA) használják. De tulajdonképpen itt is arról van szó, hogy multicast szigeteket kötnek össze egy gerincvonallal, úgy, hogy maga a gerinc a normális IP forgalomba van belecsatornázva. (IP-in-IP tunnel). Részletesebben lásd a link szekciót.

4.1.4.6.1 IGMPv3 Host Membership Query

Az IGMPv2-ben már csoportspecifikussá tettük a HMQ üzenetet - most emellett forrásspecifikus is lesz. Azaz a router már nem csak egy csoportot céloz meg a query-vel, hanem konkrét forrásokat is. Egész konkrétan a query tartalmazni fog egy forráslistát, és vagy csak a listán szereplő forrásokra csimpaszkodó hostok kapják meg (include list) vagy pont ellenkezőleg, a minden más forrásra akaszkodott hostok (exclude list).



4.40. ábra IGMPv3 Host Membership Query

Egy kicsit megnőtt a csomag. Az eleje a régi (még a TYPE=11 is), de utána kapott egy csomó mezőt.

Reserved: Sorminta.

Suppress router-side processing: Ez egy trükkös flag. Arról van szó, hogy ha - a lentebb lévő - robusztussági előírás magas, akkor a kérdezőbiztosnak akkor is ki kell küldenie a HMQ csomagokat, ha már kapott választ. Ilyenkor állítja magasra ezt a flag-et, jelezve a többi routernek, hogy habár neki muszáj volt kiküldenie ezt a kérést, de ne foglalkozzanak vele - azaz ne nullázzák le a stopperüket. (Mely futó idő lesz összehasonlítva a QUERY INTERVAL értékével.)

QUERIER'S ROBUSTNESS VARIABLE: A kérdezőbiztos - azaz a feladó router - robusztusságát jelző változó. Egy mérőszám, amely azt mutatja, hogy hány IGMP csomag veszhet el anélkül, hogy beinduljon egy helyreállítási folyamat.

QUERIER'S QUERY INTERVAL: Hány másodperc telhet el két query között.

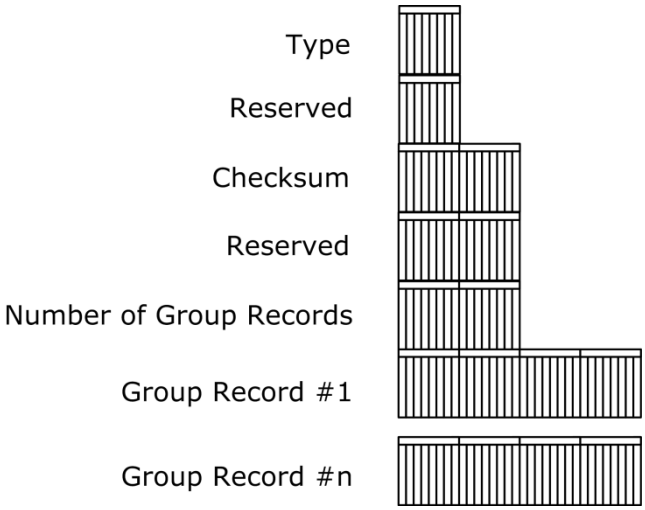
NUMBER OF SOURCES: Mennyi SOURCE ADDRESS is lesz a csomagban... azaz mennyi is az 'n'?

SOURCE ADDRESS X: A multicast forrás címe.

4.1.4.6.2 IGMPv3 Host Membership Report

Nyilván ez sem lett egyszerűbb. A két leglátványosabb változás.

- Le lett foglalva a 224.0.0.22 cím az IGMPv3-képes routerek számára. A hostok csak erre a címre küldenek IGMPv3 tipusú HMR üzenetet.
- A HMR-ben nem egyszerűen a csoporttagság szerepel, hanem úgynevezett Group Record blokkok. Minden blokk tartalmaz egy multicast címet és n darab - a multicast címen sugárzó - source címet.

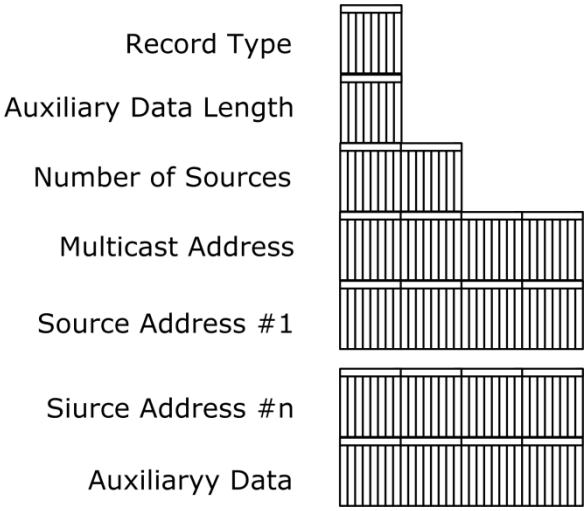


4.41. ábra IGMPv3 Host Membership Report

Type: 0x22.

Number of group records: A blokkok száma.

GROUP RECORD #X: Adatblokk.



4.42. ÁBRA IGMPV3 GROUP RECORD

RECORD TYPE: A lista vajon megengedő (include) vagy lezáró (exclude)?

AUXILIARY DATA LENGTH: Mennyi kisérő adatot rittyentettünk még a blokk végére.

NUMBER OF SOURCES: A források száma.

MULTICAST ADDRESS: A csoport multicast címe, melyhez a host kapcsolódik.

SOURCDE ADDRESS #X: A műsort szolgáló források címei.

AUXILIARY DATA: Azok a bizonyos kisérő adatok.

4.2 IPv6

Jó 10-15 évvel ezelőtt megjósolták, hogy az IP címek el fognak fogyni, az internet pedig összeszakad. Nos, ez nem történt meg, köszönhetően a korábban már említett CIDR-nek és NAT-nak.
Persze ezzel csak időt nyertünk, hiszen mindkettő csak foltozása egy kinőtt gúnyának.

- Azért a NAT csak erőforrásigényes. Gondoljunk egy kicsit vissza a folyamatra: láthatjuk, egy kapcsolathoz a routernek el kellett raktároznia a port mapping értékeket. Mi van, ha több ezer kapcsolat pörög át a routeren percenként? Füst. Meg csökkentett felhasználói élmény.
- Mi van akkor, ha erőforrásokat akarunk publikálni az internet felé? Mondjuk, két darab webszervert? A routernek csak egy darab 80-as portja lesz. A másik webszervernek már bele kell törődnie a nem szabályos portba. Örülni fognak neki a reménybeli látogatók. (Feltéve, hogy a náluk lévő tűzfalon egyáltalán engedélyezve lesz mondjuk a helyettesítőnek kitalált 81-es port.)
- A NAT nem igazán tud megbírkózni azzal a szituációval, amikor maga az alkalmazás szintű protokoll rakosgat be IP cím, illetve port értékeket a szállítandó csomagba - hiszen nonszensz lenne elvárni egy natoló eszköztől, hogy ismerje az összes protokollt.
- Titkosítás. Ha a router belepiszkál a csomagba, akkor hiába írtam alá odabent digitálisan, az egész megy a levesbe. Arról nem is beszélve, hogy egy rendesen titkosított csomagot nem is tud a router értelmezni. (Mondjuk a NAT Traversal - IPSEC NAT-T - segítségével le lehet kezelni a problémát, viszont ez erőforrásba kerül.)
- Próbáljuk meg elképzelni a multicast forgalmat egy natolt hálózat felé.

És ezek még csak a NAT hátrányai. Lehetne sorolni az IPv4 kellemetlen tulajdonságait is:

- Szűk névtér.
- A routolási szintek nehezen strukturálhatók.
- Az autokonfiguráció hiánya.
- A QOS[55] csak azzal a bizonyos módosított TOS mezővel oldható meg, de ezt nem minden hálózati elem ismeri. (Pontosabban, vannak olyan elemek, melyek a TOS régebbi értelmezését ismerik csak.)
- A mobil kliensek támogatásának nehézkessége. (Pl. ha a mobil kliens menetközben cellát vált, a kapcsolat nem szakadhat meg.) Ezt IPv4 alól lekezelni nagyon körülményes.

[55] Quality Of Services: sávszélesség-szabályozás.

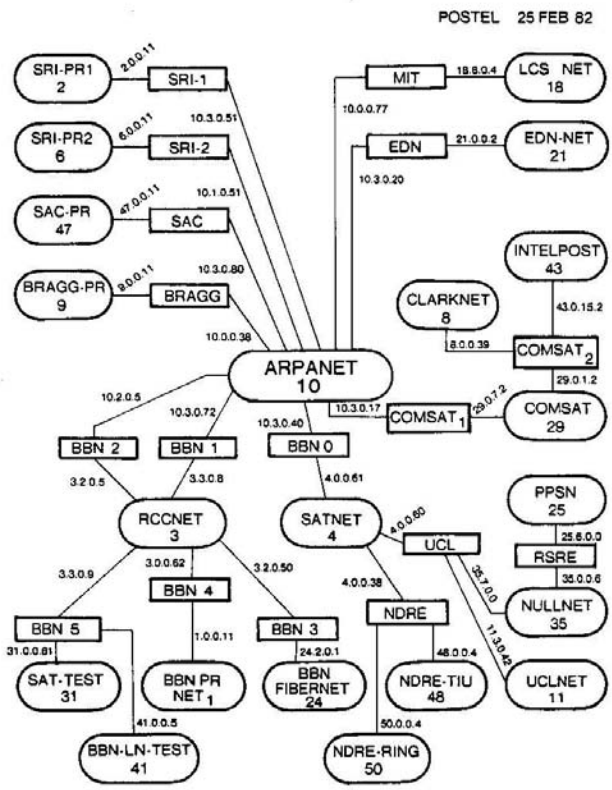
Nyilván itt jön képbe az új internet, az IP v...6. (Az öt... az menetközben elvérzett.[56])

RFC1719

Valahogy így történt:

- 1990. Az IETF megállapítja, hogy baj van. Belevágnak a CIDR-be.
- 1993. Az IETF kezdeményezésére beindul az IPng (new generation) kidolgozása. Még ebben az évben kijön az RFC1719, egyfajta irányelv gyűjtemény.
- 1995. Rengeteg felmérés, ötletelés után elméletben összeáll a kép. Az újszülöttet IPv6-nak nevezik el.

1995. Ugye, döbbenet. Én viszonylag korán lettem függő, de az első emailcímem így is csak 1997-ben keletkezett[57], az első honlapom pedig 1998-ban debütált. Akkortájt kezdtem rácsodálkozni erre az érdekes új világra.
Miközben a háttérben már faragták a még újabbat[58].



4.43. ÁBRA AZ INTERNET 1982 FEBRUÁRJÁBAN

[56]Stream jellegű adatátvitelre tervezték. De senki nem használta. 1970-ben?
[57] lazybear@rocketmail.com
[58] Emlékszem, 1999 környékén egy hibát jelentettem be a cégem internet-szolgáltatójának.
- IPv4-et vagy IPv6-ot használnak? - kérdezett vissza a rendszergazda.
- Izé... - lett hirtelen melegem - mi a különbség?

4.2.1 ALAPOK

Amikor ezt az egész internet dolgot elképzelték, senkinek még csak meg sem fordult a fejében, hogy az a töménytelen mennyiségű kiadható cím, melyet a 4 bájtos IP cím és a 4 bájtos subnet mask segítségével reprezentálni tudunk, valamikor még kevés lesz.
De kevés lett.
Az igények ugyanis nőttek. Nehogy már ne én mondhassam meg a kenyérpirítómnak, hogy mikorra érek haza és mennyire süsse át addigra a kenyeret. A hűtőgépem pedig igenis küldjön figyelmeztető emailt, ha kevés benne a sör. Ez mind-mind IP címet kíván. Méghozzá egyedit, mert egy benatolt kenyérpirító, lássuk be, nem az igazi. (Lehet, hogy a router mögött már a porszívó foglalta le a megfelelő portot.)

Mondtam, hogy nagyobb lesz a névtér. Mennyivel?

IPv4 alatt 4 bájtot használtunk, most 16-ot fogunk. Aki ebből azt a következtetést vonta le, hogy az új névtér négyszerese lesz a réginek, az menjen vissza a jó öreg alma materbe és nyomjon egy tockost a matektanárának, mert rosszul tanították meg számolni. Négy bájton 2^{32} különböző szám ábrázolható. 16 bájton pedig 2^{128}. Ha elosztjuk a két számot egymással, azt kapjuk, hogy az IPv6-ban egész pontosan 2^{96}-szor több lehetőségünk van, mint az IPv4-ben. (Más szóval nem négyszerese, hanem negyedik hatványa.) Ez borzasztóan nagy szám ahhoz, hogy legalább az elkövetkező tíz évben nyugodtan ülhessünk a hátsónkon. (Remélve, hogy Kínában és Indiában úrrá lesznek végre a népszaporulaton[59].)

Hogyan fogjuk leírni ezt az őrült nagy számot?

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.

Félelmetes. Ki a fene fog ekkora számokat megjegyezni?

[59] Rossz vicc volt, bocs.

Nos, természetesen meg lehet - de ahhoz tömörebb ábrázolás kell. Itt jön be a képbe barátunk, a hexadecimális.
Ekkor ezt kapjuk:
C0A8:170C: C0A8:170C: C0A8:170C: C0A8:170C
Nos, valamennyit már összébbment: ez már csak 8 számjegy a 16 helyett - igaz, ezekben már vannak betűk is.

A rossz hír, hogy ezt a próba címet már nem lehet tovább tömöríteni: a jövőben rendszergazdaként nyolc számjegyű címekkel kell fejben zsonglőrködnünk. A jó hír, hogy amennyiben nulla értékű bájtok is vannak a címben, akkor még tömöríthetünk egy kicsit:

- **fe80:0:0:0:fd12:2f1:1234:abcd:**
  Látható, hogy nem kell mindenhol kiírni balról a nullákat. Nem azt írjuk, hogy 0000 vagy 02f1. (Mint ahogy a tizes számrendszerben sem azt írjuk, hogy 192.168.023.012.)
- **fe80::fd12:2f1:1234:abcd:**
  Ez már kacifántosabb rövidítés. Amíg az egymás melletti blokkokban csak nullák vannak, addig az a rész elhagyható. Vegyük észre, hogy ahol kimaradtak a nullák, ott megduplázódtak a kettőspontok. Hogy mennyi maradt ki? Tudjuk, nyolc részletnek kell lenni, látunk ötöt - tehát három blokknyi nullánk van. Vigyázat, egy címen belül csak egy ilyen összevonás lehet. Nincs olyan cím, hogy fe80::abcd::21 - tekintve, hogy ekkor nem tudnánk, hogy mely részen hány üres blokk volt. (Aztán ebből elég vad dolgok is ki tudnak sülni. Hogy mást ne mondjak, a localhost úgy néz ki, hogy ::1.)

Hallom a kérdést, mi hír van a subnet mask-ról? Köszöni szépen, a lényege megmaradt, csak mostantól prefixnek nevezik. Továbbra is azt fogja mutatni, hogy az IP cím hány számjegye azonosítja a hálózatot és hány számjegye magát a hostot.
Egyszerűbb példával illusztrálni: *fe80:0:1234:adf:2::/64*. A '/' jel mögötti szám mutatja meg, hogy ha binárisan írjuk fel a számot, akkor balról hány számjegy azonosítja a hálózatot. A '/64' az ugye jelen esetben azt jelenti, hogy középen van a határvonal, a cím első fele a hálózati azonosító, a második fele pedig a host azonosító, egész konkrétan:

- hálózat : fe80:0:1234:adf
- host : 2:0:0:0, azaz 2::

RFC 2373

Az IPv4-nél megismert címzések (unicast, broadcast, multicast) köre eggyel bővült: belépett mellékük az anycast is. Ez nagyon hasonló a multicast-hoz, ugyanúgy több host is figyelhet egy csoportos címen - de bármelyik is kapta fel magához a csomagot, akkor az már fel lesz kapva - a többieknek nem jut belőle.

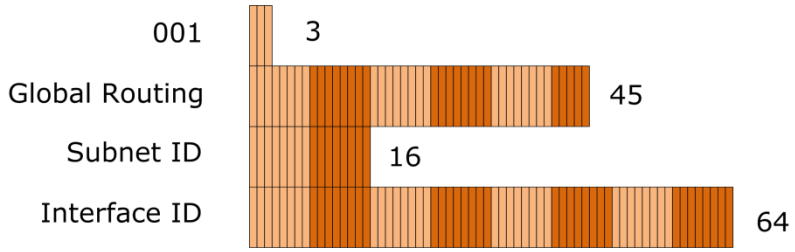
4.2.1.1 UNICAST CÍMEK

A következő unicast címek léteznek:

- UNIQUE-GLOBAL, azaz egyedi globális cím
- LINK-LOCAL cím, azaz csak a linken értelmezett cím
- SITE-LOCAL cím, azaz csak a lokális site-on értelmezett cím
- UNIQUE-LOCAL, azaz lokálisan egyedi cím
- Speciális címek
- Transition, azaz áttérési címek.

4.2.1.1.1 EGYEDI GLOBÁLIS CÍMEK

Az egyedi globális címek, mint a nevük is mutatja, abszolút egyediek. Az egész világegyetemben. A felépítésük nagyjából így néz ki:



4.44. ÁBRA EGYEDI GLOBÁLIS CÍM

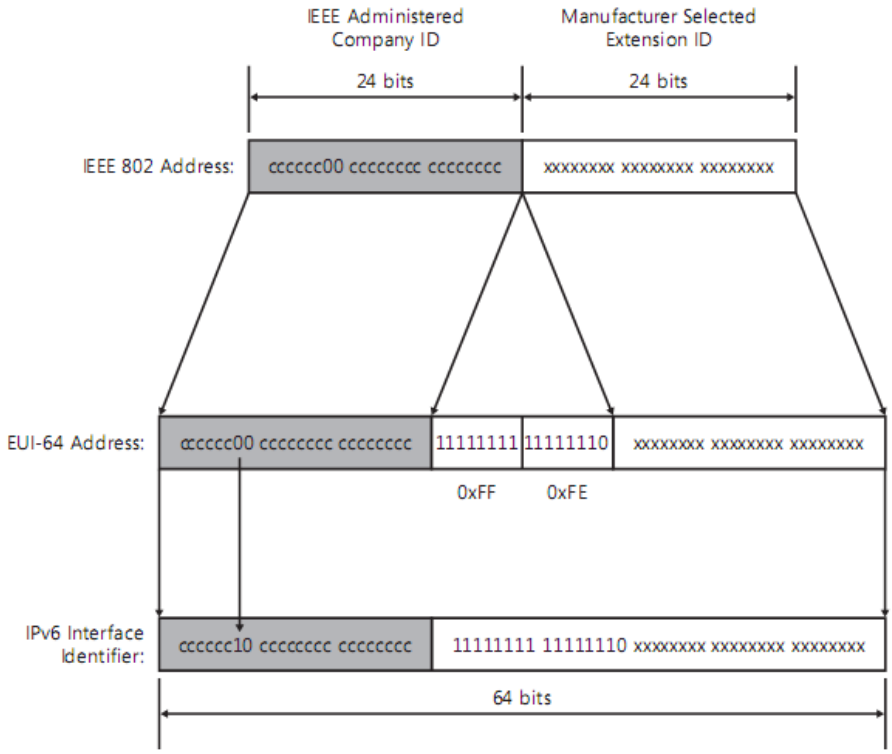
001: Ez a fix része a címnek.

GLOBAL ROUTING PREFIX: Ettől lesz a cím globális. Tulajdonképpen ebben a tartományban fognak nyüzsögni az ISP-k.

SUBNET ID: A globális címen belüli alhálózatok (site). Logikusan 65536 lehet belőlük.

INTERFACE ID: A node egyedi azonosítója. Mivel ez egy olyan elem, mely a legtöbb címzési technikánál visszaköszön, érdemes elmélyedni a generálásában. Létezik egy kódolás, 64-bit Global Identifier (IEEE EUI-64) a pontos neve. Ennek van egy olyan fejezete, mely azzal foglalkozik, hogyan is lehetne egy 48 bites MAC (IEEE 802) addressből 64 bites egyedi azonosítót fabrikálni. Le fogsz esni a székről, ha elárulom, hogyan: a MAC address közepére beszúrnak 16 bitet, méghozzá ezeket: FFFF. Pontosabban, ez az ajánlás, de az IPv6-ban inkább az FFFE értéket szúrják be. Még pontosabban, ez csak az egyik lehetséges interface id generálási mód... de a legnépszerűbb. (Emlékszünk, a MAC address első 24 bitje a gyártó azonosító kódja, a második 24 a kártyáé? Azaz a FFFE érték pont a kettő közé szúródik be.)

A fenti címképzés az elméleti változat. A gyakorlatban beszúrtak a tervezők néhány matyó csujjogást a koreográfiába: a MAC address gyártóspecifikus részében a konverzió során átfordítanak egy bitet. Pusztán a tömörebb számábrázolás kedvéért. A teljes folyamat az ábrán látható.

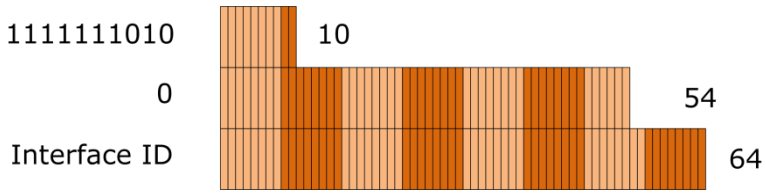


4.45. ÁBRA MÓDOSÍTOTT ÉS MEGVARIÁLT EUI64

4.2.1.1.2 LINK-LOCAL CÍMEK

A link-local címek azon címei egy node-nak, melyek csak az adott linken érvényesek. A router ezeket a címeket nem fogja kiengedni.

Képzésük:



4.46. ÁBRA LINK LOCAL

1111111010: Fix érték.

0: 54 üres bit: Meglehetősen fix érték.

INTERFACE ID: Már ismerjük.

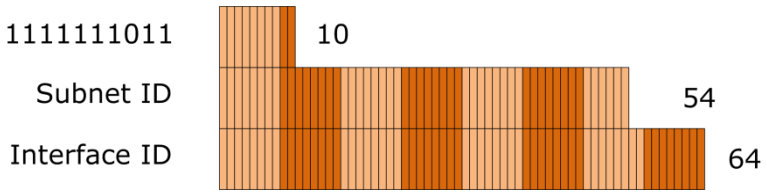
Gondolom, látod te is, hogy ez az egész felírható úgy is, hogy FE80::/64 prefix + node azonosító.

Az INTERFACE ID értékének linkenként kell egyedinek lennie, azaz eltérő linkeken simán kiadható ugyanaz az INTERFACE ID.

4.2.1.1.3 SITE-LOCAL CÍMEK

Amikor ilyen nagy címtartományunk van, simán előfordulhat, hogy ún. köztes alhálózati rendszert iktatunk be a GLOBAL ROUTING PREFIX és az INTERFACE ID közé (SUBNET ID). Ezeket a köztes alhálózatokat hívják site-nak, a site-on belülről kitörni nem képes címeket pedig site-local címeknek.

Így néznek ki:



4.47. ÁBRA SITE LOCAL

A szereplőket már ismerjük. A link local címeknél írtak itt is érvényesek, csak site szinten: az INTERFACE ID-nak site szinten kell egyedinek lenni.

Itt mondjuk elgondolkodtam. Ha megnézed a globális címeknél a Subnet ID mérete 16 bit, itt meg 54. Hát ezt meg hogyan?

Nem lehet más a megoldás, mint az, hogy a két Subnet ID név mögött eltérő tartalmak léteznek.

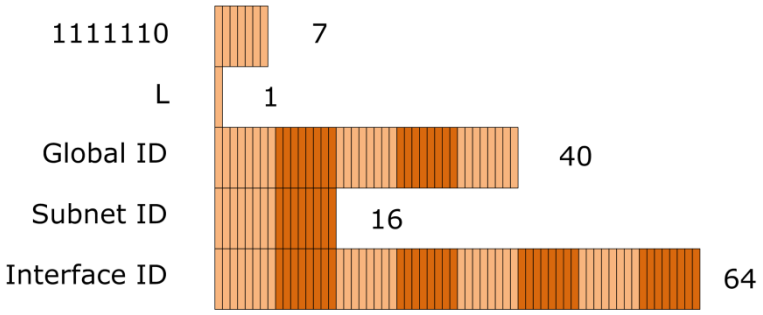
- Globális címnél az ISP-nk subnetel nekünk egy köztes szintet. 2^{16}, azaz 65536 köztes alhálózatot hozhat létre.
- A Site-local címnél viszont mi magunk - az ISP-nk mögött - hozhatunk létre egy szigorúan lokálisan értelmezett köztes subnet rendszert.

4.2.1.1.4 UNIQUE LOCAL ADDRESS:

RFC 4193

A helyzet az, hogy a site-local címek miatt nem lehetünk 100 százalékig biztosak abban, hogy a link-local címeink abszolút egyediek a linken.

RFC 3879 - Developer Pain



4.48. ÁBRA UNIQUE LOCAL CÍMEK

Emiatt vezették be a unique-local címeket. Olyan, világméretben 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 világméretben egyedi.

RFC 4193

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.

4.2.1.1.5 Speciális címek

Például a localhost: ::1/128

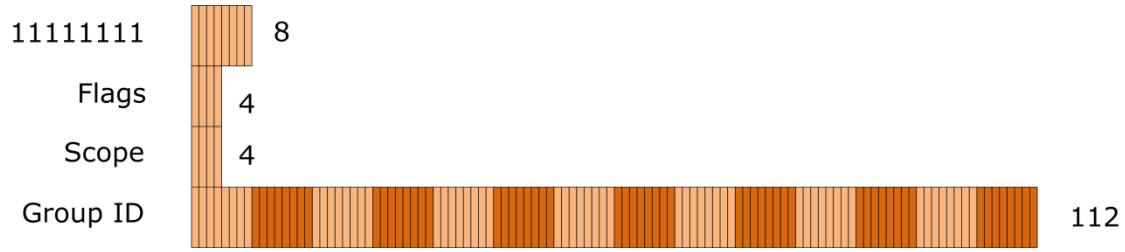
4.2.1.1.6 Transition címek

Átállítani az internetet az IPv6-ra... igen nagy falat lesz. Nem is fog menni egyik napról a másikra. A békés átmenet érdekében szükségünk lesz egy csomó kétéltű címre. Itt csak felsorolom ezeket, a részletezésükre később kerül sor.

- IPv4-compatible address
- IPv4-mapped address
- 6to4 address
- ISATAP address
- Teredo address

4.2.1.2 MULTICAST CÍMEK

Általában így néznek ki:



4.49. ÁBRA MULTICAST

11111111: Fix érték.

FLAGS: A négy bitből a három alsó - azaz jobb szélső - bit flagként viselkedik. Menjünk jobbról:

Transient, azaz T-flag.

RFC 2373

Ha alacsony, akkor ez egy well-known cím. Ha magas, akkor egy kósza, akárki által kiosztott címről beszélünk.

Prefix, azaz P-flag.

RFC 3306

Ha magas, akkor a multicast cím egy unicast global prefix-hez kapcsolódik. Ha alacsony, akkor nem.

Rendezvous Point Address, azaz R-flag.

RFC 3956

Ha magas, akkor a multicast cím tartalmaz ún. randevúpont (RP) címet. Ha alacsony, akkor nem.

SCOPE: A multicast csoport érvényességi köre.

4.20. TÁBLÁZAT

A szkóp értéke	Értelme
0	Lefoglalt, nem használható
1	Interface-local scope
2	Link-local scope
3	Lefoglalt, nem használható
4	Admin-local scope
5	Site-local scope
8	Organization-local scope
E	Global scope
F	Lefoglalt, nem használható

GROUP ID: A multicast csoport jellemző címe.

És akkor nézzük az előrecsomagolt, beépített multicast címeket. Logikusan mindegyik az FF0x blokkal fog indulni. (Miért logikusan? Az 1111 1111 fix rész az ugye FF, a flag értékek 0000, azaz 0, ezzel meg is van az FF0. Az x az meg a szkóp.)

- FF01::1 - interface-local scope all-nodes multicast address, azaz minden-node multicast cím, hoston belül[60].
- FF02::1 - link-local scope all-nodes multicast address, azaz minden-node multicast cím, linken belül. (Vegyük észre, hogy ez tulajdonképpen a broadcast! Nincs is külön, megszűnt. Ez a multicast cím van helyette.)
- FF01::2 - interface-local scope all-routers multicast address, azaz minden-router multicast cím a hoston belül.
- FF02::2 - link-local scope all-routers multicast address, azaz minden-router multicast cím a linken belül.
- FF05::2 - site-local scope all-routers multicast address, azaz minden router multicast cím a site-on belül.

Rengeteg egyéb előre definiált multicast cím létezik - pl. összes DHCPv6 szerver, meg egyéb ínyencségek. Tessék utánaolvasni.

IPv6 multicast címek:
http://www.iana.org/assignments/ipv6-multicast-addresses

[60] Tipikus több hálózati kártyával (node) rendelkező host.

Gondolom, elég sok ez így első körben. Ismétlésképpen nézzük végig, milyen címekre is kell hallgatnia egy mezei hálókártyának:

- link-local cím
- unique global
- unique local
- loopback
- minden-node hoston belül
- minden-node linken belül
- solicited-node (ez majd csak később fog megjelenni a könyvben)
- egyedi multicast címek

Durva egy kicsit. Ez ugyanis mind egy-egy IP cím, mely a kártyához rendelődik. Alaphelyzetben.

Jogos lehet a kérdés, hogyan fogunk ezek között a címek között tájékozódni? IPv4 esetén a rutinos rendszergazda ránézett egy IP címre és már vágta is, hogy az privát vagy publikus. Mi a helyzet az IPv6 esetében?

4.21. 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[61]	FD00::	FDFF::FFFF:FFFF:FFFF:FFFF
Unique local #2[62]	FC00::	FCFF::FFFF:FFFF:FFFF:FFFF
Multicast[63]	FF01::0001	FF35::FFFF:FFFF

Nem egyszerű, az tény. De nem is megtanulhatatlan.

Most, hogy már van némi fogalmunk az IPv6 címzésről, érdemes lehet végignézni annak a gépnek az IP konfigurációját, melyen jelenleg is gépelek:

```
Windows IP Configuration

Host Name                         :hq
Primary Dns Suffix:
        Node Type                 : Hybrid
        IP Routing Enabled        : No
        WINS Proxy Enabled        : No

Ethernet adapter Local Area Connection:

Connection-specific DNS Suffix  . :
Description                       Realtek RTL8168B/8111B Family PCI-E Gigabit Ethernet NIC (NDIS 6.0)
Physical Address                  : 00-1E-8C-AB-2F-88
DHCP Enabled                      : Yes
Autoconfiguration Enabled: Yes
Link-local IPv6 Address           : fe80::c5cc:1c5c:8384:4546%8(Preferred)
IPv4 Address                      : 192.168.1.101(Preferred)
Subnet Mask                       : 255.255.255.0
```

[61] Pszeudorandom algoritmussal.
[62] Jövőbeni algoritmussal - ergo ez a tartomány még nem él.
[63] Nem folytonos tartomány, még a definiált résztartományok közül sem aktív mindegyik.

```
Lease Obtained                  : 2008. június 3. 19:14:09
Lease Expires                   : 2008. június 4. 19:14:08
Default Gateway                 : 192.168.1.1
DHCP Server                     : 192.168.1.1
DHCPv6 IAID                     : 201334412
DNS Servers                     : 84.2.44.1
                                  84.2.46.1
NetBIOS over Tcpip              Enabled

Tunnel adapter Local Area Connection* 6:

Connection-specific DNS Suffix  . :
Description                     : Teredo Tunneling Pseudo-Interface
Physical Address                : 02-00-54-55-4E-01
DHCP Enabled                    : No
Autoconfiguration Enabled       : Yes
IPv6 Address                    : 2001:0:d5c7:a2ca:3c2f:2bc6:3f57:fe9a(Preferred)
Link-local IPv6 Address         : fe80::3c2f:2bc6:3f57:fe9a%9(Preferred)
Default Gateway                 : ::
NetBIOS over Tcpip              : Disabled

Tunnel adapter Local Area Connection* 7:

Connection-specific DNS Suffix  . :
Description                     : isatap.{47CE5CAA-223F-4CD4-9A17-D91D7DDC2066}
Physical Address                : 00-00-00-00-00-00-00-E0
DHCP Enabled                    : No
Autoconfiguration Enabled       : Yes
Link-local IPv6 Address         : fe80::5efe:192.168.1.101%10(Preferred)
Default Gateway                 :
DNS Servers                     : 84.2.44.1
                                  84.2.46.1
NetBIOS over Tcpip              : Disabled
```

Illetve a route tábla:

```
===========================================================================
Interface List
8       00 1e 8c ab 2f 88               Realtek  RTL8168B/8111B  Family  PCI-E  Gigabit  Ethernet  NIC
(NDIS 6.0)
1                                       Software Loopback Interface 1
9       02 00 54 55 4e 01               Teredo Tunneling Pseudo-Interface
10      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
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:
None

IPv6 Route Table
===========================================================================
Active Routes:
If       Metric   Network Destination                          Gateway
9        18       ::/0                                        On-link
1        306      ::1/128                                     On-link
9        18       2001::/32                                   On-link
9        266      2001:0:d5c7:a2ca:3c2f:2bc6:3f57:fe9a/128    On-link
```

```
8        276      fe80::/64                                   On-link
9        266      fe80::/64                                   On-link
10       281      fe80::5efe:192.168.1.101/128                On-link
9        266      fe80::3c2f:2bc6:3f57:fe9a/128               On-link
8        276      fe80::c5cc:1c5c:8384:4546/128               On-link
1        306      ff00::/8                                    On-link
9        266      ff00::/8                                    On-link
8        276      ff00::/8                                    On-link
===========================================================================
Persistent Routes:
None
```

Csemegézzünk.

Vajon mi lehet az IPv6 címem? Vigyázat, beugrató kérdés, ugye elég sok lehet. De jelen esetben csak link-local címeket látunk, az egyik pl. így néz ki: fe80::c5cc:1c5c:8384:4546%8. Ránézésre több baj is van a címmel. Először is, mi az a %8 a végén? Aztán az INTERFACE ID egyáltalán nem úgy néz ki, mintha a MAC address-ből képezték volna betoldással. Árulás? Nos, a %8 formulára egyszerű a válasz: hétköznapi cím esetén ez az interface azonosító száma, úgynevezett zónaazonosító. (A route táblánál látható, hogy konkrétan egy Realtek hálókártyáról van szó.)

Jó, akkor mi van az INTERFACE ID-val? Az, hogy az EUI-64 nem kötelező. A Windows Server 2008 illetve a Vista alaphelyzetben például véletlenszerűen generált interface ID-t használ. Erről a következő parancs segítségével tudjuk lebeszélni:

```
netsh interface ipv6 set global randomizeidentifiers=disabled
```

Miután kiadtam ezt a parancsot és megújítottam az IP címemet, a következőt kaptam:

```
Link-local IPv6 Address: fe80::21e:8cff:feab:2f88%8
```

Ugye, mindjárt más? Ez már EUI-64. A matyó csujjogatással.

Miért is nincs globálisan vagy lokálisan egyedi (global/local unique) címem? Mert a routerem azt mondta, hogy nekem olyanom nincs. Pontosabban, nem mondta, hogy van.

Aztán nagyon elszaladtunk amellett, hogy nekem nem is egy link-local IP címem van. Mi is a többi? Nos, transition címek. Azok, melyeket csak úgy megemlítettem korábban. Most sem kapnak nagy szerepet... de azért vegyünk észre valamit: mindkettő, a Teredo és az Isatap is Tunnel Adapter Local Area Connection néven fut. Tunnel... valami belecsövezve valamibe. Mi van most? IPv4. Mi lesz majd? IPv6. Nyilván logikus, hogy ezeket a címeket használva tulajdonképpen az IPv6-ot csatornázzuk bele az IPv4-be.

Végül egy kérdés: ha teszem azt a Magyar Telecom globális azonosítója 1100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1 és a én ezen belül a 0000 0000 0000 0011 subnetben vagyok, akkor mi lesz a világegyetemben az egyedi azonosítóm?
Lapátoljuk össze: 001, mert ez abszolút fix. Ehhez hozzácsapjuk az ISP globális azonosítóját és a subnet azonosítót. Ez lesz a prefix, ehhez jön majd az INTERFACE ID, melyet a link-local címből tudunk kibányászni. Jelen esetben ez c5cc:1c5c:8384:4546.
Innen már csak matekozás: 3800::1:3:c5cc:1c5c:8384:4546/64.

Long live calc.exe.

(Hint: fel kell írni bitsorozatként, bájtonként csoportosítani, átváltani, két bájt egy blokk.)

Végezetül egy táblázat, arról, melyik IPv6-os címnek melyik IPv4-es cím felel meg.

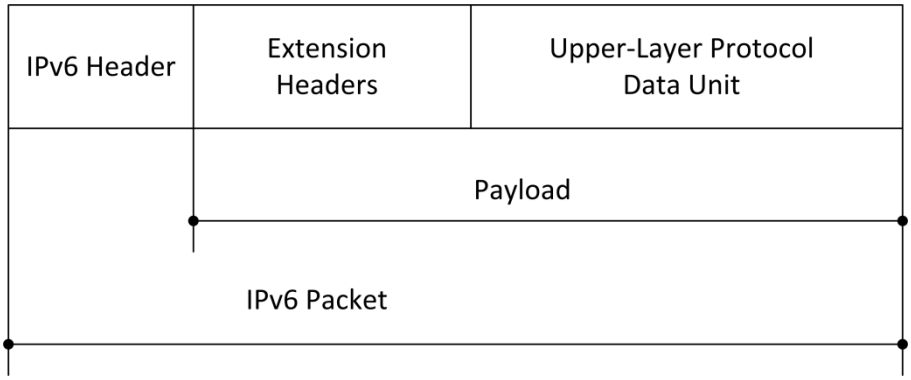
4.22. TÁBLÁZAT

IPv4	IPv6
Internet címosztályok	Elfogyott
Multicast címek (224.0.0.0/4)	Multicast címek (FF0x::/8)
Broadcast címek	Szintén nincs
Meghatározatlan cím (unspecified address): 0.0.0.0	Meghatározatlan cím (unspecified address): ::[64]
Loopback: 127.0.0.1	Loopback: ::1
Publikus IP címek	Global Unicast Address
Privát IP cím tartományok	Unique Local, Site Local Addresses
APIPA (168.254.0.0/16)	Link Local Addresses

[64] Kicsit sok a kettőspont, de gondolom érthető.

4.2.1.3 Az IPv6 Csomag

Az IPv6 csomag felépítése némileg eltér az eddig megszokott csomagokétől. Általában volt olyan, hogy header, meg payload (néhol trailer is) - de olyannal eddig még nem találkoztunk, hogy a header átcsússzon a payloadba.
Eddig.

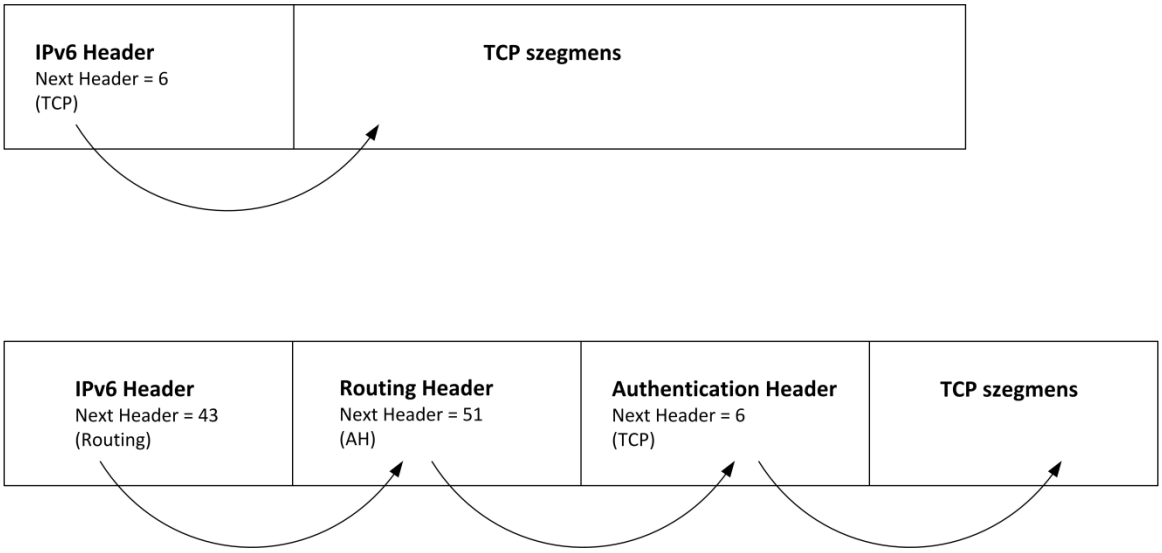


4.50. ábra IPv6 csomag

Van packet és van header, odáig rendben is volnánk - de van egy extension header a payload-on belül. Sőt, nem is egy, mert az ábrán is többes számmal lett jelölve.

Bizony, bizony. Így lett rugalmas az IPv6. Van egy rögzített fejléce, van egy rögzített PDU-ja - ez volt az IPv4-ben a payload - és a kettő között van egy tetszőlegesen bővíthető gumi gyűrődőzóna.

Habár még nem minden része érthető a rajznak, de valami ilyesmit kell elképzelni:



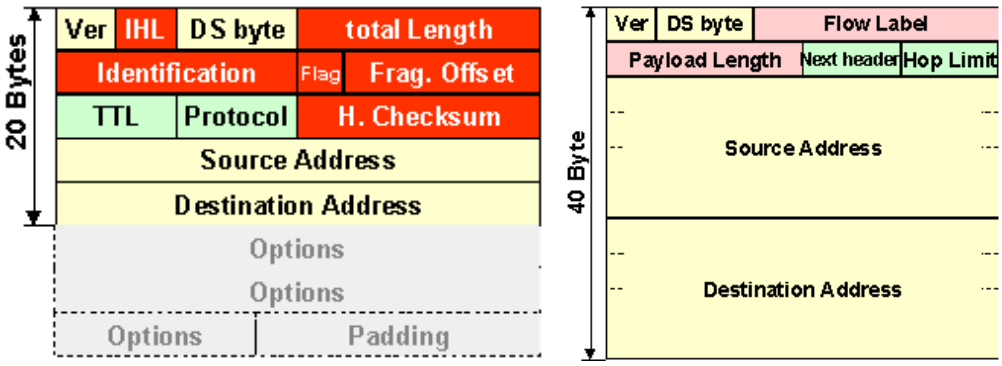
4.51. ÁBRA FEJLÉC LÁNCOLÁSOK

A felső ábrán nincs Extension Header, a fejléc közvetlenül a payload-dal, azaz a TCP szegmenssel folytatódik. Az alsó ábrán jelentősen bonyolultabb lett a helyzet, két extra fejléc is befurakodott az IPv6 fejléc és a payload közé.

Az Upper-Layer Protocol Data Unit-tal (ez neveztem PDU-nak) van a legkevesebb gondunk: mint írtam, ez felel meg az IPv4 payload-nak, azaz lehet TCP, UDP, ICMPv6.

A többi mezővel viszont érdemes részletesebben is megismerkedni.

4.2.1.3.1 IPV6 HEADER



4.52. ÁBRA IPCSOMAGOK FEJLÉCEINEK ÖSSZEHASONLÍTÁSA

Nos, itt vannak. Melyik a nagyobb? Na...? Ha jól figyeltél korábban, akkor már tudod a korrekt választ: 'attól függ'. Láthatod, hogy az IPv4 csomag áll egy fix - 20 bájt - méretű részből, majd jön az IP Options mezők végeláthatatlan sora. (Maximum 60 bájt, természetesen néggyel oszthatóra kiegészítve.)

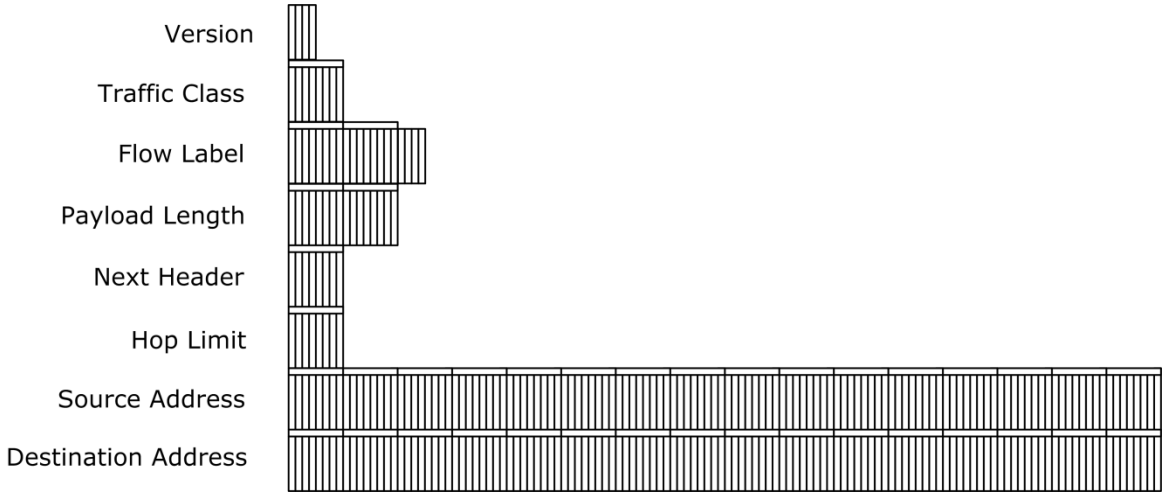
Ezzel szemben az IPv6 fejléc mérete fix 40 bájt. Tessék kérem, újabb és kisebb - pedig az IP címek négyszerannyi helyet foglalnak.

Na, ja. Csakhogy az IP Options mezőket pofátlanul kiszervezték a payload-ba, és hogy a PSzÁF se jöjjön rá, átnevezték Extension Headers névre.

Ami nem változott, az az IP csomag (header+payload) maximális mérete: az továbbra is 65535 bájt[65]. Azaz tulajdonképpen látható, hogy csak a dögöt rugdostuk át az egyik ágy alól a másik alá, a fejléc mérete a payload méretének rovására csökkent.

[65] Illetve az IPv6 csomag lehet nagyobb is, mint 65535 bájt, ekkor jumbogram-nak hívjuk, és némileg máshogyan kezeljük.

Boncolgassuk egy kicsit az IPv6 fejlécet.



4.53. ÁBRA AZ IPV6 CSOMAG FEJLÉCE

VERSION: Értelemszerűen az értéke: 6.

TRAFFIC CLASS: Teljes mértékben analóg az IPv4 csomagban lévő DS bájttal. (Tudjuk, 6 bit DSCP, 2 bit ECN - ezt az egészet Type of Services-nek (TOS) hívják és a QOS-hez kell.[66])

FLOW LABEL: 20 bites mező, egy konkrét, speciálisan kezelendő adatforgalmat jelöl.

RFC 3697

Olyan forgalomról van szó, mely az IPv4-ben elég mostohán volt kezelve - tekintve, hogy a hetvenes években ezeknek túl sok szerepük nem volt. A stream jellegű adatfolyamokról beszélek. A FLOW LABEL, mint cimke azonosítja az egyes adásokhoz tartozó csomagokat. (Mindemellett ha az értéke nem nulla, akkor jelzi a routereknek is, hogy ezeket a csomagokat különleges elbánásban kell részesíteniük.)

[66] Eddig ez a legvadabb mondat a könyvben.

PAYLOAD LENGTH: Mekkora a payload konkrét értéke. Ugye a bevezetőben már láttuk, hogy az Extension Headers miatt a payload értéke nem rögzített. Gondolom az előző oldalon említett 65535-ös felső korlát miatt senkit nem lep meg, hogy a mező mérete két bájt.

HOP LIMIT: Tulajdonképpen ugyanarra szolgál, mint az IPv4 esetében a TTL. A feladó induláskor beállít egy értéket majd mindegyik router lekap belőle egyet áthaladáskor. A csomag akkor hal hősi halált, ha ez az érték lenullázódott.

NEXT HEADER: De hiszen ezt már láttuk! (*4.51. ábra Fejléc láncolások*) Ez tulajdonképpen egy kód, azt mutatja meg, hogy a következő blokkban mi jön: valamilyen Extension Header vagy már a tényleges PDU.

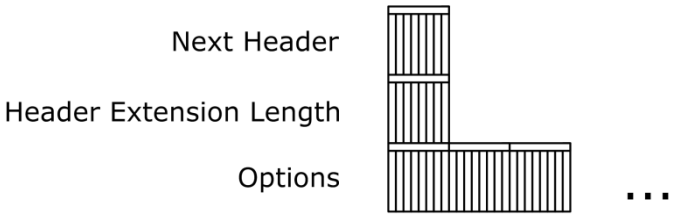
4.2.1.3.2 IPv6 Extension Headers

Mint említettem, az IPv4-ben használt IP Options blokkokat helyettesítik.

4.23. táblázat

Next Header Field érték	Megnevezés
0	Hop-by-Hop Options header
6	TCP
17	UDP
41	Encapsulated IPv6 header
43	Routing Header
44	Fragment header
50	Encapsulating Security Payload header
51	Authentication header
58	ICMPv6
59	Konyec, nincs több header
60	Destination Options header

A felépítésük is teljesen hasonló.



4.54. ábra Általános Extension Header

Next Header: Ez mutat a következő blokkra.

Header Extension Length: Tekintve, hogy az Extension Header mérete nem rögzített, valahogyan jelezni kell, hol van a vége.

Options: Itt pedig jönnek a tipusoktól függő adatblokkok. A felépítésük teljesen analóg az Extension Header felépítésével, azaz jön egy típusazonosító kód (mivel a konkrét Extension Header-en belül is lehetnek alesetek), egy hosszérték és maga az adat. (Igen, Matrjoska baba.)

4.2.1.3.2.1 HOP-BY-HOP OPTIONS HEADER

A nullás kódú. Ez az a header, melyet minden útbaeső hop felolvas, értelmez.

Rögtön egy olyan ~~opciót~~ header-t találtunk, amelynek több altipusa (options) is létezik.

RFC 2460

PAD**1**/PAD**N** OPTIONS: 1, illetve tetszőleges számú üres bájtot fűz hozzá a Hop-by-Hop header-hez. Amennyiben jól értettem, ezekre a feltöltésekre számolási optimalizálások miatt van szükség.

RFC 2675

JUMBO PAYLOAD OPTION: Egy félmondat már esett arról, hogy az IPv6 már támogat 65535 bájtnál nagyobb csomagot is. Senkinek nem támadt kényelmetlen érzése: hogyan is lehetséges ez, ha a PAYLOAD LENGTH mező mérete csak 16 bit? Nos, úgy, hogy a csomag mérete nem ott tárolódik, hanem a Jumbo Payload option 32 bites adatterületén. Ez maximum 4 GB méretű payload-ot enged.
Értjük már, hogy miért Jumbo?

RFC 2711

ROUTER ALERT OPTION: Figyelmezteti a routert, hogy ezzel a csomaggal még lesz egy kis pluszmunkája. (Emlékszünk, multicast-nál voltak ilyesmik korábban.)

4.2.1.3.2.2 Destination Options Header

A csomag szállítására vonatkozó paramétereket tartalmazhat. Értelmezni kétféleképpen lehet:

- Ha van mellette Routing Header, akkor minden útbaeső célpontra vonatkozni fognak a paraméterek.
- Ha nincs mellette Routing Header, vagy a Routing Header előrébb van, mint a Destination Options Header, akkor a paraméterek csak a legvégső célpontra fognak vonatkozni.

RFC 3775

Mik lehetnek ezek a paraméterek? Például a Home Address Option. Ezt akkor használjuk, ha mobil eszközön kommunikálunk - olyan mobil eszközön, mely ténylegesen is mozog. Ekkor az Option adatmezőjébe belekerül a mobil eszköz hazai címe - az a cím, mely eredetileg, a mobil eszköz eredeti linkjén tartozott hozzá.[67]

4.2.1.3.2.3 Routing Header

Ilyet is láttunk már korábban. (Strict/Loose Source Routing; *4.1.1.2 IP Options*)
Ez az, amikor előírjuk a csomagnak, hogy milyen útvonalon kell mennie.
Az IPv6-ban a szigorúság enyhült, azaz a Routing Header már csak a loose tipusú előírást támogatja.
A Windows meg még ezt se. A Windows Vista eleinte még elvolt vele, de aztán az IETF biztonsági szempontok miatt inkább azt javasolta, hogy hanyagolják. A Vista SP1-ből emiatt ki is vették. (A Windows Server 2008-ba meg már bele sem került.)
Ha ilyen csomagokat kapnak, akkor szó nélkül eldobják azokat.

4.2.1.3.2.4 Fragment Header

A jó öreg töredezés. Mikor is van rá szükség? Ha a csomag túl nagy. Mekkora a túl nagy: 65535 felett?
Dehogyis. Az MTU felett. Emiatt kezdjük úgy a kommunikációt, hogy meghatározzuk a PMTU értékét. Inkább, mint hogy tördeljük a csomagot.

Ha már az IPv4-ben sem szerettük a tördelést, akkor gondolhatod, hogy az IPv6-ban úgy utáljuk, mint üveges tót a hanyattesést. Például a közbenső hop-oknak tilos. Vagy a feladó tördel, vagy senki sem.

[67] A mobil IPv6-ról majd valamikor máskor.

4.2.1.3.2.5 Authentication Header

RFC 2401, 2402

A csomag integritását biztosítja egy ún. Integrity Value Check (ICV) összeg segítségével. A számoláshoz értelemszerűen nem használja fel az IP Header összes mezőjét, hiszen ezek között vannak olyanok, melyek menet közben normál üzemmenetben is változnak.
Az integritás biztosítása mellett még az AH dolga a forrás autentikáltatása és egy ún visszajátszás elleni védelem is. Ez utóbbi abból áll, hogy a csomag része egy Sequence Number, ezt minden hozzányúló hop megnöveli eggyel. (Tessék, rögtön itt is van egy elem, melyet nem lehet belevenni az ICV-be.)

4.2.1.3.2.6 Encapsulating Security Payload Header

RFC 2406

Az AH sok mindent nyújt - titkosítást speciel nem. Arra az ESP jó.

Mind az AH, mind az ESP Header, illetve a mögöttük lévő eljárások meglehetősen komplex rendszert alkotnak, ezekkel majd később foglalkozunk.

4.2.1.3.3 ICMPv6

RFC 4443

Nagyjából ugyanazok a feladatok, mint az ICMPv4 esetében - csak valamivel áramvonalasabb adatszerkezettel. Mások lettek a mezők, a nevek, a kódok - de a funkciók maradtak.

Mivel ezekről már volt szó bőven ebben a könyvben, így csak egy táblázatban teszem egymás mellé az azonos funkciókat.

4.24. TÁBLÁZAT

ICMPv4	ICMPv6
Destination Unreachable-Network Unreachable (Type3, Code 0)	Destination Unreachable-No Route to Destination (Type 1, Code 0)
Destination Unreachable-Host Unreachable (Type 3, Code 1)	Destination Unreachable-Address Unreachable (Type1, Code 3)
Destination Unreachable-Protocol Unreachable (Type 3, Code 2)	Parameter Problem-Unrecognized Next Header Type Encountered (Type4, Code 1)
Destination Unreachable-Port Unreachable (Type 3, Code 3)	Destination Unreachable-Port Unreachable (Type 1, Code 4)
Destination Unreachable-Fragmentation Needed and DF Set (Type 3, Code 4)	Packet Too Big (Type 2, Code 0)
Destination Unreachable-Communication with Destination Host Administratively Prohibited (Type 3, Code 10)	Destination Unreachable-Communication with Destination Host Administratively Prohibited (Type 1, Code 1)
Source Quench (Type 4, Code 0)	Nyema
Redirect (Type 5, Code 0)	Neighbor Discovery Redirect message (Type 137, Code 0)
Time Exceeded-TTL Exceeded (Type 11, Code 0)	Time Exceeded-Hop Limit Exceeded in Transit (Type 3, Code 0)
Time Exceeded-Fragment Reassembly Time Exceeded (Type 11, Code 1)	Time Exceeded-Fragment Reassembly Time Exceeded (Type 3, Code 1)
Parameter Problem (Type 12, Code 0)	Parameter Problem (Type 4, Code 0 v. 2)

4.2.1.4 NEIGHBOR DISCOVERY, ND

RFC 4861, 4862

Mint a neve is mutatja, arról van szó, hogy az azonos linken lévő node-ok valamilyen mechanizmus alapján megkeresik egymást. Azonkívül, hogy jó társaságban jobban telik az idő, a módszernek van több előnye is:

- Ha csomagot kell küldenünk, jóval hatékonyabban be tudjuk szerezni a célállomás MAC címét.
- Ha egy MAC address megváltozik, automatikusan értesülünk róla. Hasonlóképpen arról is, ha a hostot lekapcsolják. Vagy lefagy.
- Képbe kerülünk: tudni fogjuk, hol vannak a linken routerek.
- DHCP nélkül valósíthatunk meg autokonfigurációt.
- A routereknek lehetőségük van a szolgáltatásaikat reklámozni. (Ráadásul nem is kell fizetniük érte.)
- A routerek átirányíthatnak forgalmakat, ha létezik jobb útvonal.

Érezzük, hogy ez több dologból lett összegyúrva. Ott vannak benne az ICMPv4 fejezetben taglalt mechanizmusok, de a MAC vadászatra korábban használt ARP folyamatok is.

Itt mindent ICMPv6 üzenetekkel fogunk megvalósítani. Konkrétan:

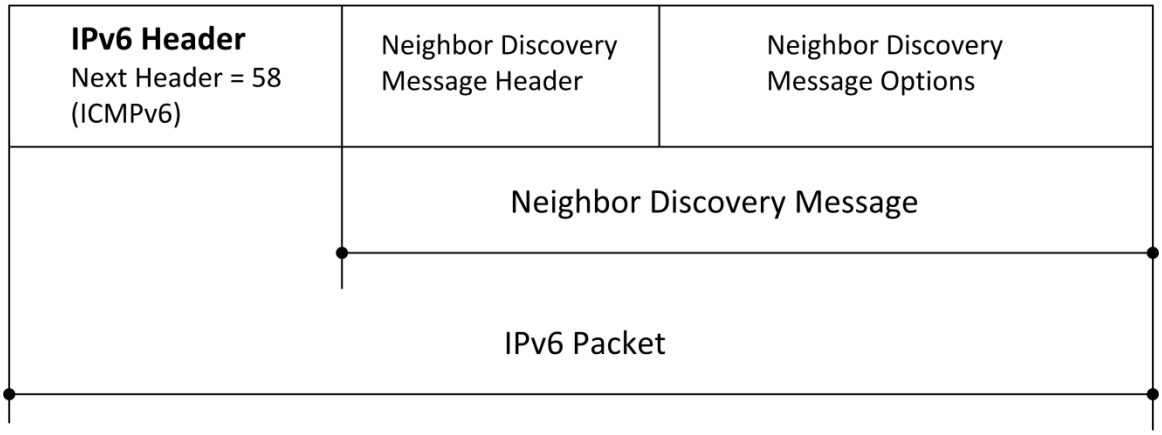
- Router Solicitation (Type 133)
- Router Advertisement (Type 134)
- Neighbor Solicitation (Type 135)
- Neighbor Advertisement (Type 136)
- Redirect (Type 137)

Referenciatáblázat az IPv4 és az IPv6 mechanizmusok között:

4.25. TÁBLÁZAT

IPv4	IPv6
ARP Request message	Neighbor Solicitation message
ARP Reply message	Neighbor Advertisement message
ARP cache	Neighbor cache
Gratuitous ARP	Duplicate Address detection
Router Solicitation message	Router Solicitation message
Router Advertisement message	Router Advertisement message
Redirect message	Redirect message

Csomagszinten úgy kell elképzelni, mintha ICMPv6 lenne. Leginkább azért, mert ugye, az.

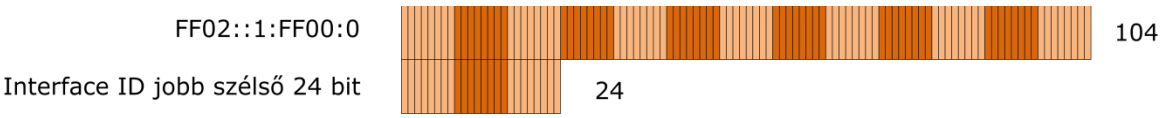


4.55. ÁBRA NEIGHBOR DISCOVERY ÜZENET

Az eddigiektől eltérően ebbe a fejezetbe nem mennék bele annyira részletesen - inkább csak elv szintén mutatnám be a Neighbor Solicitation / Neighbor Advertisement folyamatokat.

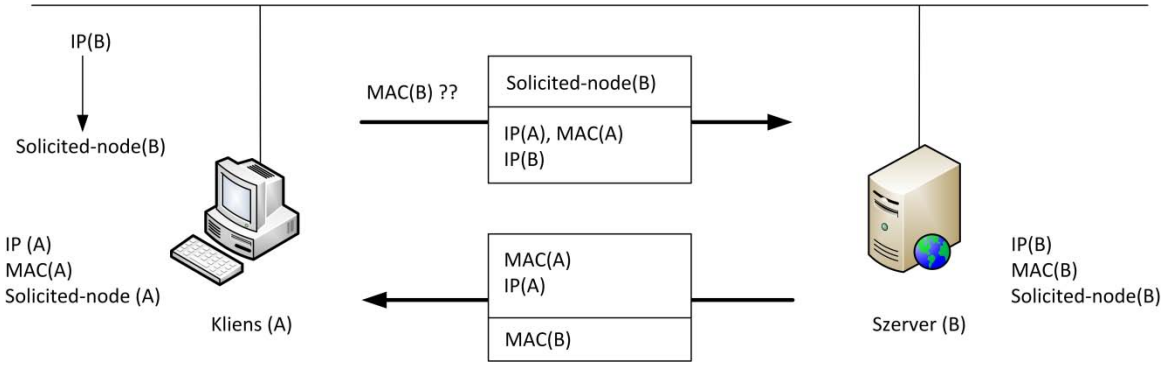
Azt tudjuk, hogy ez a két üzenet típus az ARP Request / Reply üzeneteket váltja fel. Mi is volt az ARP Request legnagyobb hibája? Az ARP broadcast. Egyrészt nagy terhelést jelentett a hálózatnak, másrészt pedig a routerek lenyakazták.

Az IPv6-ban ugyan eljátszhatták volna mindezt a minden-node link-local címre küldött üzenettel, de akkor ugyanott lennénk, mint az ARP broadcast-tal.
Ehelyett egy speciális multicast címre megy ki az érdeklődő üzenet. Ezt a speciális címet hívják úgy, hogy solicited-node cím. A következőképpen képződik:



4.56. ÁBRA SOLICITED-NODE

A megértés kulcsa az alsó mező. Kinek is az Interface ID-ja? Nyilván nem a feladóé. Annak nem lenne semmi értelme. Természetesen a keresett IP címből - IP(B) - képződik a solicited-node cím.



4.57. ÁBRA MŰKÖDÉSBEN A SOLICITED NODE CÍMZÉS

Az érdeklődő (A) node tehát legyártja ezt a solicited -node címet és elküldi neki a saját IP címét - IP(A) - saját MAC címét - MAC(A) - és persze a keresett IP címet - IP(B). Mely node-ok fogják felkapni ezt a kérést? Akiknek a - saját maguknak már korábban legyártott solicited-node - címük megegyezik a feladó által megadott solicited-node multicast címmel. Egész konkrétan: az a pár node fogja felkapni, ahol az interface ID jobb szélső 24 bitje megegyezik. A többi node békésen kérődzik tovább. Tiszta haszon az ARP broadcast-hoz képest.
A többi már hasonló: a megtalált node visszaküldi a MAC(B) címet és már mehet is a traccs.

Egy újabb gondolkodós kérdés: a korábban bemásolt adataim alapján vajon mi is lesz az én solicited-node címem?
Az első 104 bit, az ugye adott. A maradék 24 bit pedig kiszámolható az Interface ID-ból.
Merre is vagy, calc.exe?
Tessék, a cím: ff02:1::ff00:0084:4546[68]. Most az egyszer nem segítek.

Elképzelhető, hogy valakiben felmerül a kérdés: mire is jó ez az egész cécó? Hiszen az interface ID a MAC address-ből képződik... egyszerűen számoljunk belőle vissza és már miénk is a MAC.
Az ötlet jó... de sajnos ez csak ajánlás. Az interface ID sokféleképpen keletkezhet, semmi garancia sincs rá, hogy pont az EUI-64 volt a keletkezésének alapja.

[68] Azt ugye látjuk, hogy az értéke (ff02) alapján ez egy link-local multicast cím?

4.2.1.5 MULTICAST LISTENER DISCOVERY (MLD)

RFC 2710

MLD: ez volt az IPv4-ben az IGMP. Pontosabban az IGMPv2.

RFC 3810

Az IGMPv3-nak pedig az MLD2 felel meg.

Oké, képben vagyunk. Jöhetnek a részletek.

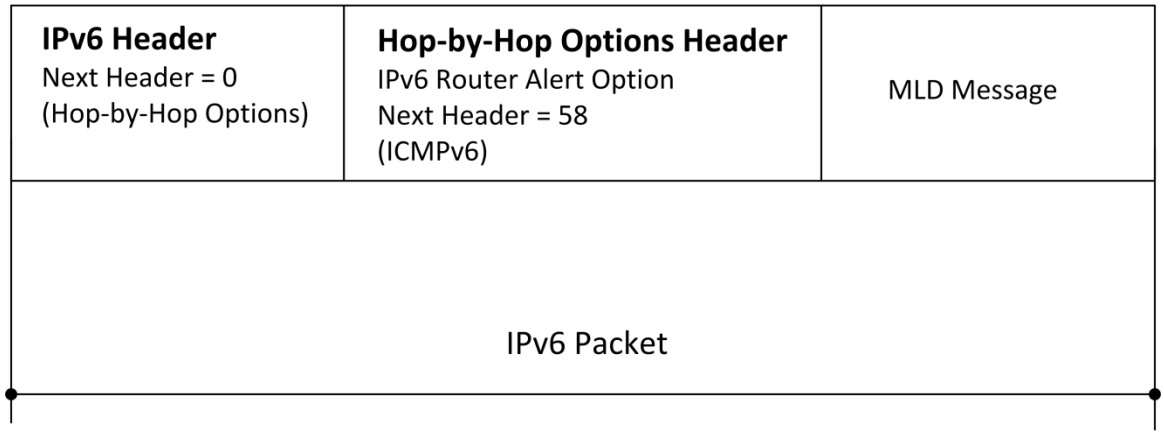
Rögtön egy különbség: IPv4-ben egy host vagy támogatta a multicast küldést/fogadást, vagy nem. IPv6-ban nincs ilyen lazaság: a multicast ismerete kötelező.

Tudom, szomorú leszel, de a részletekbe olyan túlzottan azért nem fogok belemenni. Egyrészt az MLD tényleg nagyon hasonlóan működik, mint az IGMP, így a mechanizmusok ismertetésébe nincs értelme túlzottan elmélyedni.
Itt is vannak üzenetek, az analógiákat az alábbi táblázat mutatja:

4.26. TÁBLÁZAT

IGMPv2	MLD
Host Membership Query (Type 17)	Multicast Listener Query (ICMPv6 Type 130)
Host Membership Report (Type 22)	Multicast Listener Report (ICMPv6 Type 131)
Leave Group (Type 23)	Multicast Listener Done (ICMPv6 Type 132)

Maguk az üzenetek - mint a táblázatból is látható - ICMPv6 tipusú üzenetek. Mint a Neighbor Discovery-nél. A csomag szerkezete is hasonló:



4.58. ÁBRA MLD ÜZENET

Hasonló... de nem ugyanaz. Vegyük észre, hogy egy Extension Header befurakodott a képbe. Router Alert Option. Mire is jó ez? Lapozzunk vissza.
Úgy van. Figyelmezteti a routert, hogy ezzel a csomaggal lesz némi adminisztrációs munkája. (Lásd IGMP.)

4.2.1.6 AUTOKONFIGURÁCIÓ

RFC 2464

Ez megint egy szép téma. Kezdjük megint azzal, hogy milyen lehetőségünk volt autokonfigurációra IPv4 alatt? Igen, a DHCP. (Illetve, bármilyen furcsán hangzik, de egyfajta stateless autokonfiguráció volt az APIPA is.)

Az IPv6 alatt tágabb lett a horizontunk:

- Stateless autoconfiguration
- Stateful autoconfiguration
- Kombinált bérlet

Az egész úgy kezdődik, hogy a router - ha van egyáltalán - kiküld a linkre egy Router Advertisement üzenetet. Ennek meglététől, illetve tartalmától függően az egyes hostok eldöntik, hogy itt most a fenti három autokonfigurációs esetből melyik áll fent.

Itt most csak a stateless változatot vizsgálgatjuk, a DHCP működésével majd az alkalmazás rétegben fogunk megismerkedni.

RFC 4862

Van egy hostunk, egy hálózati kártyával. Feldugjuk egy IPv6 hálózatra. DHCPv6 nincs. Router van.
A router meghatározott időnként küld egy üzenetet (Router Advertisement) a minden-node link címre. Az üzenet sokmindent tartalmaz, többek között:

- A router MAC címét
- A linken élő prefixeket
- A linken érvényes MTU-t
- A linkre vonatkozó spéci routolásokat
- A linken érvényes maximális hop számot
- Van-e DHCP a linken?

És még sok egyebet. Mennyire jó ez nekünk? Nagyon. Központilag szabályozható MTU? Hmm... finom. A link összes prefixe? Álljunk csak meg. Ha adottak a prefixek (régi szóhasználatban a subnetek) - Router Advertisement, azon belül Prefix Information opció - a hálókártya meg ismeri a saját MAC címét, ismeri az EUI-64 módszert... simán le tudja gyártani magának a link összes prefixére a prefixnek megfelelő IPv6 címet.

Mennyivel jobb már, mint egy hasraütésszerűen kiadott cím a 169.254.0.0./16 IP tartományban?

Szóval belép az új állomás a linkre. Olyan korban élünk, hogy türelmetlenek vagyunk: nem várjuk meg, hogy a router körbeküldje az üzenetét, belerúgunk: Router, küldj üzenetet! (Router Solicitation.) És az küld. Ez alapján a node összerakja magának a link-local címeit. Elképzelhető, hogy ütközni fog másik node címeivel? Bizony, igen. Ekkor jön a jó öreg Neighbor Discovery. Megszólítjuk a solicited-node címen azt az IP címet, melyet magunknak kívánunk lefoglalni. Ha nem jön válasz, nyertünk. Miénk a cím.
(Ki kell rá térnem, hogy a Windows implementációnál ez is másképp van egy kicsit. Arról már írtam, hogy az Interface ID nem EUI-64 módon generálódik, hanem véletlenszerűen. Arról viszont nem, hogy a tervezők úgy saccolták, nagyon kicsi az esélye annak, hogy a véletlen azonosítók között egyformák legyenek: ezért elhagyták az ellenőrzést.)

Mi van, ha megváltozik a MAC címünk? Úgy csinálunk, mintha ND kérdéseket kaptunk volna, elárasztjuk a linket válaszokkal.

Mi van, ha két hálókártyával is kapaszkodunk egy linken? Hol ez, hol az fog válaszolni az ND kérésekre - terheléselosztás.

4.2.2 IPv4 -> IPv6 KONVERZIÓK

RFC 2893

Szóval itt van ez a szebb és jobb IPv6. De még mióta, hogy itt van. Hogyan fogunk erre áttérni? És mikor? Mikortól kell hexa-dec-bin konvertálós számológépet integrálni a rendszergazdák mobiltelefonjába?

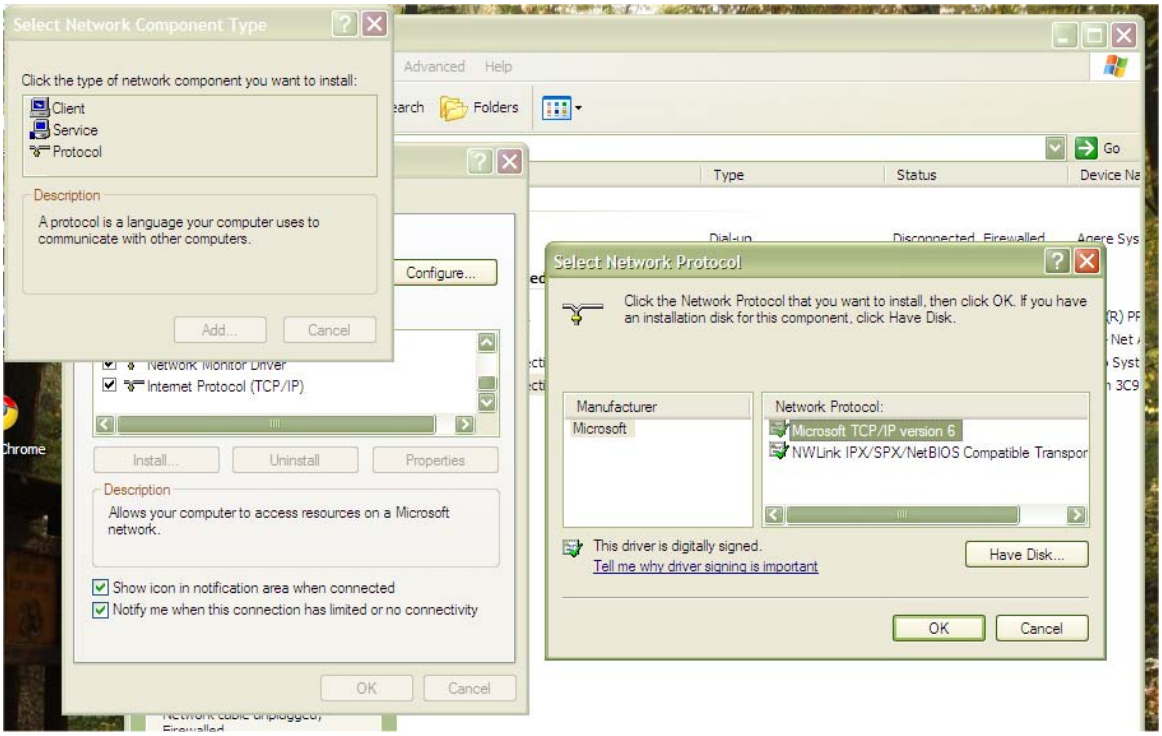
Próbáljuk meg kicsiben elképzelni. Van egy magyar értelemben kis/közepes céges hálózatunk: 1-200 gép, 10-15 szerver, egy core.net a központban, két telephely (site1.net, site2.net), meg egy dmz.net. Hogyan állnánk neki?

Első körben felállítanánk három kategóriát:

- IPv4 only host : olyan gép, amelyik csak az IPv4 módot ismeri.
- IPv6 only host : olyan gép, amelyik csak az IPv6 módot ismeri.
- IPv4/IPv6 host : olyan gép, amelyik mind a kettőt ismeri.

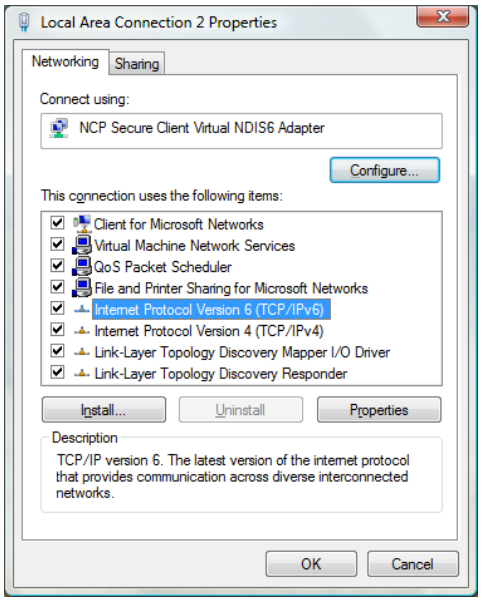
Mikor fogunk eljutni oda, hogy egy szerverünk IPv6 only legyen? Talán az unokánk.

Teljesen természetes, ha először azt célozzuk be, hogy minden host, minden router IPv4/IPv6 host legyen. Szerencsére ez ma már nem akkora nagy kihívás, a Service Pack 2 óta az XP is képes rá.



4.59. ÁBRA IPV6 TELEPÍTÉSE WINDOWS XP ALÁ

Értelemszerűen az utódok - Vista, Windows7 - már gyárilag beépítve tartalmazzák az IPv6 ismeretét.



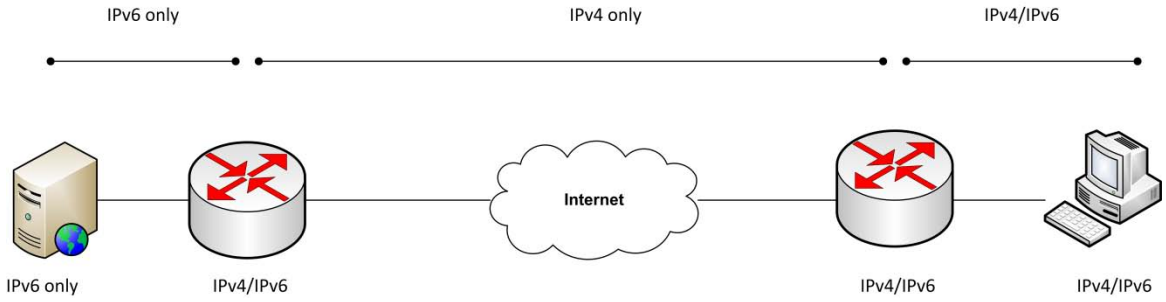
4.60. ÁBRA TIPIKUS IPV4/IPV6 HOST

Már csak az őskövület NT4 használókat[69] kell meggyőzni, hogy ha a későbbiekben is el szeretnék érni a hálózatot, akkor esetleg elkezdhetnének a frissítés gondolatával kacérkodni. (Bár valószínűleg inkább be fogják szerezni a Trumpet Winsock IPv6-ot.)
Ezzel a kliens oldal rendben is lenne. De ott vannak még a szerverek és a hálózati eszközök. Rossz hírem van: a Windows Server 2000 e tekintetben elég gázos. Amennyire tudom, nem erőszakolható meg. Viszont Windows Server 2003-tól felfelé már minden rendben. A hálózati eszközök cseréje meg pusztán csak elszántság és pénz kérdése.

IPv6 Microsoft rendszerekben:
http://www.windowsnetworking.com/articles_tutorials/IPv6-Support-Microsoft-Windows.html

Akkor már végeztünk is volna? Hát, a saját kis szemétdombunkon jó eséllyel igen. Be mernénk rakni IPv6 only munkaállomást? Miért ne? És szervert? Tuti, hogy a vezérigazgató egyik fontos külsős kapcsolata fog elhasalni, amikor be akar vépéenezni. DMZ-be webszervert? DMZ szélére tűzfalat, routert?
Nagyjából ezek a fenntartások léteznek ez interneten is. Jelenleg egyre több az IPv4/IPv6 host - de azért szerverben, hálózati eszközben még nagyon bátor az, aki IPv6 only hostot tesz elérhetővé.

Pedig a helyzet annyira nem is reménytelen. Mit saccolsz, ha van egy IPv4/IPv6 munkaállomásod, egy IPv4/IPv6 router mögött, mely router már közvetlenül csatlakozik az internetre - és el akarsz érni egy IPv6 only webszervert, mely előtt egy IPv4/IPv6 router van... sikerülni fog?

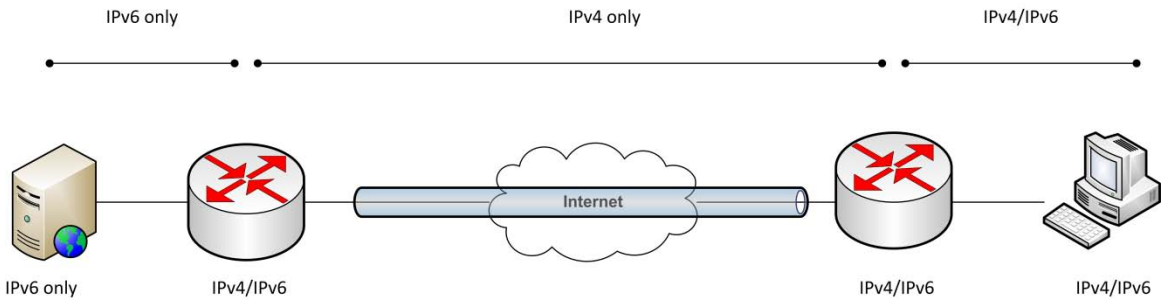


4.61. ÁBRA IPV6 ONLY WEBSZERVER ELÉRÉSE I

Igen.

[69] Ne röhögj, tudom, hogy nálatok is van.

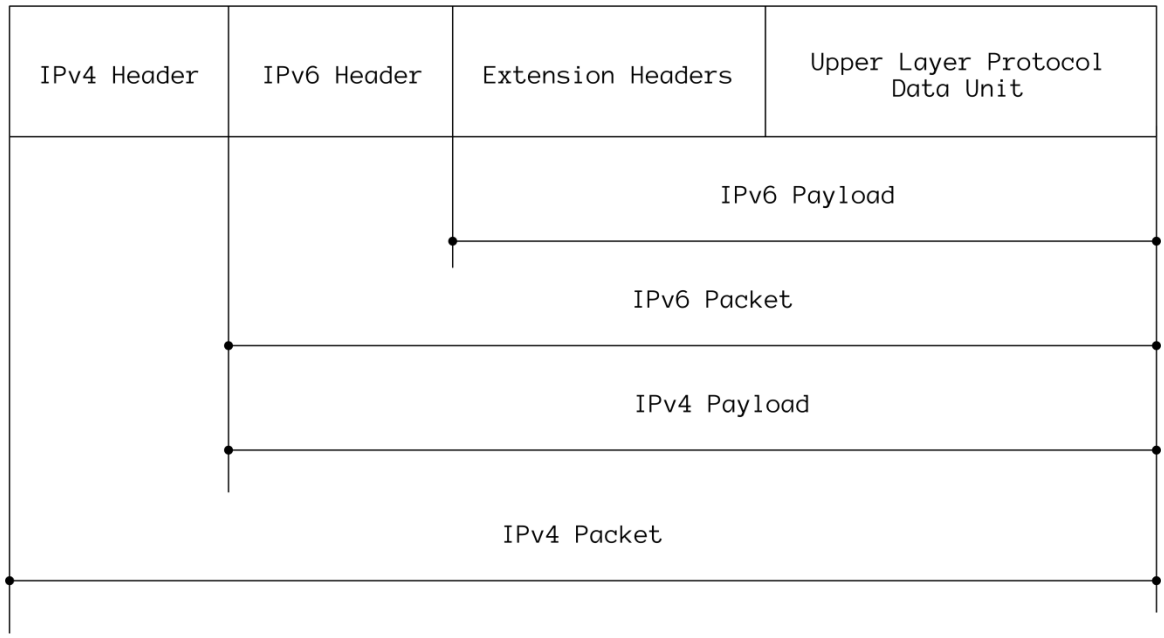
A kulcs a csatornázás, azaz a tunneling. Hol jön be a képbe az ismeretlen faktor? Hát az interneten. Jelenleg akkor kapunk jobb eredményt, ha az internetet IPv4 only hálózatnak feltételezzük.



4.62. ÁBRA IPV6 ONLY WEBSZERVER ELÉRÉSE II

Ekkor az történik, hogy a két router között létrejön egy csatorna - IPv6-over-IPv4 Tunneling - melyen keresztül az IPv6 csomagok IPv4 csomagokban utaznak át.

Hogyan is kell ezt elképzelni? Előveszek fizikailag egy vakondot. felhúzom kulccsal, beállítom a megfelelő irányba - és már fúrja is az utat az interneten keresztül?
Nyilván nem. Hogyan is lehetne értelmezni, hogy elkezdünk alagutat fúrni egy olyan megfoghatatlan valamibe, mint az internet?
Sokkal inkább arról van szó, hogy a megszokott csomagszerkezetbe valamelyik jól meghatározott ponton berakunk egy plusz csomagolást.



4.63. ÁBRA IPV6-OVER-IPV4 TUNNELING

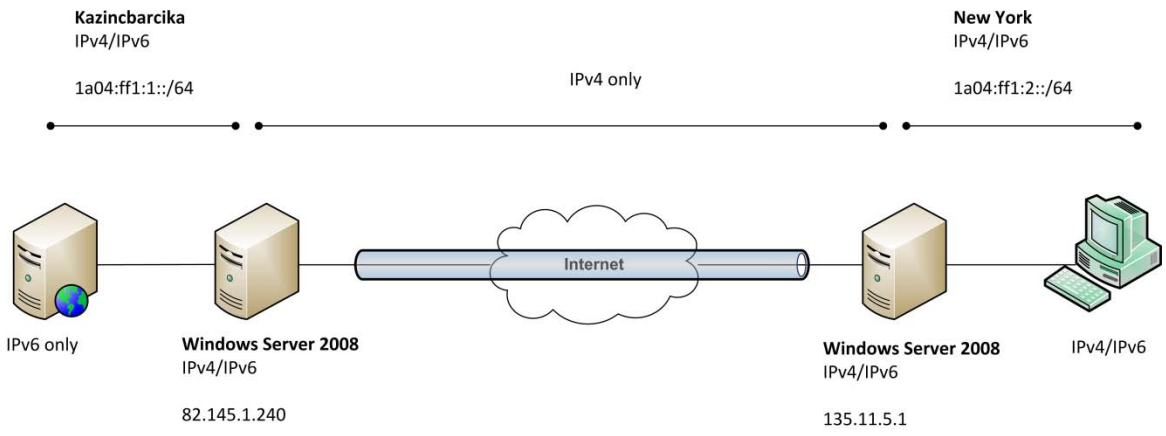
Amit így becsomagoltunk, azt megóvtuk attól, hogy útközben bárki piszkálja. Ez a plusz csomagolás lehet autentikációs fejléc, lehet titkosítás (később látunk mindegyikre példát) - és lehet egyszerűen inkompatibilis információkat elrejtő csomagolás.

Hogyan keletkezik egy ilyen csatorna?

- Vagy magunkat nem kímélve kemény fizikai munkával létrehozzuk.
- Vagy automatikusan keletkezik, felismerve, hogy itt bizony csatornázni kell.

Mikortól tekinthetünk kiépültnek egy csatornát? Nagyon egyszerű: ha a két végpontja beazonosította magát és létrejött köztük a kapcsolat. A legszebb az egészben, hogy ennyi elég is. Teljesen mindegy, hogy a két végpont között mennyit bóklásznak a csomagok. Az se érdekel senkit, hogy odafelé más útvonalon megy a csomag, mint visszafelé. Csak odaérjen.

Azért az megér még egy gondolatfutamot, hogy mi is ez a beazonosítás? Melyik címüket használják a hídfőállások a csatorna kiépítéséhez?
Bármilyen meglepő is, az IPv4 címüket. Tehát amikor manuálisan építünk ki egy IPv6-over-IPv4 csatornát, akkor a két végpont IPv4-es címeit kell megadnunk.
Viszont a csatornán már IPv6 forgalom fog keresztülfolyni, ahol a csatorna egy plusz routingot, egy plusz ugrást jelent. Ez viszont azt jelenti, hogy a csomagok továbbításakor már a hídfőállások IPv6 címei fognak számítani.



4.64. ÁBRA CSATORNAÉPÍTÉS MANUÁLISAN

Száz eszmefuttatásnál is többet ér egy konkrét példa. Van nekünk egy IPv6 alhálózatunk Kazincbarcikán (1a04:ff1:1::/64) és van egy másik New Yorkban (1a04:ff1:2::/64). A két alhálózatot szeretnénk összekötni az interneten keresztül.
Mindkét alhálózatban üzemel egy-egy Windows Server 2008, melyeket routerként használunk. (Tudom, nyakatekert a dolog, de ez alapvetően egy Windows könyv, én pedig nem akarok belemenni router célhardver konfigurálásába. Nem is tudnék.)

Értelemszerűen mindkét szervernek vannak IPv4 és IPv6 címei is - de a jelen példában csak az internet felé forduló IPv4 címük a lényeges.

Első lépésben építsük fel a csatornát Kazincbarcika felől:
```
netsh interface ipv6 add v6v4tunnel ToNewYork 82.145.1.240 135.11.5.1
```

Második lépésben építsük fel a csatornát New York felől:
```
netsh interface ipv6 add v6v4tunnel ToKazincbarcika 135.11.5.1 82.145.1.240
```

Mint látható, a csatorna kiépítéséhez a hídfőállások IPv4 címeit használtuk.
Most már csak meg kellene mondani a routereknek, hogy merre vannak az egyes IPv6 alhálózatok. Ezt megtehetnénk a korábban már említett 'route add' paranccsal is, de a sorminta kedvéért legyen most ez is netsh-val.

Kazincbarcika:
```
netsh interface ipv6 add route 1a04:ff1:2::/64 ToNewYork
```

New York:
```
netsh interface ipv6 add route 1a04:ff1:1::/64 ToKazincbarcika
```

És készen is vagyunk.

Egy gondolatkisérlettel nézzük meg, mit is csináltunk.

New York-ból szeretnék elérni az egyik kazincbarcikai gép fájlmegosztását. Kazincbarcikán csak IPv6only hostok vannak, tehát az IPv4 szóba sem jöhet.
A kazincbarcikai gép IPv6 címét valahogy (DNS, host fájl) megtudjuk. Megpróbálunk rákapcsolódni. Elkészül a SYN csomag, elindul. A default gateway-ként is működő routerbe épp nemrég égettük bele, hogy a kazincbarcikai IPv6 címtartomány (1a04:ff1:1::/64) esetében melyik adapterét használja: ez a ToKazincbarcika adapter. A router szépen becsomagolja IPv4 csomagba az IPv6 csomagot és elküldi az IPv4-es csatornán Kazincbarcikára. Az ottani router lehántolja a külső IPv4 csomagolást, az IPv6 csomagot pedig már el tudja küldeni a helyi gépnek az IPv6only hálózaton.

Jó, most már ismerjük a manuális csatornaépítést. Hogyan néz ki akkor az automatikus?

Első kérdés: milyen entitások között van értelme automatikus csatornaépítésről beszélni? Nyilván nem routerekről van szó, mert azokat a szakik manuálisan konfigurálják. De mi van akkor, ha egy host akar csatornázni egy másik host-tal, vagy

egy routerrel? Elvárható-e Mari nénitől a bérszámfejtésen, hogy kiépítsen egy IPv6-over-IPv4 csatornát az anyavállalatnál üldögélő Auntie Mary gépével?

Költői kérdés volt.

Ilyenkor maga az operációs rendszer építi ki automatikusan a szükséges csatornát. Mik a feltételek?

- Mindkét host IPv4/IPv6 host legyen.
- Ismerjük a távoli host IPv4 címét.

A szituáció:
Az egyik IPv4/IPv6 hostról (Mari néni) el szeretnénk érni egy másik IPv4/IPv6 hostot (Auntie Mary), IPv6 protokollon keresztül. A kettő között viszont ott feszül valahol egy IPv4 only hálózat.

Látható, a csatornát ki tudnánk építeni, hiszen megvannak az IPv4 címek. De nem tudunk semmit a távoli host IPv6 címéről.

Illetve, dehogyisnem.

```
Windows IP Configuration

Host Name                       :hq
Primary Dns Suffix:
        Node Type               : Hybrid
        IP Routing Enabled      : No
        WINS Proxy Enabled      : No

Ethernet adapter Local Area Connection:

Connection-specific DNS Suffix  . :
Description                     Realtek RTL8168B/8111B Family PCI-E Gigabit Ethernet NIC
(NDIS 6.0)
Physical Address                : 00-1E-8C-AB-2F-88
DHCP Enabled                    : Yes
Autoconfiguration Enabled       : Yes
Link-local IPv6 Address         : fe80::c5cc:1c5c:8384:4546%8(Preferred)
IPv4 Address                    : 192.168.1.101(Preferred)
Subnet Mask                     : 255.255.255.0
Lease Obtained                  : 2008. június 3. 19:14:09
Lease Expires                   : 2008. június 4. 19:14:08
Default Gateway                 : 192.168.1.1
DHCP Server                     : 192.168.1.1
DHCPv6 IAID                     : 201334412
DNS Servers                     : 84.2.44.1
                                  84.2.46.1
NetBIOS over Tcpip              Enabled
```

```
Tunnel adapter Local Area Connection* 6:

Connection-specific DNS Suffix  . :
Description                     : Teredo Tunneling Pseudo-Interface
Physical Address                : 02-00-54-55-4E-01
DHCP Enabled                    : No
Autoconfiguration Enabled       : Yes
IPv6 Address                    : 2001:0:d5c7:a2ca:3c2f:2bc6:3f57:fe9a(Preferred)
Link-local IPv6 Address         : fe80::3c2f:2bc6:3f57:fe9a%9(Preferred)
Default Gateway                 : ::
NetBIOS over Tcpip              : Disabled

Tunnel adapter Local Area Connection* 7:

Connection-specific DNS Suffix  . :
Description                     : isatap.{47CE5CAA-223F-4CD4-9A17-D91D7DDC2066}
Physical Address                : 00-00-00-00-00-00-00-E0
DHCP Enabled                    : No
Autoconfiguration Enabled       : Yes
Link-local IPv6 Address         : fe80::5efe:192.168.1.101%10(Preferred)
Default Gateway                 :
DNS Servers                     : 84.2.44.1
                                  84.2.46.1
NetBIOS over Tcpip              : Disabled
```

Ez a számítógépem IP konfigurációja. (ipconfig -all) Megesküszöm bármire, hogy csak egy hálózati kártyám van - mégis a listában három interface látszik.

A pirossal jelöltek ugyanis nem fizikai kártyák. Ezek különböző - automatikus - csatornatipusokhoz (Teredo, Isatap) előkészített interface-k. Közös tulajdonságuk, hogy a hozzájuk tartozó IPv6 cím egy rögzített algoritmus alapján generálódik az IPv4 címükből, azaz a feladó is ki tudja számolni ezeket.

4.2.2.1 Intra-Site Automatic Tunnel Addressing Protocol, ISATAP

RFC 4214

A nevéből is látszik, hogy ez a csatornázás csak site-on belül működik. Bármennyire is szomorú, de Mari néni és Auntie Mary nem ezt a technikát fogják használni az interneten keresztüli csatornaépítéshez. A keletkező IPv6 cím nem lesz globális.

Az algoritmus:

- Mivel link-local cím lesz belőle, így az első 64 bit adott: fe80::/64.
- A node azonosító egy kicsit cifra:
  Ha az IPv4 címem publikus IP cím, akkor ::200:5efe:w.x.y.z.
  Ha az IPv4 címem privát, akkor ::0:5efe:w.x.y.z.
  Ahol w.x.y.z az IPv4 cím oktettjei.

Ellenőrizzük le a saját gépem adataival:

- IPv4 cím: 192.168.1.101
- Isatap cím: fe80::5efe:192.168.1.101

Stimmel.

> Esetleg gondot okozhat a következetlen írásmód.
> Mi az, hogy fe80::5efe:192.168.1.101? Hogyan kerültek hirtelen a címbe pontok? Nem kell kétségbe esni, ezt csak a szemléletesség kedvéért írják így. Akit zavar, az legfeljebb gyorsan átszámolja az IPv4 címét hexadecimálisba (c0a8:165) és így írja fel az Isatap címet: fe80::5efe:c0a8:165.

Ez a legegyszerűbb csatornázás, de a lényeg remekül látszódik. Ha én egy IPv4/IPv6 host vagyok és küldeni akarok egy másik IPv4/IPv6 hostnak egy IPv6 csomagot, miközben az átviteli közegként funkcionáló hálózat csak IPv4-es, akkor a címzett IPv4 címéből tudni fogom az ISATAP csatornájának a végponti címét - és már indulnak is az építőipari vakondok kiépíteni a csatornát. Amint az megvan, mehet a korábban említett IPv6-over-IPv4 Tunneling.

4.2.2.2 6TO4

RFC 3056

Ennél az automatikus csatornatípusnál már unique-global címet kapunk. Azaz elméletileg Mari néni és Auntie Mary már képesek lennének ezt a csatornát használni. De csak elméletileg. A csatornának van ugyanis egy kellemetlen feltétele: nem lehet útközben címfordító (natoló) hálózati eszköz. Más szóval, a hídfőállásoknak publikus IP címekkel kell bírniuk - márpedig ha a két éltes hölgy céges hálózatban dolgozik, akkor ez valószínűleg nem áll.

De azért nézzük meg, hogyan generálódik ennél a tipusnál az automatikus cím.

Mivel unique-global címről van szó, a képzése is hasonló.

- Lesz egy 48 bites prefix: 2002:w.x.y.z
- 16 bit site azonosító
- 64 bit node azonosító.

Mint említettem, ha a hostnak privát IP címe van, akkor nem generálódik 6to4 címe, így az én gépemnek sincs.
Csak a példa kedvéért képzeljük el, hogy dafke generálok magamnak 6to4 címet. Legyen a site azonosítóm: c01d. Ekkor a következő 6to4 címet kapnám: 2002:c0a8:165:c01d: c5cc:1c5c:8384:4546.
(A node azonosítót a link-local címből lehet kiolvasni.)

Gondolom, látszik a bukta: mivel a prefix képzéshez az IP címemet használtuk és az nem publikus cím, így egyáltalán nem garantálja semmi sem, hogy az egyediséget biztosító prefixem világméretben tényleg egyedi is lett. Azaz dafke ide vagy oda, nem tudnám biztonságosan használni.

Ennél a csatornatípusnál is az IPv6-over-IPv4 kapszulázást használjuk.

4.2.2.3 TEREDO

RFC 4380

Itt is unique-local címeket kapunk - és ez a csatorna már áthidalja a címfordításokat is. Igen, ez az, melyet végre becsületben megőszült kolléganőink is tudnak majd használni.

Szakemberként viszont azt kell mondjam, ez a legbonyolultabb eljárás.

Kezdjük ott, hogy nem a megszokott IPv6-over-IPv4 kapszulázást használja. Annak ugyanis volt egy kellemetlen tulajdonsága: az IPv4 csomagban, amikor jelezni akartuk, hogy milyen protokollt is tartalmaz a payload, azt mondtuk, hogy protocol41, azaz IPv6. Ettől viszont a legtöbb tűzfal eldobja az agyát. Meg a csomagot.
A Teredo ravaszabb egy kicsit: IPv4-en belüli UDP csomagba kapszulázza be magát, így már sértetlenül átsuhan a kerberoszok mellett.

Fontos, hogy a Teredo csatornák kiépítéséhez nem elég a két végpont. Szükség van némi segítségre is, mégpedig a megfelelő IPv6 címek kiszámolásához. (Ne felejtsük el, privát címekből képezünk unique-global címeket.) Ezt a segítséget egy ún. Teredo szerver biztosítja a végpontoknak, azaz a Teredo klienseknek. (Emellett a szerver a kommunikáció beindításánál is segédkezik.)
Honnan szerzünk ilyen szervert? Van. Legyen elég ennyi. A Microsoft üzemeltet ilyen szervereket valahol a felhőben, az operációs rendszerei pedig használják a szolgáltatásaikat, anélkül, hogy külön értesítenének róla minket.

Ennyi bevezetés után nézzük, hogyan keletkezik egy Teredo cím.

- Kezdünk egy 32 bites prefix-szel. Ez abszolút konstans, az értéke 2001:0000[70].
- Itt jön a Teredo szerver IPv4 címe, szintén 32 biten.
- 16 bitnyi flag. Ebbe most nem mennék bele.
- Megkutyult külső port. 16 bit.
- Megkutyult külső cím. 32 bit.

[70] Látod, ugye, hogy global unique?

Hát, izé. Ebből egyedül a konstans érthető elsőre.
Viszont van egy élő Teredo címünk, ott vigyorog a pár oldallal korábbi listában:
2001:0:d5c7:a2ca:3c2f:2bc6:3f57:fe9a

Analizáljuk szénné.

Az első 32 bit oké, az a konstans.

A második 32 bit már izgalmasabb. D5c7:a2ca, visszaszámolva decimálisba: 213.199.162.202. Ő az anyuka kis segítője, a Microsoft háttérben serénykedő Teredo szervere[71].

A következő blokk értéke 3c2f, ez flag formára alakítva: 0011110000101111. Ha nem haragszol, ezt nem részletezem. (Az RFC-ben szépen le van írva.)

Aztán megyünk bele a málnásba. A külső port azt az értéket jelenti, amelyre a NAT-nak fordítania kell az UDP kommunikációt. Ezt az értéket a Teredo szerver adja. A megkutyulás pedig azt jelenti, hogy a tényleges értéket lekizáróvagyozzák (XOR[72]) a 0xFFFF értékkel. Ezzel bizonyos NAT implementációkat vágnak át. Jelen esetben az érték 2bc6, ezt visszaxorozzuk, akkor azt kapom decimálisban, hogy 32686.

A külső 32 bites cím pedig egy publikus IP címet fog adni. Ezzel az IP címmel - és az előzőekben kiszámolt port értékkel - fogok megjelenni a natoló linksys routerem külső felén. A kutyulás elve ugyanaz, csak mivel most 32 bites számot xorozunk, így a konstans a 0xFFFFFFFF szám lesz. Számoljuk ezt is vissza: 3f57fe9a xor FFFFFFFF, az annyi mint C0A80165, ez IPv4 formátumba alakítva 192.168.1.101.

Hoppá. Itt valami árulás történt. Külső címként ugyanazt a címet kaptam vissza, mint ami a belső, privát IP címem - pedig itt egy publikus IP címet kellett volna találnom.

Leellenőriztem.

---

[71] Nem fix IP cím, változik. Hivatalosan a teredo.ipv6.microsoft.com címen található az aktuális Teredo szerver.

[72] A kizáró vagy (XOR) igazságtáblája:

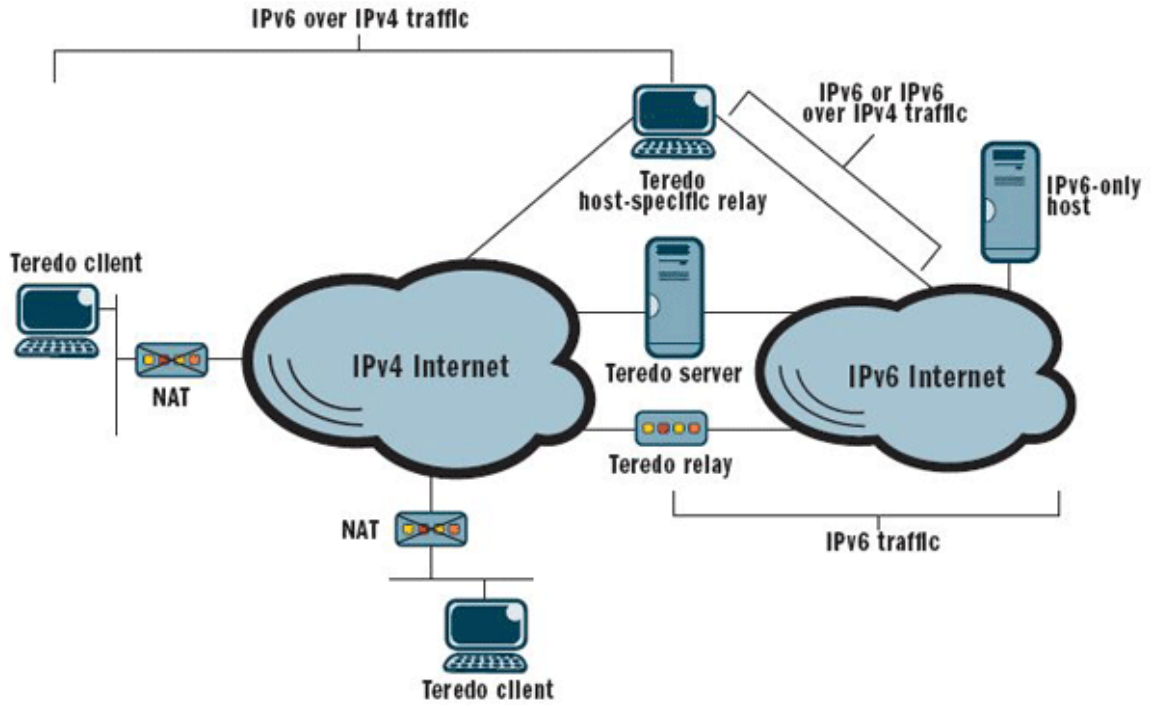
0	0	->	0
0	1	->	1
1	0	->	1
1	1	->	**0** (Ez miatt kizáró.)

A műveletnek van egy érdekes trükkje: ha tetszőleges számot xorolunk egy számmal, majd a művelet eredményét xoroljuk ugyanazzal a számmal, akkor az eredeti számot kapjuk vissza. Érthetőbben: (A xor B) xor B = A.

Elindítottam a Wireshark programot, majd menetközben megváltoztattam az IP címemet. Nos, a Vista többször is lekérdezte a teredo.ipv6.microsoft.com címet - de semmit nem kommunikált vele. Ebből arra következtettem, hogy a fenti értékeknek egyelőre nincs jelentősége. Amikor először próbálok majd meg Teredo csatornát építeni valahová, valószínűleg akkor kapom csak meg az igazi port, illetve IP cím értékeket.

Mivel az IPv4-IPv6 konverziók közül messze ez a leghasználhatóbb, sőt, mi több, egy átlagos környezetben ez az egy lesz használható, érdemes a Teredo csatornázásban jobban elmélyülni.

Kezdjük a sors fintorával. A korábbi kiadásokban azt írtam, hogy a Teredo szerver semmilyen Microsoft terméknek nem lesz része.
Aztán kijött a Windows Server R2.
Amelyikben már van Teredo szerver és Teredo Relay is. (Teredo kliens már a Vistában is volt.)



4.65. ÁBRA TEREDO RENDSZER ELEMEI

Az ábráról leolvasható, hogy a NAT mögötti Teredo kliens vagy egy Teredo relay-n, vagy egy Teredo host-specific relay-n keresztül tud kommunikálni a túloldali Teredo klienssel, illetve IPv6 node-dal. A Teredo szerver a csatorna kiépítéséhez kell (ez oszt IPv6 címet és indítja be a kommunikációt.) A Windows Sever R2 előtt mind a

Teredo szervert, mind a Teredo relay-t a Microsoft adta (valahol a felhőben), illetve megadta a lehetőséget az ISP-knek saját Teredo eszközök telepítésére.
A Windows Server R2 megjelenésével annyi változott, hogy mi is üzemeltethetünk Teredo szervert és Teredo relay-t saját magunknak. A Windows Teredo kliensek helyből ugyan a teredo.ipv6.microsoft.com címre vannak bedrótozva, de ezt a netsh segédprogrammal tudjuk módosítani. (Illetve csoportházirendből.)

Cable Guy
http://technet.microsoft.com/en-gb/magazine/2009.07.cableguy.aspx#id0270013

5 A SZÁLLÍTÁSI RÉTEG PROTOKOLLJAI

Megvagyunk a címzésekkel és az útjelző táblákkal. Épp itt az ideje, hogy a szekérrel is foglalkozzunk.
De melyikkel? Hiszen kettő közül is választhatunk: van egy Trabantunk és egy Volvónk.

5.1 USER DATAGRAM PROTOCOL - UDP

RFC 768

Ő a Trabant. Hogy miért?

- **Kapcsolatmentes.** A feladó és a címzett nem beszélgetnek: a feladó elküldi a csomagot, a címzett vagy megkapja, vagy nem. De semmilyen formában sem jelez vissza.
- **Megbízhatatlan.** A csomagokban nincs semmi sorszámozás, nincs semmi elveszés elleni védelem. UDP esetében minden ilyesmit arra az alkalmazásra bíznak, amelyik küldi/fogadja a csomagokat. (CRC ellenőrzés ugyan van, de az csak szótlan csomageldobáshoz vezet.)
- **Nem tárol.** Ahogy beérkezik egy UDP csomag, az rögtön megy is tovább az alkalmazás réteg felé. Ha az nem veszi át, akkor úgy járt.
- **Nem tördel.** Tördeljen helyette más. (Ezt mondjuk megteszi az IP réteg.)
- **Nincs benne folyamatszabályozás**. Azaz nem tud se gyorsítani, se lassítani.
- **Hagyományos**. A megszokott dobozolós technológiát követi.

Szóval, fapad. Cserébe egyszerű, mint a tégla, nem rakódik óriási ~~vízfej~~ header a csomagokra és a kisérő kommunikáció hiánya miatt gyors is.

> Hogyan is írták anno a Trabant kezelési kézikönyvében?
>
> *"A Trabant útfekvése kitűnő, és gyorsulása kifogástalan.*
> *Ez azonban nem szabad, hogy könnyelműségre csábítson."*[73]

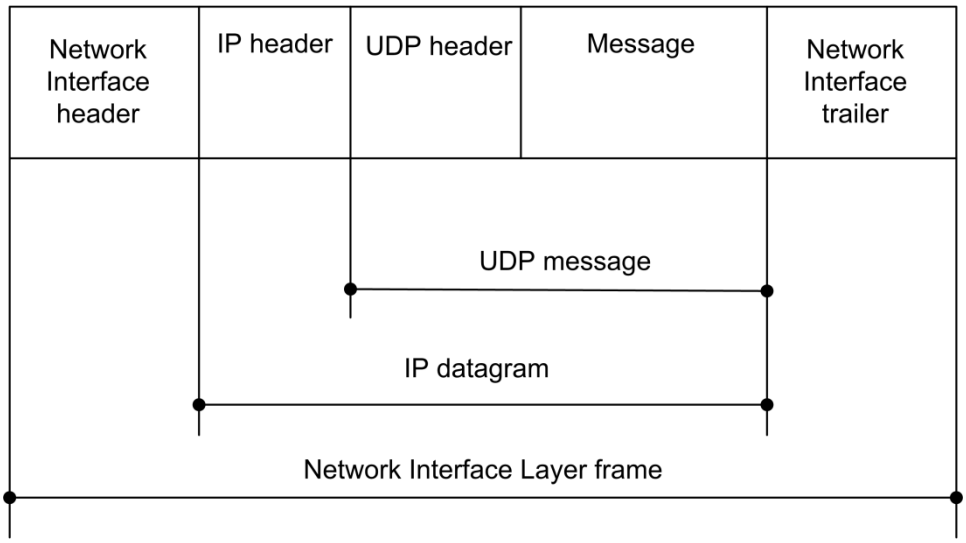
[73] Az irodalomba átemelve Esterházy Péter által.

A teljesség kedvéért említsünk meg néhány példát a használatára.

- A DNS nagyon jó példa, ugyanis az vegyesen használ UDP-t, illetve TCP-t. Ha elfér az üzenet egy UDP csomagban - tipikusan névfeloldási kérés, illetve válasz - akkor az UDP lesz a befutó. Ha több csomagra lesz szükség - tipikusan zónatranszfer - akkor viszont a TCP.
- Van, amikor maga az alkalmazás szintű protokoll foglalkozik a csomagok szállításának megbízhatóságával. Ilyen pl. a TFTP.
- Van, amikor a feladó periodikusan küldi el az információkat. Ilyenkor nem gond, ha az egyik küldemény nem érkezik meg, hiszen hamarosan megy a másik. Ezt tudja a korábban már említett RIP.
- Végül a multicast. A TCP csak egy-egy tipusú kapcsolatot tud kezelni.

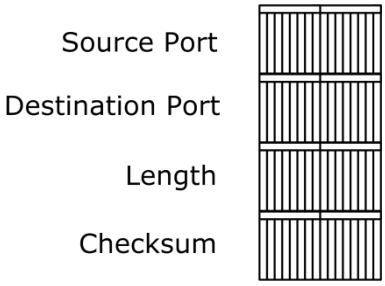
Az UDP protokoll:
http://en.wikipedia.org/wiki/User_Datagram_Protocol

Aki figyelmesen olvasta eddig a könyvet, nyilván nem fog meglepődni a következő ábrán.



5.1. ÁBRA AZ UDP ÜZENET HELYE A CSOMAGBAN

Szépen láthatóak az egyes rétegek, látható, hogy ami az egyik rétegnek payload, az a másikon belül tovább oszlik header+payload darabokra - és így tovább.



5.2. ÁBRA AZ UDP HEADER

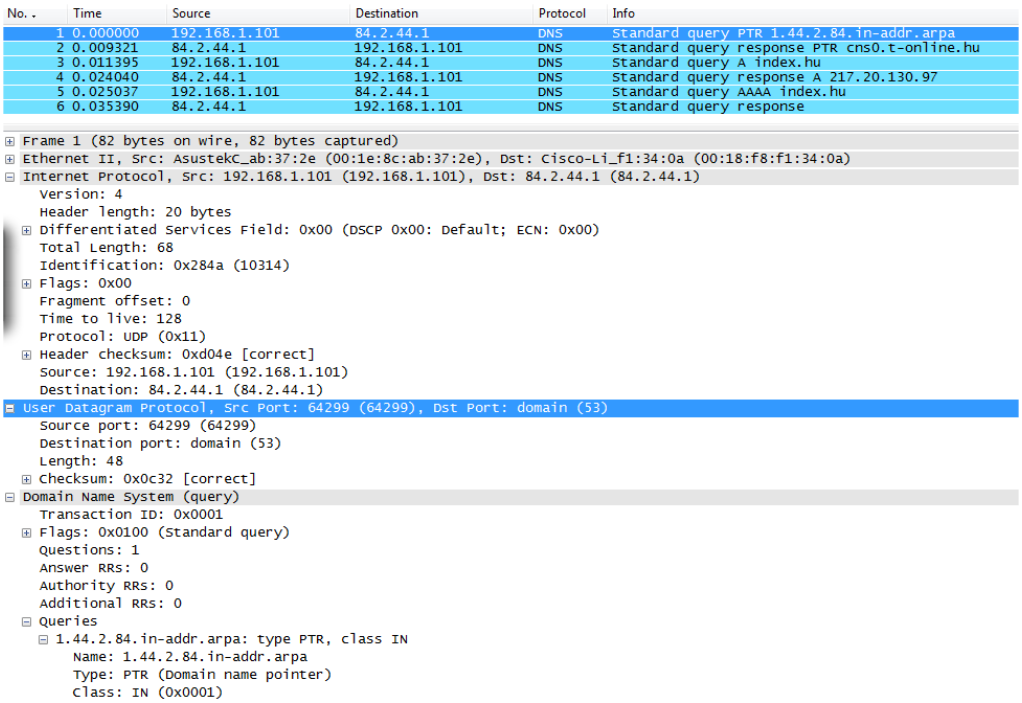
SOURCE PORT: Azt a portot azonosítja, amelyiken az alkalmazás elküldte a csomagot.

DESTINATION PORT: Azt a portot tartalmazza, amelyiken a fogadó oldali alkalmazás fogadja a csomagot.

LENGTH: Az UDP csomag mérete. Minimum 8 bájt (ekkor csak a header van), maximum... na, mennyi? Mivel a LENGTH 16 bites, így maximum 65535 bájt lehet.

CHECKSUM: A jó öreg CRC.

Nézzük meg mindezt a gyakorlatban is.
Az *nslookup index.hu* parancs hatására a következő történt.



5.3. ÁBRA UDP CSOMAG FELÉPÍTÉSE

Szépen látszik az UDP header és alatta az UDP message is.

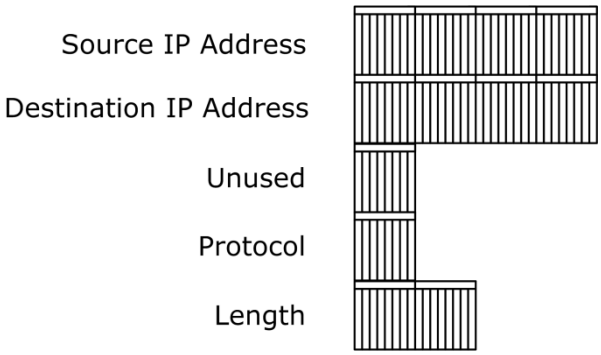
Ez egy szép, tiszta protokoll, az égegyadta világon semmi kavarás nincs vele.

Vagy mégis?

Ekkor futottam bele az UDP Pseudo Header fogalomba.

Nem tréfálok, egy csomó időt foglalkoztam vele, áttúrtam a fél internetet. Mi az, hogy pszeudó? Most akkor van? Ha igen, akkor hol? Az UDP üzenetben? De hiszen ott nincs. Az IP csomagban? Ott sem volt. De ha nem létezik, akkor meg miért emlegetik? Nos, hosszas kutatások után meglett a végső megoldás.
Az UDP Pseudo Header létezik is... meg nem is.

Jó, mi?



5.4. ÁBRA UDP PSEUDO HEADER

Erről a szerkezetről van szó. Vizsgáld meg alaposan a korábbi ábrát (*5.3. ábra UDP csomag felépítése*) Látsz valahol a capture fájlban ilyen szerkezetet? Ugye nem.
Pedig ott van.
Csak darabokban. Ott van ugyanis a Source IP Address, a Destination IP Address és a Protocol az IP fejlécben, a Length pedig az UDP fejlécben. (Az Unused bájt csak padding célt szolgál.)
Nos, ez a szerencsétlenség azért pszeudó, mert külön nem utazik a csomagban, hanem a csomag *egyéb mezőiből rakja össze a küldő/fogadó host*.

Jogos a kérdés, hogy mi szükség lehet rá?

Ez a kapocs. Ez fogja össze az IP headert és az UDP üzenetet. Méghozzá úgy, hogy a korábban nagyvonalúan elintézett CHECKSUM mezőnél nem árultam el, mely mezők is vesznek részt az ellenőrző összeg generálásában. Nos, egész konkrétan az UDP Pseudo Header és a teljes UDP csomag, beleértve a fejlécet és az üzenetet. (Logikusan, ahogy változnak az IP csomag adatai - pl. a Destination IP Address - vagy natolás miatt az UDP portszámok, a CHECKSUM mindig újraszámolódik.)

Felmerülhet még egy kérdés: mi van a multicastnál? Akkor hogyan is kezeljük ezt a CHECKSUM mezőt? A válasz az, hogy UDP esetében a mező használata opcionális, ha zéró az értéke, akkor senki nem foglalkozik vele.

Meg még egy kérdés: és az IPv6? A válasz az, hogy igen. Nyilván be fog játszani: egyfelől a pszeudo header változni fog (nemcsak a nagyobb IP címek miatt, de a mezők sem lesznek ugyanazok), másfelől IPv6-ban már kötelező a CHECKSUM mező használata.

The Checksum field and the UDP Pseudo Header:
http://www.tcpipguide.com/free/t_UDPMessageFormat-2.htm

5.2 TRANSMISSION CONTROL PROTOCOL - TCP

Hé, ki ájul el,
ha meglát egy Volgát?
Hé, ki látta már
a Zaporozsec szobrát?

Ki az az őrült, ki Trabantra vágyik,
talán csak egy múzeum.
Kerüljön inkább valamivel többe,
manapság ez a minimum.

Volvo!
Volvo!
Volvo!

A KFT szösszenetétől eltérően azért van értelme a Trabantnak (illetve a manapság helyette használt fapados járgányoknak) és persze megvan a funkciója a Volvónak is. Csak éppen az utóbbinak meg is kell fizetni az árát.

RFC 793

Nézzük, mit is tud a luxuskategória?

- **Kapcsolatcentrikus.** No, nem úgy, mint egy iwiw-függő plázacica, hanem úgy, hogy mielőtt bármekkora darabot is elküldene a rábízott adathalmazból, kiépít egy kapcsolatot a küldő és a fogadó között, majd küldés közben ápolja is ezt a kapcsolatot.
- **Önmegtartóztató.** Mind küldő, mind fogadó oldalon képes a küldési sebesség szabályozására.
- **Kétirányú.** Ugyanazon a kapcsolaton egyszerre mehet mindkét irányú forgalom.
- **Megbízható.** Az adatfolyam darabkái számozottak és mindegyiket leltár szerint át kell vennie a fogadónak. Ha ez nem történik meg, akkor a feladó újra elküldi.
- **Törhetetlen.** A TCP réteg képes kommunikálni a Network Interface réteggel, így az MTU ismeretében akkora adagokat kér az alkalmazási réteg adataiból, hogy azokat később IP szinten ne kelljen tördelni.
- **Monogám**. A kapcsolatot csak két host között tudja kiépíteni.
- **Folyamatos**. Az átvitel adatfolyam (stream) jellegű. Ezért mondjuk azt, hogy a TCP payload az szegmens és nem datagram. (Ettől függetlenül persze a stream darabjai csomagokban - IP datagram - utaznak.)

Cserébe egy kicsit bürokratikus. Na jó, nem kicsit. Ez a rengeteg oda-vissza szóló vezérlőinformáció, a kapcsolat menedzselése nem megy ingyen - sávszélességgel, processzoridővel, memóriával fizetünk érte. Értelemszerűen akkor célszerű használni, ha kiemelten fontos a megbízható adattovábbítás.

Most egy kicsit elkalandozok a megszokott tárgyalási módtól. Egy vallomás következik.

Miközben írtam ezt a könyvet, meglehetősen lineárisan haladtam a tanulás/megemésztés/megírás vonalon. Nekifut, dobbant, elugrik.
Ez a linearitás itt, ennél a résznél torpant meg.

A parszerem nem bírt megbírkózni a TCP stream jellegével. A jó ég áldja meg, hiszen eddig itt mindent dobozoltunk. Hogyan mondhatjuk ki egy alapvetően diszkrét doboz tartalmára, hogy az folyamatos adatáram?

Napokig olvasgattam, túrtam a netet.

Aztán hagytam a fenébe az egészet, hirtelen közbejött egy csomó egyéb probléma.
Három nap múlva ültem le megint a kézirathoz - és nem értettem, mit nem értettem korábban. Az agy egy hihetetlenül érdekes szerkezet.

Példa. Egyszerűen úgy kell elképzelni a dolgot, hogy mondjuk szép lassan hömpölyög a Duna. Igenám, csakhogy a Vaskapunál üzembeállítottak egy gátat, arra szereltek egy futószalagot, mely csak dobozokat képes szállítani. A futószalag elején a szakik villámgyorsan bedobozolják a vizet, ráfirkantanak egy azonosító számot és már mehet is a szalagra. A túloldalon pedig sorbarendezik a dobozokat a számok alapján, kiborítják belőlük a vizet, az üres dobozra ráírják, hogy a csomag rendben megérkezett, majd visszaküldik a futószalag elejére.
A Duna a futószalag előtt a feladó alkalmazás, a Duna a futószalag után a fogadó alkalmazás. A futószalag pedig maga a hálózati infrastruktúra.

Jó-jó, de miért tennének ilyen őrültséget a román vízügyes szakik?

Nos, innentől sántít a hasonlat. Javaslom, térjünk is vissza az informatika világába.

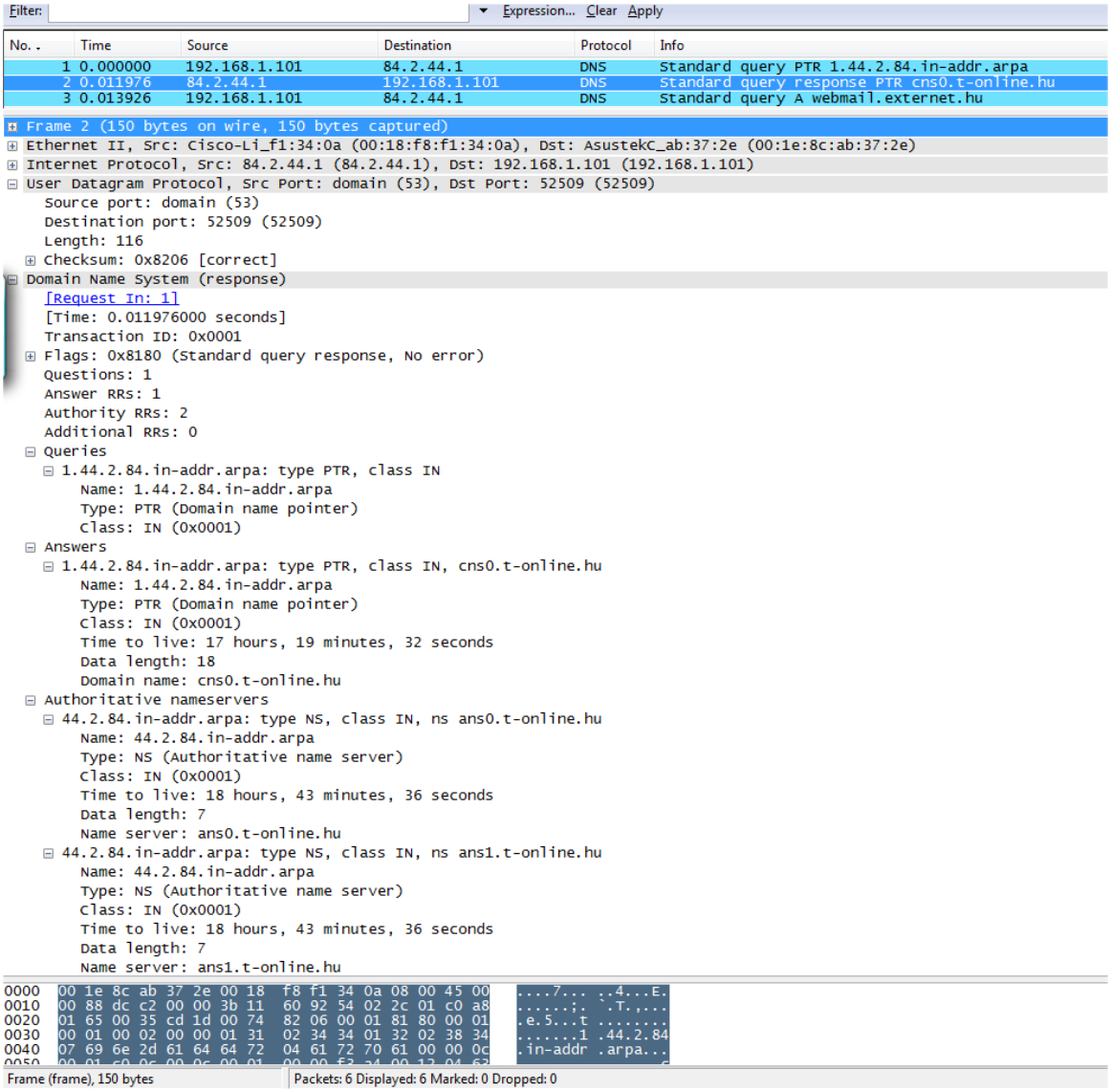
Itt a kérdés máshogyan hangzik el: kérem szépen, hogyan is lehetne másképp?

Gondoljunk bele: eddig minden réteg ismert, jól definiált rétegtől vette át az információt tartalmazó dobozt. Szabvány szerint tudtuk, hogy az IP csomagban a payload csak ICMP/IGMP, illetve a Transport rétegtől származó csomag lehet. Más nem. Mintha minden tipusnak lenne egy más beosztású fakkos doboza.
De a szegény, szerencsétlen TCP-nek már az alkalmazástól kell elvennie az információt. Azt kellene szépen fakkokra osztott dobozba raknia. Tudnánk annyi fajta fakkos dobozt használni, ahány féle alkalmazás van? Még akkor sem, ha csak kommunikációs szabványokra épülő alkalmazásaink lennének. Mihez kezdenénk akkor egy egyedi alkalmazással, mely mondjuk az 1964-es porton a saját egyedi protokollján keresztül kommunikál? Reménytelen. Egész egyszerűen nincs értelme annak, hogy a TCP protokoll nekiálljon parszolni a beléje érkező információt. Egyszerűen fog egy üres dobozt, beleszórja az érkező folyam bitjeit és továbblöki azt az IP rétegnek. Majd parszol a túloldalon lévő alkalmazás - feltéve, hogy megkapott minden dobozt.

Oké. De mi van akkor az UDP-vel? Az is szállítási protokoll... és mégsem stream. Hogyan tudja megugrani az a cseles UDP ezt az akadályt?
Úgy, hogy az UDP üzenet belefér egy csomagba. Egyszerűen muszáj[74][75] neki, mivel UDP-n belül nincs sorszámozás, nincs visszajelzési mechanizmus. Természetesen az UDP sem fogja parszolni a megkapott üzenetet. (Szépen is néznénk ki, ha mondjuk a DNS Name Query Response megváltoztatása magával vonná az UDP szabvány megváltoztatását is.)

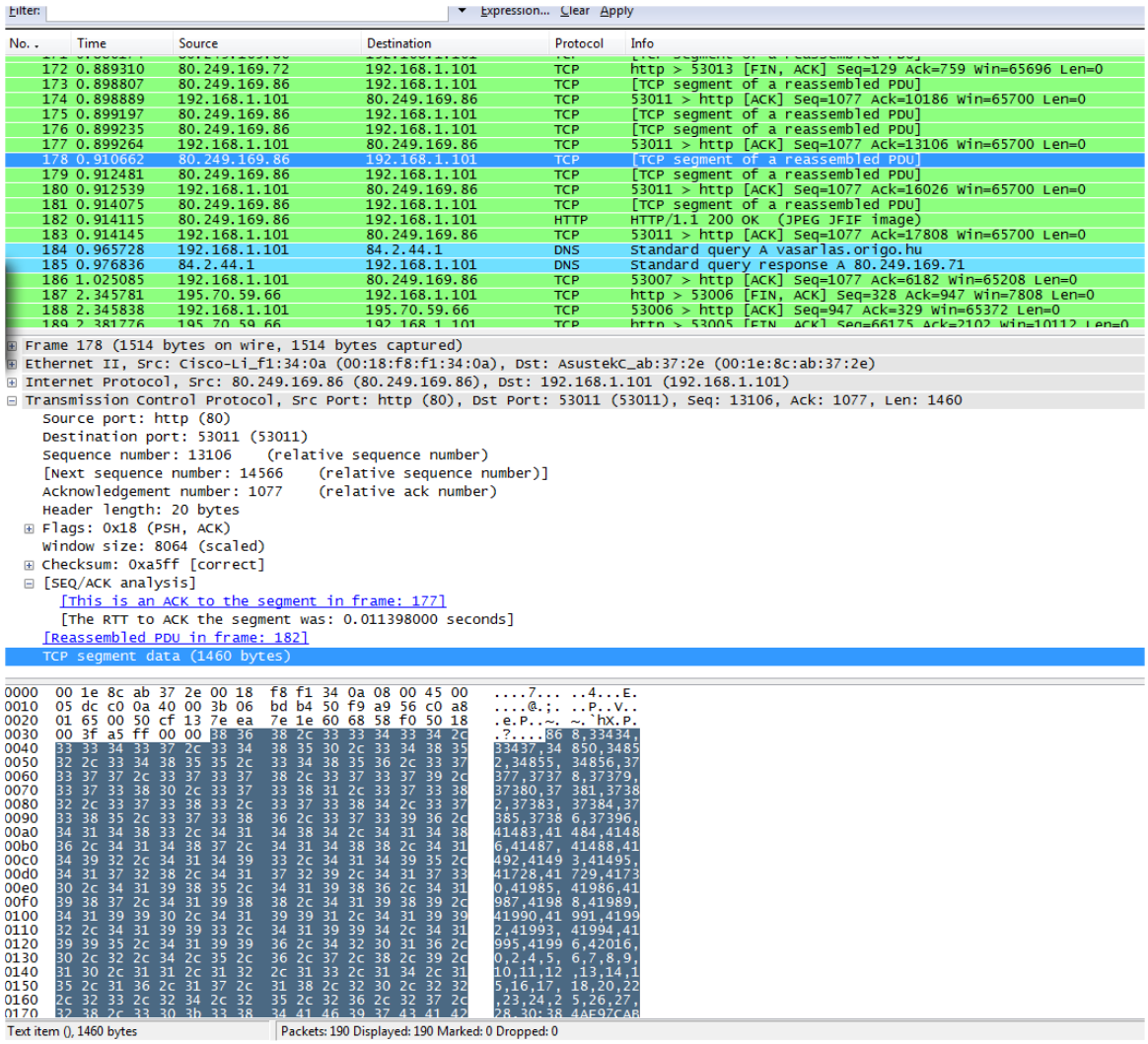
Nézzük meg ezt két ábrán.

[74] muss sein
[75] Eddig bírtam féken tartani a népművelőt.



5.5. ÁBRA EGY UDP CSOMAG

Az UDP csomag teljesen ki van bontva. Látható, hogy a csomagban lévő információ teljes, a monitorozó szoftver parszere képes volt értelmezni.
Úgy is mondhatnám, hogy az UDP tulajdonképpen egy annyira rövid stream, hogy belefér egy csomagba. Ezért diszkrét.



5.6. ÁBRA EGY TCP CSOMAG

Itt viszont megfeküdt a Wireshark parszer. Ugyanis a TCP szegmensben egy adatfolyam egy darabja utazott, melyet önállóan képtelen volt értelmezni. Látható, hogy csak annyit tudott rámondani, hogy ez egy TCP szegmensnek az adata. Alul, a Packet Bytes részen pedig látjuk, hogy ez valami jó nagy zagyvaság. Szerencsére nem is nekünk szól, hanem a túloldali alkalmazásnak.

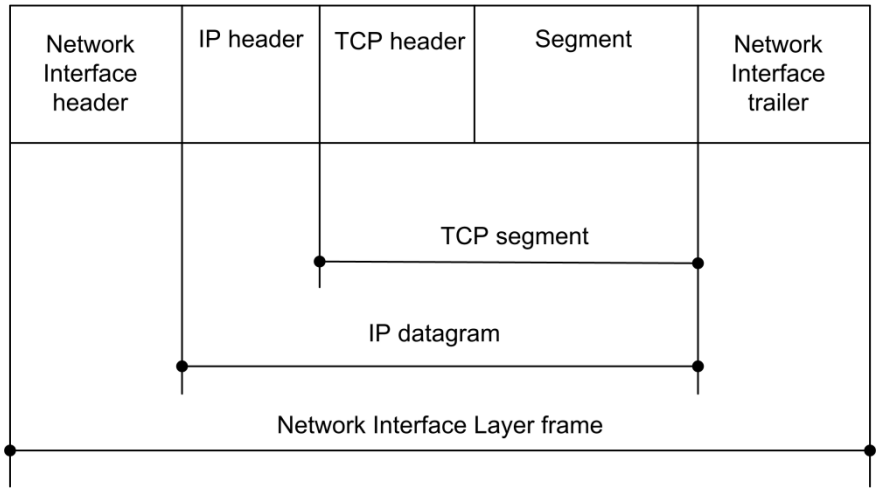
TCP:
http://en.wikipedia.org/wiki/Transmission_Control_Protocol

Végül álljon itt egy összefoglaló táblázat a kétfajta szállítási protokollról:

5.1. TÁBLÁZAT

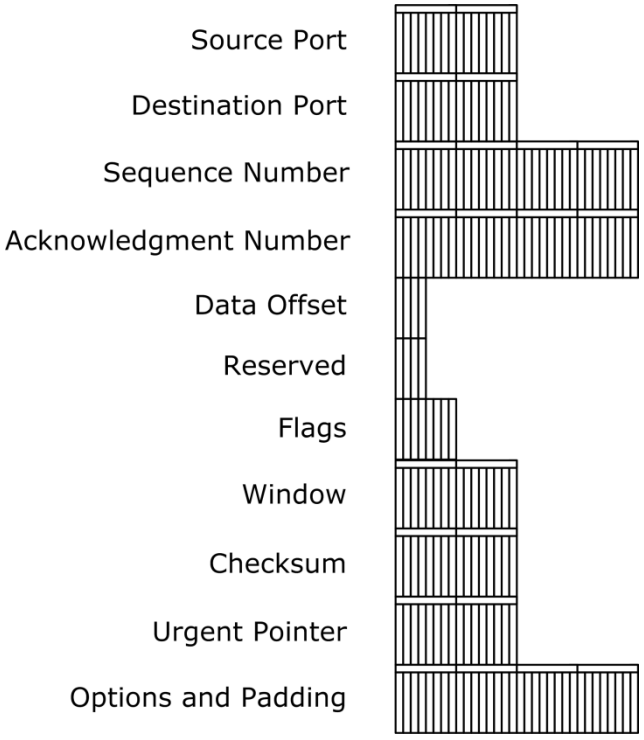
	UDP	TCP
Kapcsolat	**Kapcsolatmentes.** A feladó elküldi a csomagot, a címzett megkapja. Ezen a tranzakción kívül semmit nem beszélgetnek.	**Kapcsolatcentrikus.** Még a tényleges adatküldés előtt a felek kiépítenek egy kapcsolatot és azt ápolják is.
Megbízhatóság	**Megbízhatatlan.** A küldő/fogadó alkalmazásokra 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.

Oké, térjünk vissza a megszokott, bitbúvár mederbe.
Nézzük meg, hol is helyezkedik el a TCP a nagy csomagon belül.



5.7. ÁBRA A TCP SZEGMENS ELHELYEZKEDÉSE A CSOMAGBAN

És akkor a TCP fejléc.



5.8. ÁBRA A TCP FEJLÉC

SOURCE PORT: Az a port, amelyikről a küldő alkalmazás küldi a csomagot.

DESTINATION PORT: Az a port, amelyiken a fogadó alkalmazás fogadja a csomagot.

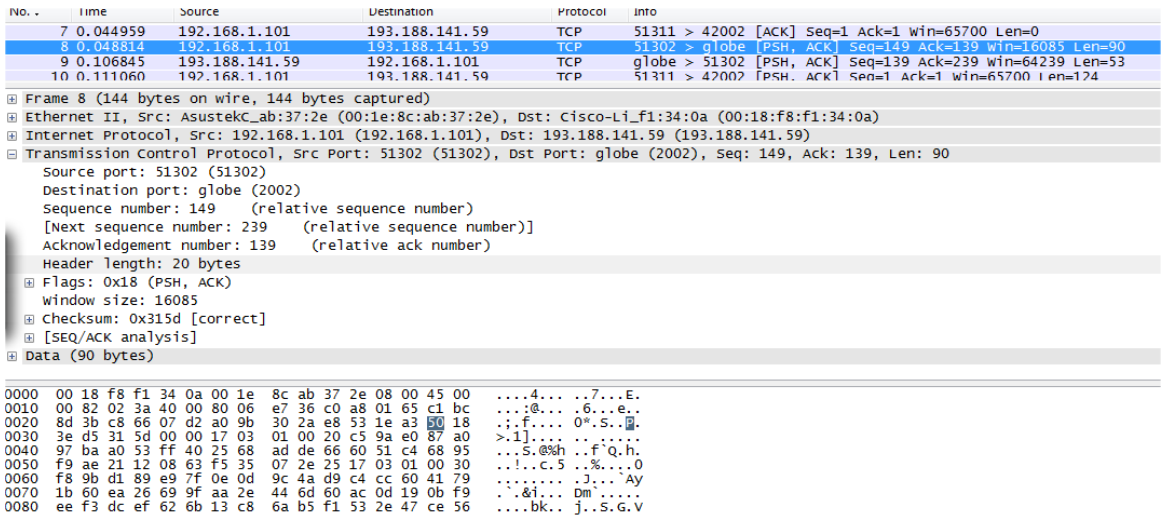
> Megjegyzés:
> A korrekt cím, melynek minden eleme meg kell legyen ahhoz, hogy a csomag a megfelelő helyen landoljon, az az IP cím és a portszám - együtt. Ez egy annyira fontos elem, hogy külön neve is van: socket. Dacára annak, hogy az IP cím az IP headerben található, a portszám pedig a TCP/UDP headerben. Természetesen külön beszélünk source, illetve destination socket-ről.

SEQUENCE NUMBER: 4 bájt, azonosító szám. Mint a neve is mutatja, ez alapján fogja a fogadó sorbarendezni a csomagokat. Ha a később bemutatandó SYN flag értéke magas, akkor az inicializáló csomagról beszélünk, az ehhez tartozó sorszámot pedig INITIAL SEQUENCE NUMBER -nek nevezzük. (ISN)

ACKNOWLEDGEMENT NUMBER: Szintén 4 bájt. A neve alapján ez szolgálna a csomag megérkezésének azonosítására. De később látni fogjuk, hogy a sorbarendezés/visszajelzés funkciókat a két szám közösen, együttműködve valósítja meg.

DATA OFFSET: A névadó csoport megint nem volt csúcsformában. Ez ugyanis gyakorlatilag a TCP header méretét tartalmazza[76]. Persze nem ilyen egyszerűen, hogy odaírjuk, oszt annyi: az IP csomagnál megszokott módon ide a szavak számát írják.

[76] Eddig az ilyen adatokat valamilyen LENGTH névvel illették. A TCP-s fiúk szakítottak ezzel a szemlélettel, ők azt mondták, hogy ez a szám azt jelenti, hogy honnan kezdődik a DATA. Így pontosabb név rá az OFFSET. A lelkük rajta. Az eredménye a két megközelítésnek mindenesetre ugyanaz.



5.9. ÁBRA DATA OFFSET

Ezen egyszer úgyis túl kell esnünk: nézzük meg, konkrétan hogyan is számítódik ki ez az érték. A Wireshark szolgálatkészen már a végeredményt mutatja, azt, hogy a header hossza 20 bájt. De mi van legalul? Amikor ráállunk a HEADER LENGTH sorra (melyről persze tudjuk, hogy ez ~~az igazi Trebitsch~~ a DATA OFFSET), akkor azt látjuk, hogy 50. Izé. Ezmiez?

Mivel a Packet Bytes résznél látjuk, így nyilván egy hexadecimális érték. Decimálisra átszámolva 80 - melynek megint nincs túl sok értelme. Nem is lehet, hiszen csak mi, elkorcsosult tízujjú emberek számolunk tizes számrendszerben. Váltsuk át a számot binárisba: 01010000. Ha megnézzük az ábrát (*5.8. ábra A TCP fejléc*), láthatjuk, hogy a bájt alsó fele a RESERVED érték és kötelezően nulla. Marad a 0101 érték, mely egész pontosan annyit tesz emberül, hogy 5. (Meg hexában is.) Tehát ez a szám van letárolva a mezőben. Hogyan lesz ebből 20? Úgy, hogy mivel ez a szavak számát jelenti, és egy szó az 32 bit, így a végeredmény 5*32, azaz 160 bit, azaz 20 bájt.

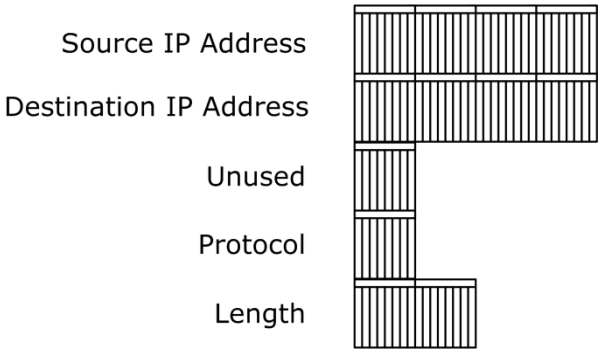
RESERVED: Mint korábban is kiderült, későbbi célokra fenntartott mező, értéke nulla.

RFC 793, 3168

FLAGS: 8 kicsi zászlócska. Szokták ezeket control biteknek is nevezni. A TCP kapcsolat kiépítésében és kezelésében játszanak fontos szerepet. Ott is lesznek részletesen bemutatva.

WINDOW: Ez tulajdonképpen egy adatméret. Azt mutatja meg, hogy a fogadó mekkora adatkupacot képes egyszerre elfogadni a következő csomagban. Az értéke nem állandó, a fogadó a terheléstől függően változtatja. Ha például 0, akkor a fogadó egyelőre bezárta az ajtót. Majd szól - küld egy csomagot, nullától eltérő WINDOW értékkel - amint újra kinyitott.

CHECKSUM: Ellenőrző érték, ahol a kiszámítás alapja a TCP header és a szegmens. Megsúgom, itt is használni fogunk pszeudo headert:



5.10. ÁBRA TCP PSEUDO HEADER

Ránézésre teljesen olyan, mint az UDP pseudo header (*5.4. ábra UDP Pseudo Header*), az egyetlen különbség, hogy itt a PROTOCOL mező értéke 6 (TCP), ott pedig 17 (UDP). A szerepe is hasonló: összeköti az IP headert és a TCP headert, úgy, hogy a pszeudo headert is hozzácsapja a TCP header és a szegmens mellé, amikor a CRC-t kiszámolja.

URGENT POINTER: Egy mutató, mely megmutatja, hol van a szegmensen belül rendkívüli sürgősséggel szállítandó adat.
Nem tudod elképzelni, mi? Először én se tudtam. Most akkor mi van? A sürgősséggel szállítandó adat kitör a dobozból és beelőzi a csomagot?

Majdnem, majdnem.

Léteznek olyan információk, melyek az adott - alkalmazás szintű - protokoll vezérléséhez szükségesek. Ezek az információk - hiába az adatok között utaznak - de kiemelten fontosak az alkalmazás szempontjából. Ezeket a vezérlő információkat szokták out-of-band adatoknak is nevezni. Ilyesmit kétféleképpen lehet összehozni:

- Külön TCP csatorna. Ezt az utat választotta az FTP: van egy adatcsatornája, ahol hosszú tömött sorokban hömpölyögnek a bitek és van egy kontrollcsatornája, ahol a vezérlő információk utaznak.
- Egy TCP csatornán utazik minden. Ekkor a vezérlő információkat az Urgent Pointer hasítja ki a szegmensből. Természetesen az adat nem fogja beelőzni a csomagot, de amint beérkezik a TCP pufferbe, egyből feldolgozásra is kerül. Így működik a Telnet.

RFC 793, 1122

A kihasítást pedig úgy kell elképzelni, hogy a vezérlőinformációk mindig a szegmens elején utaznak, a pointer pedig oda mutat, ahol a vezérlőinformációknak vége van. Vagy ahol a nem sürgős adatok kezdődnek. Itt ugyanis történt egy tragikus félreértés: az egyik RFC az egyiket írta elő, a másik a másikat. Átjárás nincs: habár a különbség mindösszesen egy bájt, de ez épp elég ahhoz, hogy az egyik implementáció ne ismerje fel a másik által küldött információt.

A vezérlőinformációk kezeléséhez az URGENT POINTER önmagában nem elég, a csomagban magasra kell állítani az URG flag-et is. Ez jelzi azt, hogy komolyan gondoltuk a pointert.

OPTIONS: Az IP OPTIONS mezőhöz hasonlóan ez egy olyan mező, ahová rengeteg addicionális funkciót építettek be. Ezekkel is találkozni fogunk később.

És most vagyok megint zavarban. Itt volt a második nagy megbicsaklás az anyag feldolgozásában. Egyszerűen képtelen voltam megérteni a Sequence Number (SN) és az Acknowledgement Number (ACK) értékek közötti összjátékot. Egész pontosan nem találtam sem az alanyt, sem az állítmányt. Nem tudtam, hogy ami történik, azért történik-e, mert a node-oknak szabad akaratuk van, vagy inkább egymásra kényszerítenek dolgokat? Ha az utóbbi, akkor ki kire? És mit?

Nyilván van rengeteg írás a neten. Egyik homályosabb, mint a másik. Rendszeresen visszatérő fordulat volt, hogy az ACK egy olyan érték, melyre a node számít. Könyörgöm, mi az, hogy számít? Elmegy a jósnőhöz?

Nyilván meg tudom sniffelni a saját gépem forgalmát és megpróbálhatok valamit kiokoskodni a capture fájlból. De itt a Wireshark sajnos inkább hátráltatott: roppant szolgálatkészen jóval többet mutatott, mint amennyit értelmezni tudtam. Ez egy profinak nyilván jó - de annak, aki ebből akarja megérteni a működést, felettébb zavaró.

Végül egy hekker oldalon találtam egy kellően mély, de azért szájbarágó leírást- és sikerült megértenem a szisztémát. Viszont belinkelni nem fogom. (-;

Mondjuk ez sem rossz:
http://www.firewall.cx/tcp-analysis-section-2.php

De még ekkor is maradt egy megoldandó probléma. A következő nagyobb téma a TCP opciók bemutatása lenne - de azoknak úgy nekiszaladni, hogy nem értjük az SN/ACK pingpongmeccset, nem érdemes. Viszont az SN/ACK rendszert meg a TCP

kapcsolatok boncolgatásánál lehetne rendesen bemutatni - csakhogy ahhoz meg kell a TCP opciók ismerete.

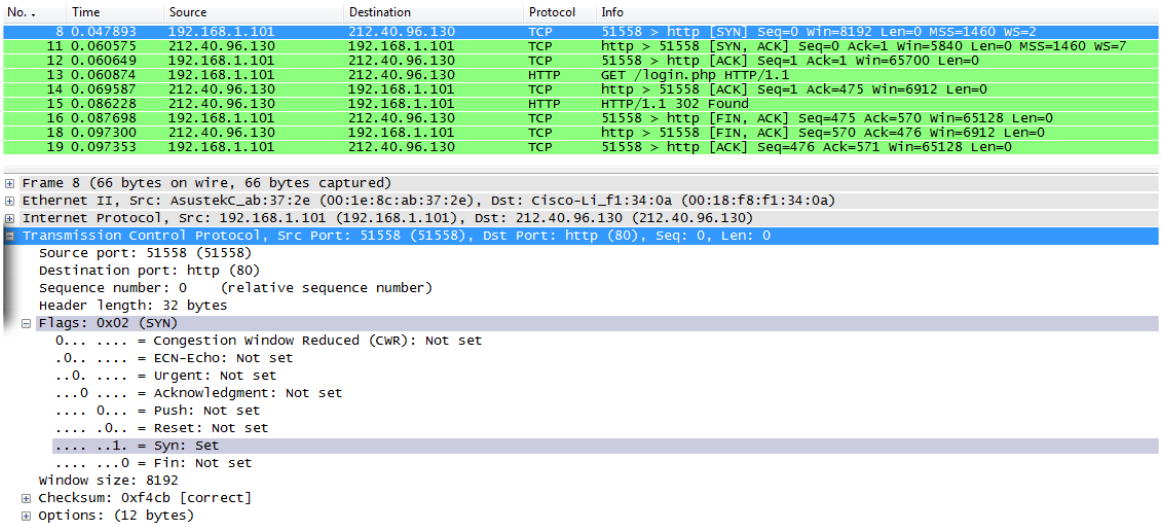
Jó kis csapdahelyzet.

Végül úgy hidaltam át, hogy most részletesen végigmegyünk egy példán. Ugyan látjuk benne, hogyan épül ki egy TCP kapcsolat, futólag meg is lesznek említve benne ismeretlen dolgok - de elsősorban az SN/ACK kommunikációt fogom benne részletezni.
A példa után jön majd egy TCP opciókat bemutató rész, végül jöhet a TCP kapcsolat működését részletező fejezet. Igaz, itt már lesz egy kis redundancia, hiszen a kapcsolat kialakulását már a példában is láthattuk - de inkább redundancia legyen, mint hiány.

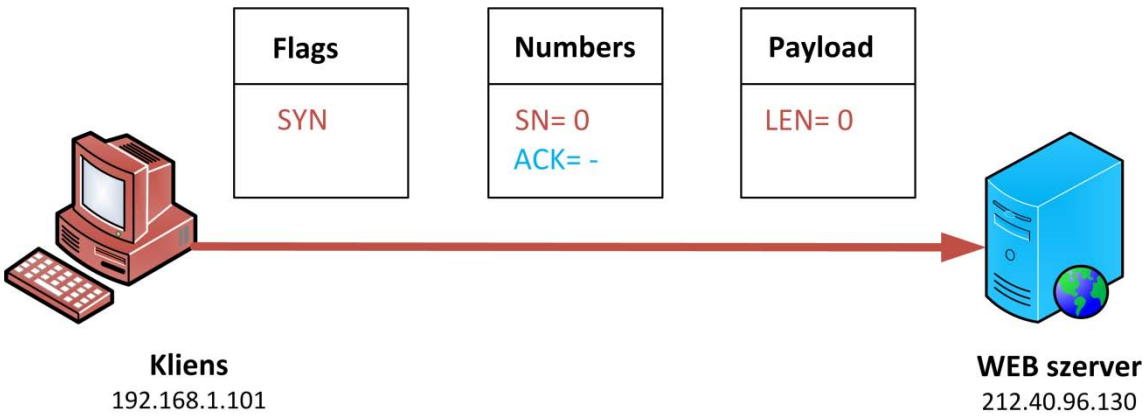
5.2.1 A SEQUENCE NUMBER ÉS AZ ACKNOWLEDGMENT NUMBER PINGPONGCSATA

A következőkben bemutatom, hogyan épül ki egy TCP csatorna és hogyan indul be rajta a forgalom. A történet a következő: a 192.168.1.101-es IP címmel rendelkező kliens számítógép le akar tölteni a 212.40.96.130-as IP címmel rendelkező webszerverről egy weblapot.

5.2.1.1 1. LÉPÉS: SYN



5.11. ÁBRA PACKET #08



5.12. ÁBRA SYN

A kliens gép elindítja az ún. háromlépéses kézrázást, avagy a csip-csip-csókát[77]. Az első lépésben magasra állítja a SYN flag-et és elküldi a kezdő SN-t. Ugye tudjuk, ha a

[77] Copyright by Fóti Marcell.

SYN flag magas, akkor az SN egyben ISN is. Ebben az esetben az ACK még értelmezhetetlen.

Jogos lehet a kérdés, miért pont nulla az ISN? Egyáltalán, mindig nulla?

TCP Connection Establishment Sequence Number Synchronization and Parameter Exchange:
http://www.tcpipguide.com/free/t_TCPConnectionEstablishmentSequenceNumberSynchroniz.htm

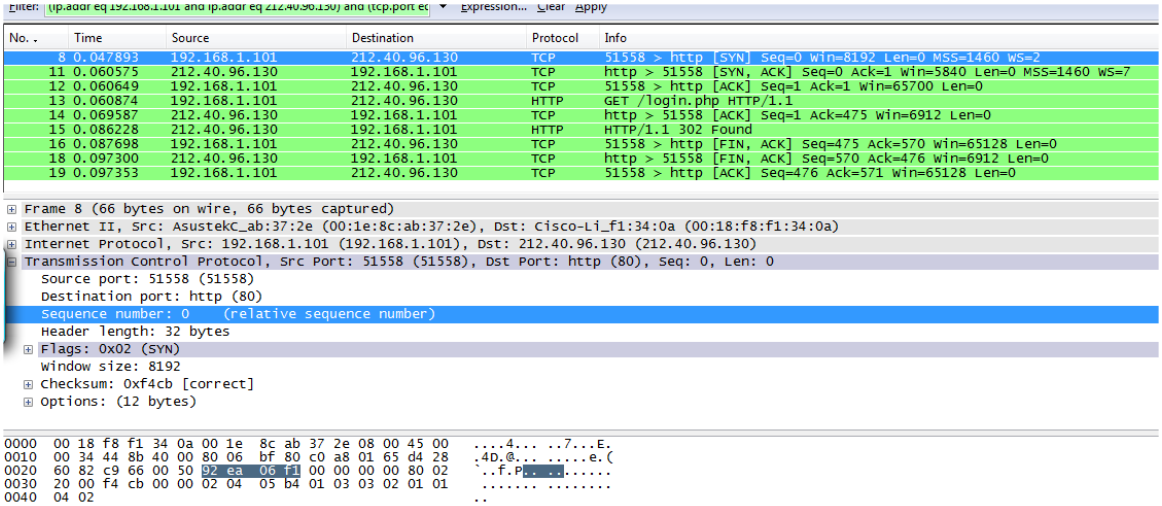
Az interneten meglehetősen sok cikk foglalkozik vele, miért okoz komoly problémákat, ha egy operációs rendszerben mindig ugyanaz az ISN. A Sequence Number értékek generálása ugyanis egy algoritmus szerint történik. Ez az algoritmus operációs rendszerenként változik. Viszont ha a rosszfiú ismeri egy kapcsolat esetén az ISN-t, ismeri az operációs rendszert, azaz a használt algoritmust, akkor simán el tudja téríteni egy kapcsolat felépítését, hiszen tudni fogja, milyen számsorozatot követnek az ACK értékek, tehát tudja, milyen SN értékeket kell visszaválaszolnia.

Szóval semmiképpen sem illik, hogy mindig ugyanaz legyen az ISN, az meg végképp nonszensz, hogy mindig nulla.

Ehhez képest a Microsoft két zászlóshajójánál[78] - a capture fájlok alapján - az ISN mindig nulla.
Izé. Hogy is van ez?

Hát úgy, hogy megint a Wireshark volt túlságosan is szolgálatkész. A megoldás ismét a Packet Bytes részen rejtőzik.

[78] Jó-jó, mire ez a könyv megjelenik, az egyik már nem lesz zászlós. De most még az.



5.13. ÁBRA A TRÜKKÖS KLIENS ISN

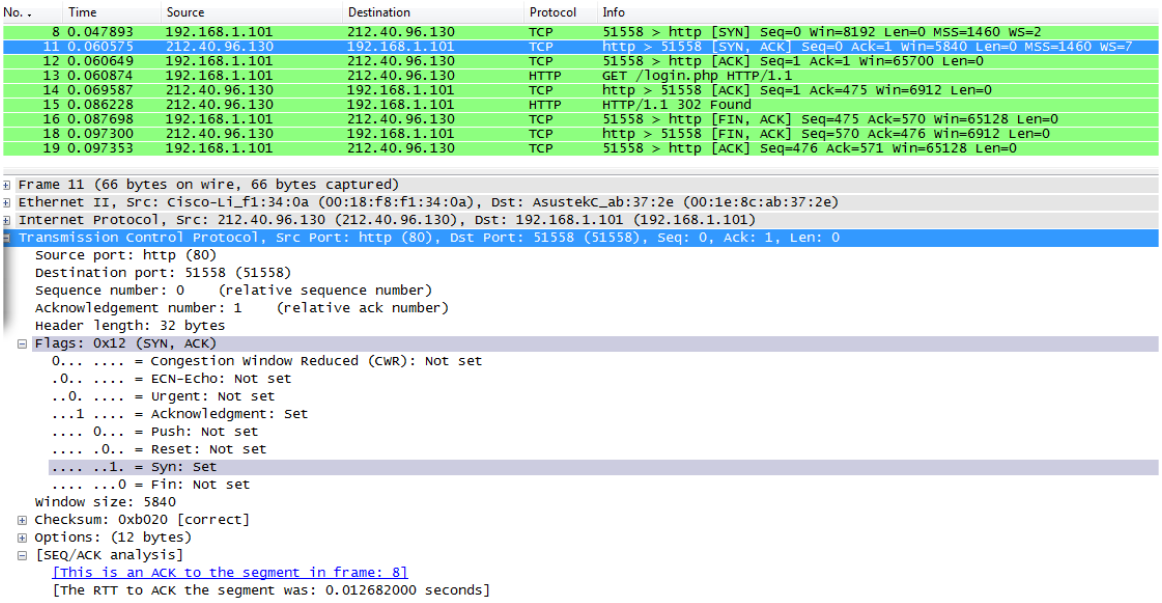
Nézzük csak meg. Azt mondja, hogy az ISN értéke nulla. (Becsületére legyen mondva, figyelmeztet, hogy ez relative ISN.)

De a helyes értéket akkor kapjuk meg, ha a Packet Bytes részre vetjük a pillantásunkat:
92 EA 06 F1 - azaz 2464810737. Szép nagy szám. Könnyen bele is lehet zavarodni. Mivel a későbbi SN/ACK értékek ennél csak nagyobbak lesznek, a Wireshark nemes egyszerűséggel csak a növekményeket jelzi ki.

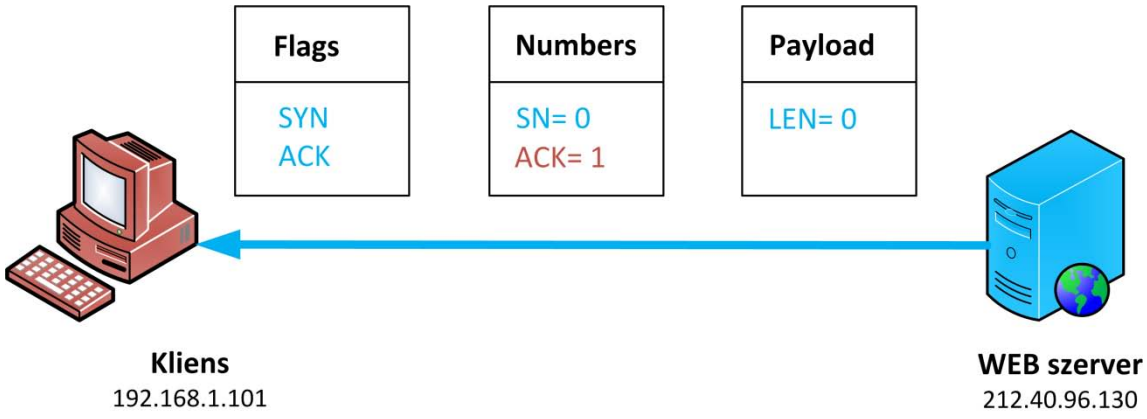
Hogy követhetőek legyenek a számok, az ábrákon különböző színekkel jelöltem, hogy mikor melyik alap növekvényéről lesz szó. A bordó jelöli a kliens oldalt, tehát ha azt látjuk, hogy **ACK=1**, akkor az gyakorlatilag azt jelenti, hogy ACK= ISN_{kliens} + 1, azaz 2464810737+1.

Na, ez az ISN érték már nem nulla. Meg nem is tűnik túl szabályosnak sem. Nem is az. Windows Server 2008/Vista esetén ugyanis az ISN egy megsózott, megfűszerezett, generált véletlenszám.

5.2.1.2 2. LÉPÉS: SYN-ACK

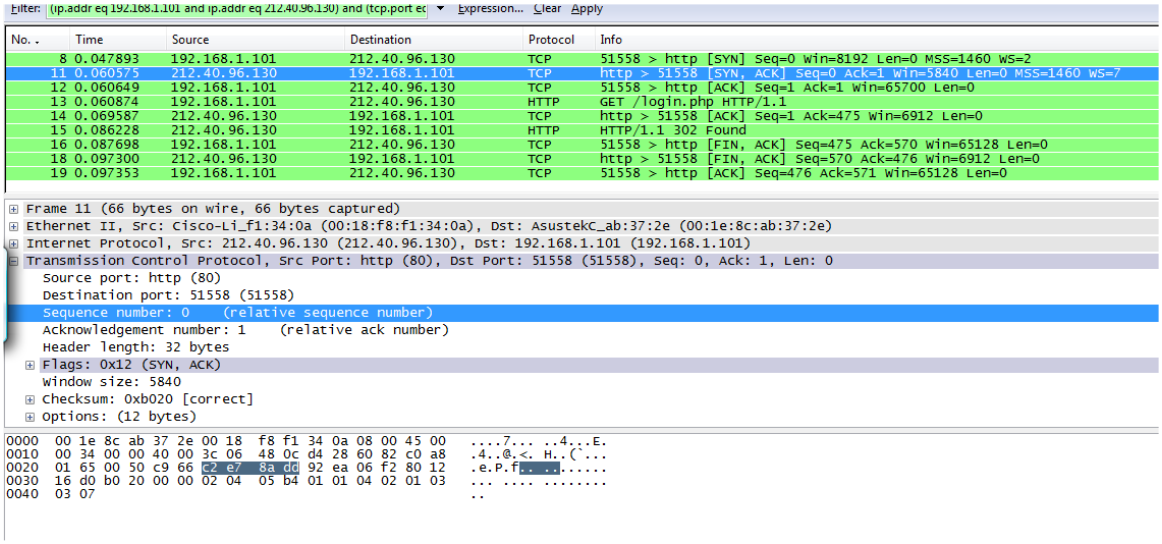


5.14. ÁBRA PACKET #11



5.15. ÁBRA SYN-ACK

A webszerver - mely egyáltalán nem biztos, hogy Windows - szintén generál magának egy Sequence Number értéket. Nézzük csak... a SYN flag magas, tehát ez is ISN. Ez konkrétan a webszerver ISN-je.



5.16. ÁBRA A TRÜKKÖS WEBSZERVER ISN

Csak a teljesség kedvéért: ez sem nulla. C2 E7 8A DD, azaz 3269954269. Az előző ponthoz hasonlóan a szerver oldali ISN-hez képest értelmezett növekményeket a kék szín jelöli az ábrákon.

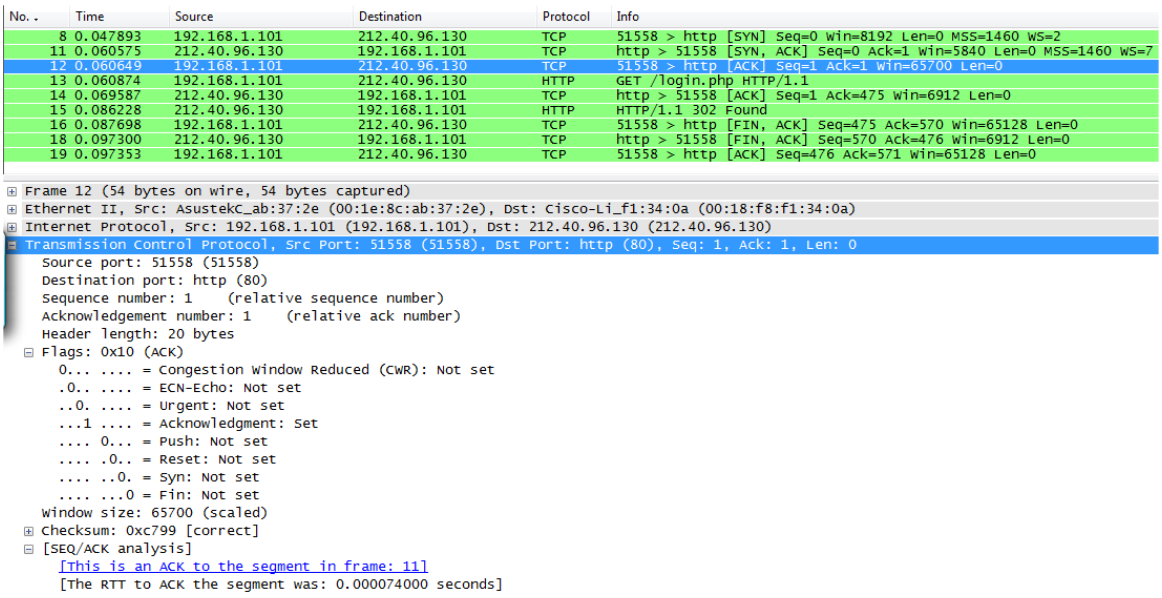
Újdonság, hogy a csomagban már nem csak a SYN flag magas, hanem az ACK flag is. Ez azt jelzi, hogy tessék figyelni, mert az Acknowledgment Number mezőnek már van értéke.
Mit takar ez az érték?
Visszajelzés? De kérem, hova? Kinek?
Nem, nem.

Ez egy mandiner. *Azt mondja, hogy küldök neked egy értéket az ACK mezőben. Ha rendben megkaptad a csomagot, akkor a következő küldeményed SN értéke legyen az, mint amit én neked most ebben az ACK mezőben elküldtem.*

Mennyi az értéke? Az ábrán 1... de tudjuk, hogy ez relatív. A szín bordó, tehát ez az érték most egész pontosan: $ISN_{kliens} + 1$.

Miért pont ennyi? Türelem, később minden kiderül. Pontosabban, még bonyolultabb lesz. Igaz, logikusabb is.

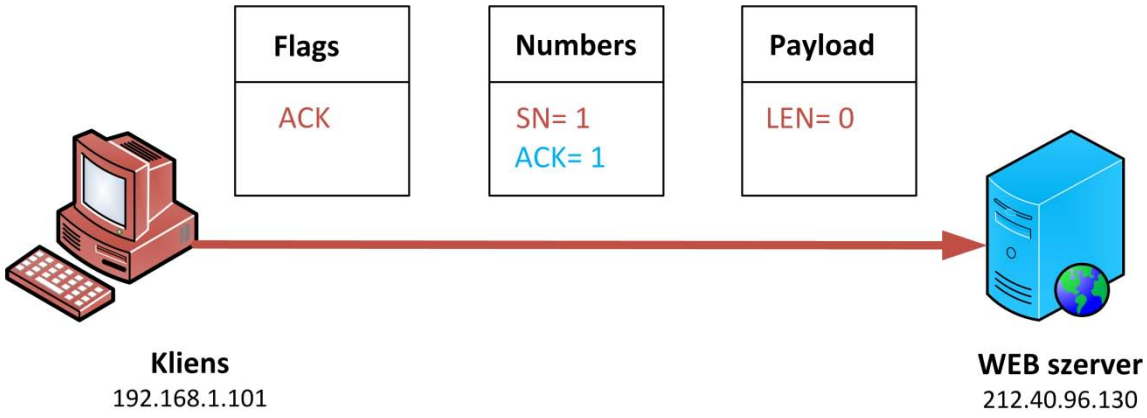
5.2.1.3 3. LÉPÉS: ACK



No.	Time	Source	Destination	Protocol	Info
8	0.047893	192.168.1.101	212.40.96.130	TCP	51558 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
11	0.060575	212.40.96.130	192.168.1.101	TCP	http > 51558 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 WS=7
12	0.060649	192.168.1.101	212.40.96.130	TCP	51558 > http [ACK] Seq=1 Ack=1 Win=65700 Len=0
13	0.060874	192.168.1.101	212.40.96.130	HTTP	GET /login.php HTTP/1.1
14	0.069587	212.40.96.130	192.168.1.101	TCP	http > 51558 [ACK] Seq=1 Ack=475 Win=6912 Len=0
15	0.086228	212.40.96.130	192.168.1.101	HTTP	HTTP/1.1 302 Found
16	0.087698	192.168.1.101	212.40.96.130	TCP	51558 > http [FIN, ACK] Seq=475 Ack=570 Win=65128 Len=0
18	0.097300	212.40.96.130	192.168.1.101	TCP	http > 51558 [FIN, ACK] Seq=570 Ack=476 Win=6912 Len=0
19	0.097353	192.168.1.101	212.40.96.130	TCP	51558 > http [ACK] Seq=476 Ack=571 Win=65128 Len=0

```
⊞ Frame 12 (54 bytes on wire, 54 bytes captured)
⊞ Ethernet II, Src: AsustekC_ab:37:2e (00:1e:8c:ab:37:2e), Dst: Cisco-Li_f1:34:0a (00:18:f8:f1:34:0a)
⊞ Internet Protocol, Src: 192.168.1.101 (192.168.1.101), Dst: 212.40.96.130 (212.40.96.130)
⊟ Transmission Control Protocol, Src Port: 51558 (51558), Dst Port: http (80), Seq: 1, Ack: 1, Len: 0
    Source port: 51558 (51558)
    Destination port: http (80)
    Sequence number: 1    (relative sequence number)
    Acknowledgement number: 1    (relative ack number)
    Header length: 20 bytes
  ⊟ Flags: 0x10 (ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgment: Set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..0. = Syn: Not set
      .... ...0 = Fin: Not set
    Window size: 65700 (scaled)
  ⊞ Checksum: 0xc799 [correct]
  ⊟ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 11]
      [The RTT to ACK the segment was: 0.000074000 seconds]
```

5.17. ÁBRA PACKET #12

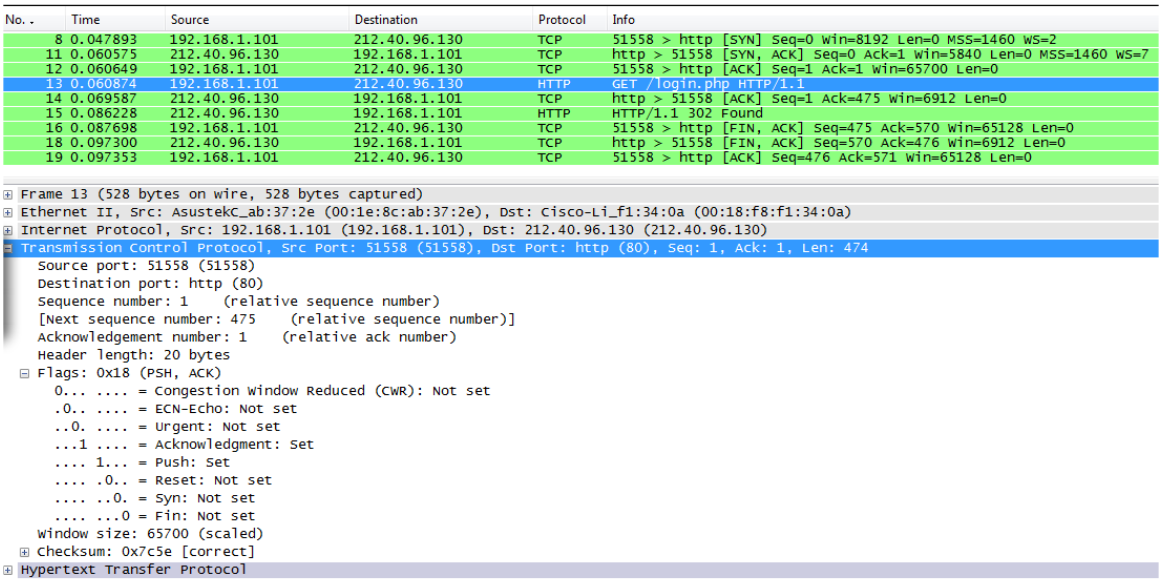


5.18. ÁBRA ACK

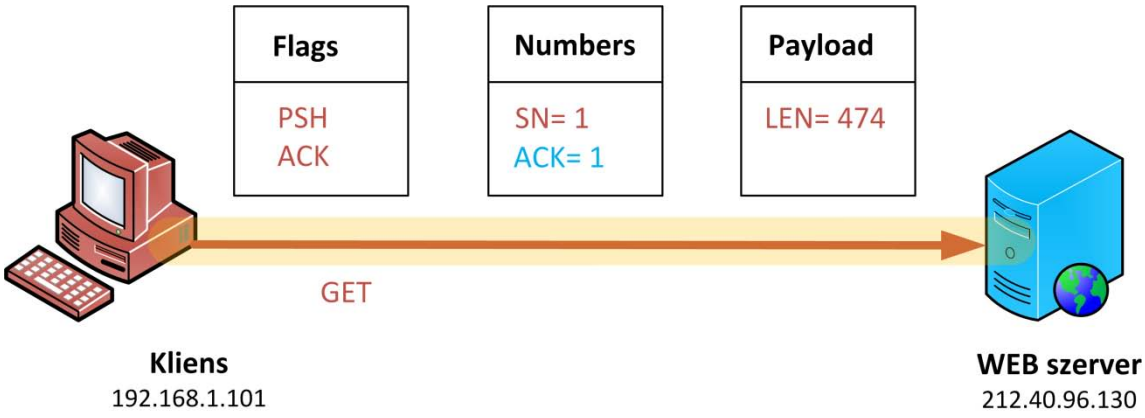
A kliens megkapta a csomagot, majd jófiúhoz méltóan az SN mezőbe a kért értéket tette. De hogy ő is tudja ellenőrizni azt, hogy a webszerver is megkapott-e minden csomagot, ő is küld egy ACK értéket. (Természetesen az ACK flag magas.)
És ez az ACK mennyi? Nyilván ez sem 1. A szín kék, az érték pedig egész pontosan most az $ISN_{webszerver}$ + 1.
Azaz az a csúfság esett meg, hogy az ábrán mind az SN, mind az ACK értéke ugyanúgy 1, a valóságban viszont a két szám teljesen különböző.

5.2.1.4 4. LÉPÉS: GET

Vegyük észre, fontos dolog történt. Megtörtént a háromfogásos kézrázás, kiépült a csatorna. Indulhat rajta a forgalom.



5.19. ÁBRA PACKET #13



5.20. ÁBRA GET

Nocsak. Megint a kliens akciózott. Persze ez logikus, hiszen az egész történet úgy indult, hogy a kliens akart valamit a webszervertől. Először természetesen kiépítik a csatornát, és mivel az 3 lépésből áll, így mindig a kliens fejezi be, majd HTTP esetében utána mondja csak meg, hogy mit is akart. Jelen esetben elküldött egy kérést a webszervernek, miszerint szeretne letölteni egy weblapot.

Ebben a kérésben sem üresek az SN/ACK értékek, de nem is változtak, hiszen az előző csomaghoz képest nem történt változás. (Ugyanúgy a kliens küld.)

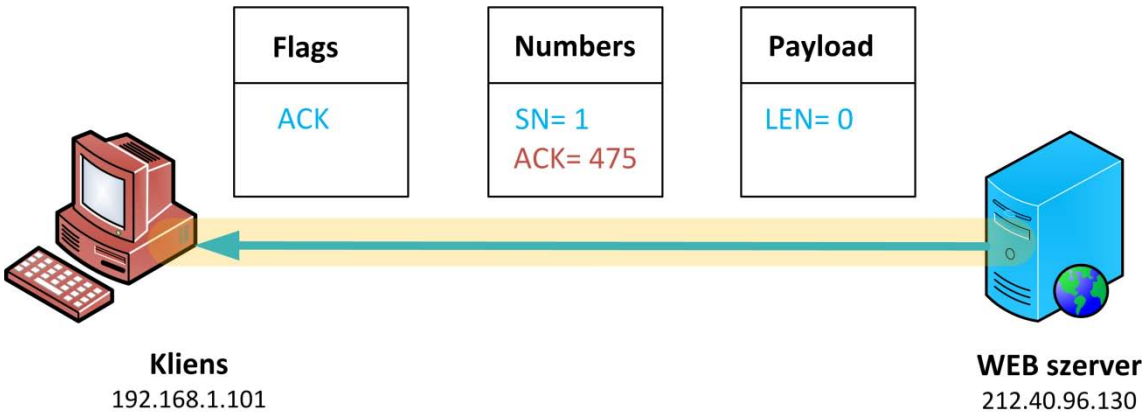
Azért egy érdekesség van, a flag-ek között megjelent a PSH flag. Ezzel majd később fogunk találkozni, most elég annyi róla, hogy ez kiüríti a fogadó állomás pufferét - azaz a kérés egyből felkerül az alkalmazáshoz.

5.2.1.5 5. LÉPÉS: ACK

No.	Time	Source	Destination	Protocol	Info
8	0.047893	192.168.1.101	212.40.96.130	TCP	51558 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
11	0.060575	212.40.96.130	192.168.1.101	TCP	http > 51558 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 WS=7
12	0.060649	192.168.1.101	212.40.96.130	TCP	51558 > http [ACK] Seq=1 Ack=1 Win=65700 Len=0
13	0.060874	192.168.1.101	212.40.96.130	HTTP	GET /login.php HTTP/1.1
14	0.069587	212.40.96.130	192.168.1.101	TCP	http > 51558 [ACK] Seq=1 Ack=475 Win=6912 Len=0
15	0.086228	212.40.96.130	192.168.1.101	HTTP	HTTP/1.1 302 Found
16	0.087698	192.168.1.101	212.40.96.130	TCP	51558 > http [FIN, ACK] Seq=475 Ack=570 Win=65128 Len=0
18	0.097300	212.40.96.130	192.168.1.101	TCP	http > 51558 [FIN, ACK] Seq=570 Ack=476 Win=6912 Len=0
19	0.097353	192.168.1.101	212.40.96.130	TCP	51558 > http [ACK] Seq=476 Ack=571 Win=65128 Len=0

```
⊞ Frame 14 (60 bytes on wire, 60 bytes captured)
⊞ Ethernet II, Src: Cisco-Li_f1:34:0a (00:18:f8:f1:34:0a), Dst: AsustekC_ab:37:2e (00:1e:8c:ab:37:2e)
⊞ Internet Protocol, Src: 212.40.96.130 (212.40.96.130), Dst: 192.168.1.101 (192.168.1.101)
⊟ Transmission Control Protocol, Src Port: http (80), Dst Port: 51558 (51558), Seq: 1, Ack: 475, Len: 0
    Source port: http (80)
    Destination port: 51558 (51558)
    Sequence number: 1    (relative sequence number)
    Acknowledgement number: 475    (relative ack number)
    Header length: 20 bytes
  ⊟ Flags: 0x10 (ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgment: Set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..0. = Syn: Not set
      .... ...0 = Fin: Not set
    Window size: 6912 (scaled)
  ⊞ Checksum: 0x05b3 [correct]
  ⊟ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 13]
      [The RTT to ACK the segment was: 0.008713000 seconds]
```

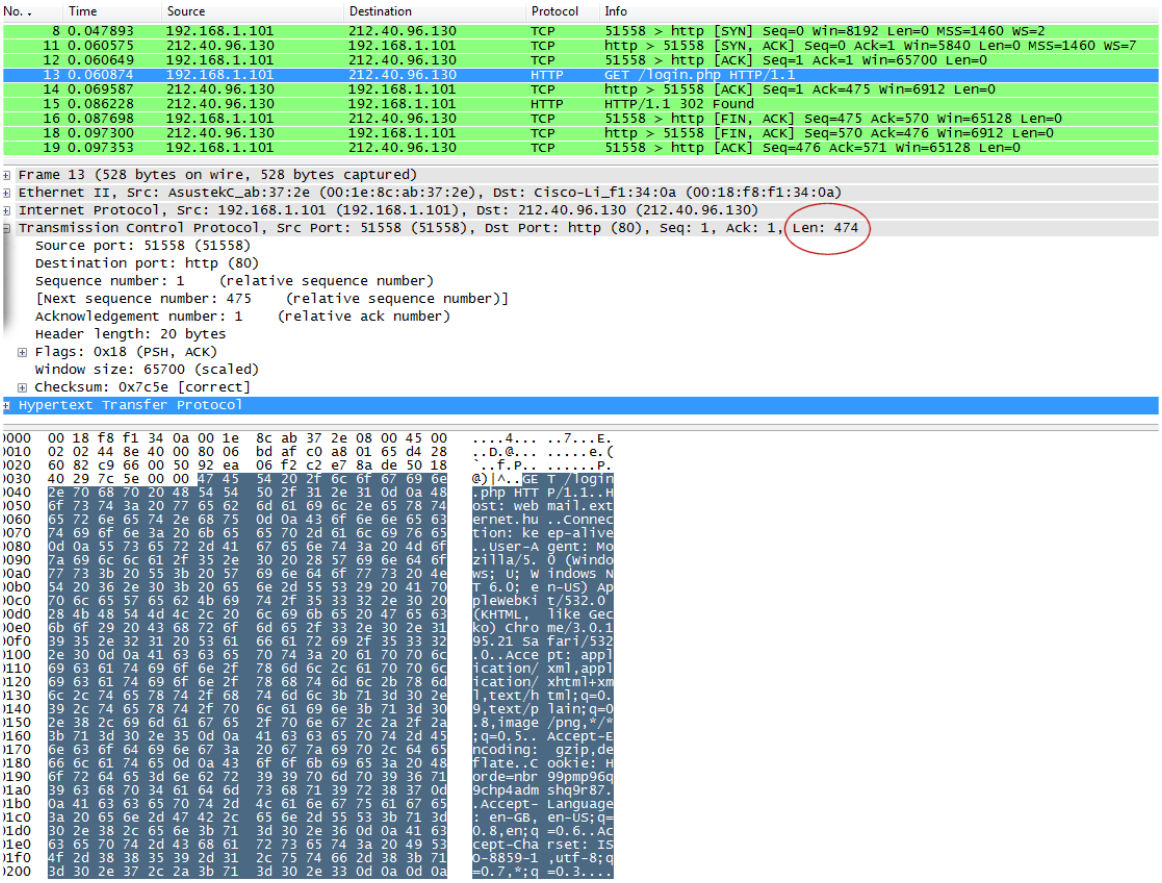
5.21. ÁBRA PACKET #14



5.22. ÁBRA ACK

Kezd bevadulni a kommunikáció. A webszerver az SN mezőbe még a megkívánt $ISN_{webszerver}$+1 értéket írja, de az ACK-be már feladja a leckét: a következő csomagban kérek egy ISN_{kliens}+475-ös SN értéket. (Hogyan jött ez ki neki? A fene tudja. Azt se tudom, milyen operációs rendszer fut rajta, honnan tudnám akkor a generáló algoritmusát?)

Vigyázat, hazudok.



5.23. ÁBRA A HTTP KÉRÉS TARTALMA

Egyfelől a 15-ös csomagból tudom, hogy egy Apache webszerver szolgált ki, ami azért behatárolja az operációs rendszert.
De nem ez volt a legnagyobb hazugság. Hanem az, hogy nem tudom, miért pont 475.

Nézzük csak meg a korábbi (13-as csomag) GET kérésünket a Packet Bytes rovatban a fenti ábrán. Ez egy szép nagy adatkupac. Mekkora? Azt mondja, 16 oszlop, 30 sor, az annyi mint 480 bájt, ebből lejön az első hat, mert az nincs bemeszelve, akkor az pont 474. Azaz ennyi a TCP payload, azaz a TCP szegmens. Ez az érték nem utazik ugyan a csomagban, de a Wireshark szolgálatkészen kijelzi a TCP összegző sorban.

Pár oldallal ezelőtt problémáztam azon, hogy miért pont annyi az ACK, amennyi. Nos, ezért. A fogadó azt mondja, hogy megkaptam tőled az Y SN értékű csomagban X bájt payload-ot, a következő csomagnak tehát Y+X+1-től kell indulnia - azaz erre a Sequence Number értékre fogok számítani.

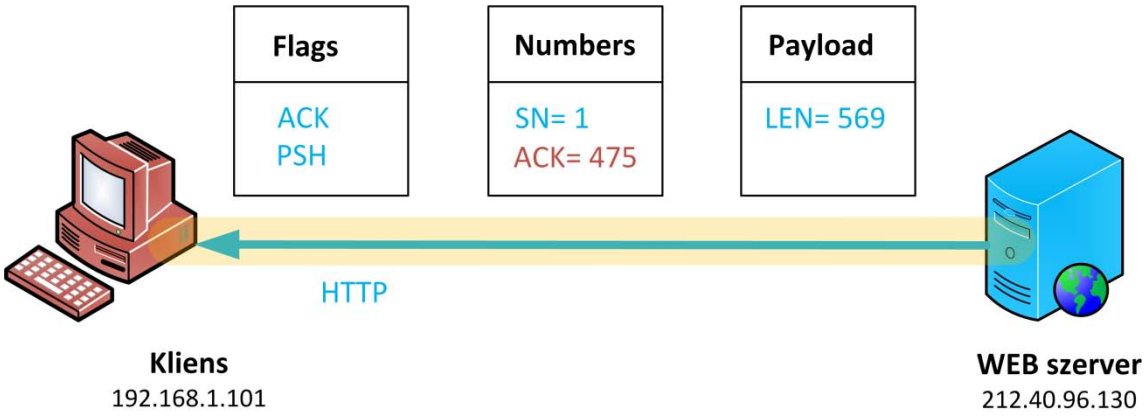
Nézzük meg, milyen ACK-et küld vissza a webszerver? ACK=475. Azaz őt kérem legközelebb SN-ként. És akkor én is könnyebben tudom sorbarakni a csomagokat.

5.2.1.6 6. LÉPÉS: HTTP

No.	Time	Source	Destination	Protocol	Info
8	0.047893	192.168.1.101	212.40.96.130	TCP	51558 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
11	0.060575	212.40.96.130	192.168.1.101	TCP	http > 51558 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 WS=7
12	0.060649	192.168.1.101	212.40.96.130	TCP	51558 > http [ACK] Seq=1 Ack=1 Win=65700 Len=0
13	0.060874	192.168.1.101	212.40.96.130	HTTP	GET /login.php HTTP/1.1
14	0.069587	212.40.96.130	192.168.1.101	TCP	http > 51558 [ACK] Seq=1 Ack=475 Win=6912 Len=0
15	0.086228	212.40.96.130	192.168.1.101	HTTP	HTTP/1.1 302 Found
16	0.087698	192.168.1.101	212.40.96.130	TCP	51558 > http [FIN, ACK] Seq=475 Ack=570 Win=65128 Len=0
18	0.097300	212.40.96.130	192.168.1.101	TCP	http > 51558 [FIN, ACK] Seq=570 Ack=476 Win=6912 Len=0
19	0.097353	192.168.1.101	212.40.96.130	TCP	51558 > http [ACK] Seq=476 Ack=571 Win=65128 Len=0

```
⊞ Frame 15 (623 bytes on wire, 623 bytes captured)
⊞ Ethernet II, Src: Cisco-Li_f1:34:0a (00:18:f8:f1:34:0a), Dst: AsustekC_ab:37:2e (00:1e:8c:ab:37:2e)
⊞ Internet Protocol, Src: 212.40.96.130 (212.40.96.130), Dst: 192.168.1.101 (192.168.1.101)
⊟ Transmission Control Protocol, Src Port: http (80), Dst Port: 51558 (51558), Seq: 1, Ack: 475, Len: 569
    Source port: http (80)
    Destination port: 51558 (51558)
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 570    (relative sequence number)]
    Acknowledgement number: 475    (relative ack number)
    Header length: 20 bytes
  ⊟ Flags: 0x18 (PSH, ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgment: Set
      .... 1... = Push: Set
      .... .0.. = Reset: Not set
      .... ..0. = Syn: Not set
      .... ...0 = Fin: Not set
    Window size: 6912 (scaled)
  ⊞ Checksum: 0xab4b [correct]
⊞ Hypertext Transfer Protocol
```

5.24. ÁBRA PACKET #15



5.25. ÁBRA HTTP

Most megint valami nem várt dolog történt: ismét a webszerver küldött egy csomagot. Miért is? Nos, az előző, azaz a 13-as csomag gyakorlatilag üres volt, pusztán egy ACK volt benne. Az igazi válasz, azaz a kliens által kért weblap a mostani csomagban megy vissza.

Miért nem lehetett egyszerre, egy csomagban leackolni a kliens kérését és rögtön abba bele is rakni a választ?

Ezen most el fogunk agyalni egy kicsit, ugyanis a TCP egyik sarkalatos pontját csíptük meg. Mikor kell leackolni egy csomagot? Mindig? Minek?

Gondolj bele, ott van egy böszme nagy weblap - mondjuk a www.index.hu - vagy egy 300 MB méretű HP printer driver. Én meg mezei Ethernet hálózaton vagyok, 1460 bájtos MSS értékkel. Ez testvérek között is 215460 ACK csomag küldését jelentené tőlem a printer driver esetében. Csak úgy parasztosan megszámoltam a 14-es csomagban - mely egy üres ACK - a bájtokat, az jött ki, hogy 76. Azaz a 300 MB méretű driver letöltéséhez nekem 15.6 MB ACK csomagot kellene feltöltenem.

Nem hangzik túl jól.

Mit lehetne kitalálni, hogy csökkenjen ez a vízfej?

Például létrehozunk a fogadó oldalon egy ablakot. Azt mondjuk, hogy ez az ablak befogad mondjuk 10 csomagot, ha betelt, csak akkor küldünk majd vissza egy ACK-et. Máris csak 1,5 MB a plusz forgalom.

Csakhogy. Gondolkozzunk, Béláim.

Egy kicsit előrerohanok. (Később jobban is ki lesz tárgyalva ez a rész.)
A helyzet az, hogy nem csak fogadó oldalon van egy puffer, hanem küldő oldalon is, méghozzá a küldő oldali eleinte kisebb, mint a fogadó oldali és később is csak maximum egyenlőek lehetnek. A küldő oldali puffer paramétere a Send Window, ami azt a méretet jelenti, amíg a feladónak nem kell várnia a visszaküldött ACK-re.
Az egyszerűség kedvéért tételezzük fel, hogy a Send Window értéke 5 csomag. (Figyelem, valójában itt bájtokban mérünk, de most a szemléltetés a cél.)
Ekkor küldünk és csak küldünk. A fogadó meg csak vár és vár, mert azt mondtuk neki, hogy 10 csomag után ackoljon. A feladó viszont 5 csomag után leáll, mert megtelt a Send Window. És azóta is vártak, míg meg nem haltak.
Egy másik lehetséges epic fail ebben az esetben az, ha a küldendő csomag mérete nem éri el fogadó puffer méretét. A feladó befejezte a küldést, a fogadó meg nem igazol vissza, mert még nem érte el a mennyiség a 10 csomagot.

Szóval ez így nem működik.

Marad a csomagonkénti ACK, a jó 5%-os vízfejjel.

Illetve egy picit lehet még kozmetikázni rajta. Ezt nevezik úgy, hogy delayed ACK. Nem egyből válaszolunk vissza a feladónak, hanem várunk egy kicsit. Ha 0,5 másodpercen belül[79] megy vissza valamilyen adatcsomag, akkor nyertünk, mert ennek a csomagnak a nyakába ültetjük (piggyback) az ACK-et.

Ezt a csomagot látjuk ebben az alfejezetben.

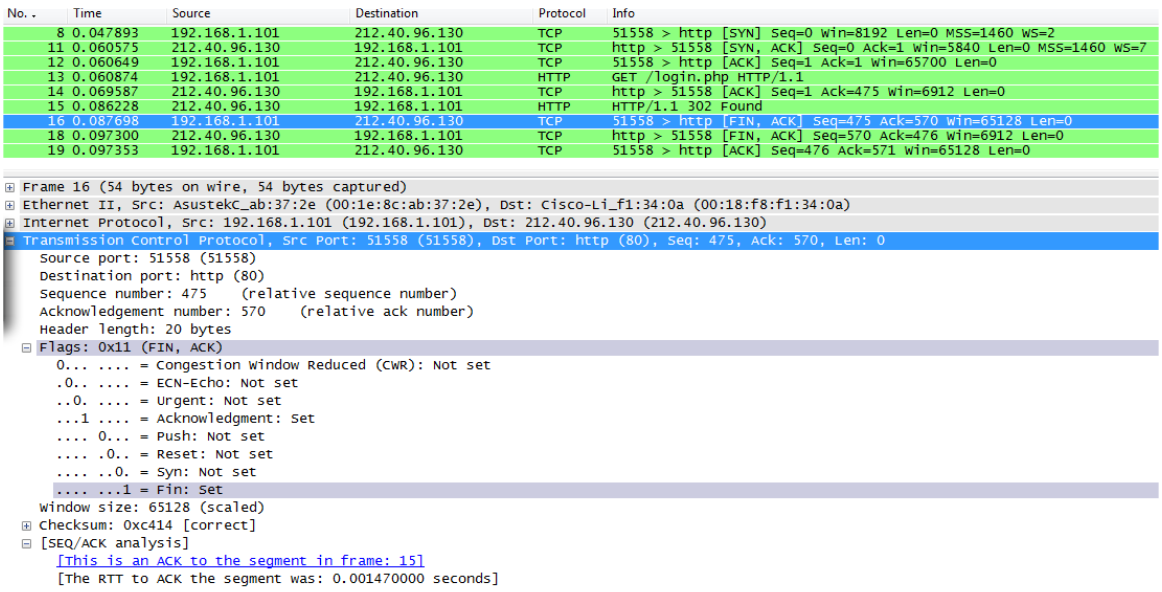
Igen, érzem én is, hogy sántít egy kicsit a példa - hiszen egyszer már leackoltuk a csomagot (14), miért kell most még egyszer?
Őszintén szólva, nem tudom. Mivel ez egy piggyback ACK - azaz nem jelent számottevő terhelést a forgalomban - csak arra tudok gondolni, hogy a fogadó oldali implementáció inkább nem vizsgálgatott, hanem ész nélkül rányomta még egyszer az ACK-et, ha már úgyis ment éppen arra egy csomag.

De ha már itt járunk, említsünk meg még néhány trükköt. Pl. nagyméretű adatmennyiség küldésénél megelégszünk a két csomagonkénti ACK-kel, ezzel rögtön felére redukálva a visszirányú forgalmat. Vagy PSH flag-gel ellátott csomagot nem kell egyből ackelni. Aztán vannak olyan esetek, amikor korábban elveszett csomagokat küldünk át újra (SACK), ilyenkor a több csomagból álló übercsomaghoz már csak egy ACK jár. Vagy ha már cifrázzuk: például küldenek nekünk egy hat csomagból álló vonatot, de mondjuk az első csomag elveszik. Ekkor hiába kapjuk meg a többi ötöt, nem tudjuk leackolni. Majd amikor lejár az idő és a feladó nem kap semmit és emiatt újraküldi mind a hat csomagot és végre fogadó oldalon összeáll a csomag - ekkor az utolsó csomag leackolása automatikusan visszaigazolja mind a hat csomagot. (Kleine Fische, gute Fische - nem egy nagy spórolások, de sok kicsi sokra megy.)

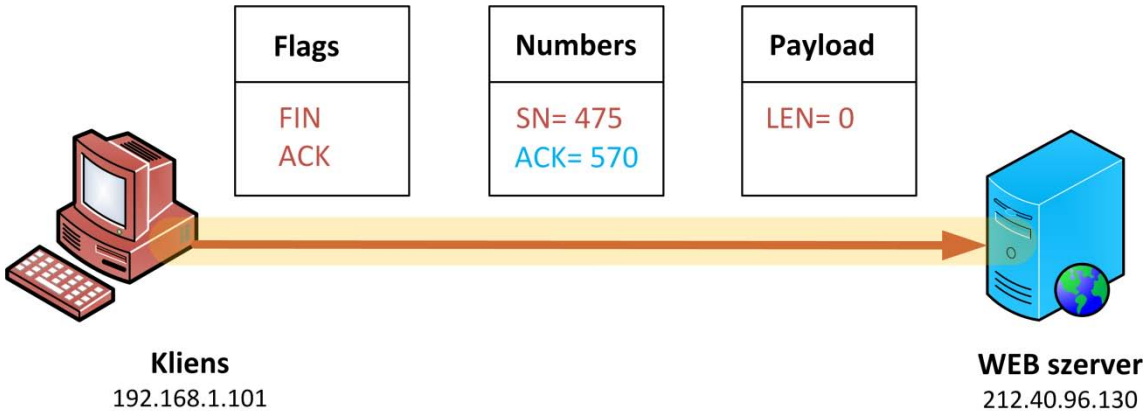
De most már tényleg nagyon előrerohantam, térjünk vissza a pingpongmeccshez.

[79] Az RFC szerint 0,5 sec. Windows Server 2008 és Vista esetében ez konkrétan 0,2 sec.

5.2.1.7 7. LÉPÉS: FIN-ACK



5.26. ÁBRA PACKET #16



5.27. ÁBRA FIN-ACK

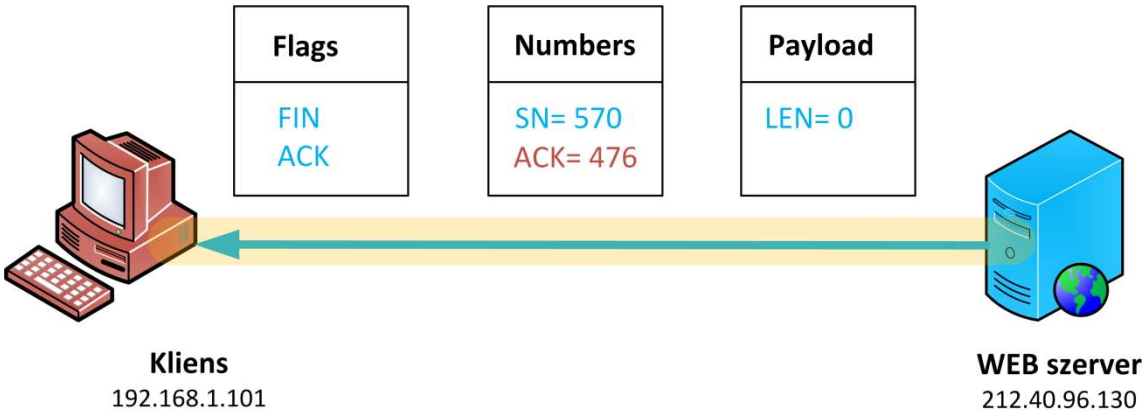
A kliens megkapta a kért tartalmat - és mivel egyelőre másra nincs szüksége, kezdeményezte is a kapcsolat bontását. Ezt jelenti a FIN flag bebillentése.
Ami sokkal izgalmasabb, hogy a középső dobozban begerjedtek a számok. Mivel a szerver az előző ACK-ben a 475-ös értéket kérte, így ez is lett a Sequence Number. Csakhogy a szerver által küldött TCP payload-nak is volt egy mérete (*5.25. ábra HTTP*), egész pontosan 569 bájt. Ezért a kliens azt kérte a szervertől, hogy az ő következő SN értéke 570 legyen.

5.2.1.8 8. LÉPÉS: ACK-FIN

No. .	Time	Source	Destination	Protocol	Info
8	0.047893	192.168.1.101	212.40.96.130	TCP	51558 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
11	0.060575	212.40.96.130	192.168.1.101	TCP	http > 51558 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 WS=7
12	0.060649	192.168.1.101	212.40.96.130	TCP	51558 > http [ACK] Seq=1 Ack=1 Win=65700 Len=0
13	0.060874	192.168.1.101	212.40.96.130	HTTP	GET /login.php HTTP/1.1
14	0.069587	212.40.96.130	192.168.1.101	TCP	http > 51558 [ACK] Seq=1 Ack=475 Win=6912 Len=0
15	0.086228	212.40.96.130	192.168.1.101	HTTP	HTTP/1.1 302 Found
16	0.087698	192.168.1.101	212.40.96.130	TCP	51558 > http [FIN, ACK] Seq=475 Ack=570 Win=65128 Len=0
18	0.097300	212.40.96.130	192.168.1.101	TCP	http > 51558 [FIN, ACK] Seq=570 Ack=476 Win=6912 Len=0
19	0.097353	192.168.1.101	212.40.96.130	TCP	51558 > http [ACK] Seq=476 Ack=571 Win=65128 Len=0

```
Frame 18 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: Cisco-Li_f1:34:0a (00:18:f8:f1:34:0a), Dst: AsustekC_ab:37:2e (00:1e:8c:ab:37:2e)
Internet Protocol, Src: 212.40.96.130 (212.40.96.130), Dst: 192.168.1.101 (192.168.1.101)
Transmission Control Protocol, Src Port: http (80), Dst Port: 51558 (51558), Seq: 570, Ack: 476, Len: 0
    Source port: http (80)
    Destination port: 51558 (51558)
    Sequence number: 570    (relative sequence number)
    Acknowledgement number: 476    (relative ack number)
    Header length: 20 bytes
    Flags: 0x11 (FIN, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...1 = Fin: Set
    Window size: 6912 (scaled)
    Checksum: 0x0378 [correct]
    [SEQ/ACK analysis]
        [This is an ACK to the segment in frame: 16]
        [The RTT to ACK the segment was: 0.009602000 seconds]
```

5.28. ÁBRA PACKET #18



5.29. ÁBRA ACK-FIN

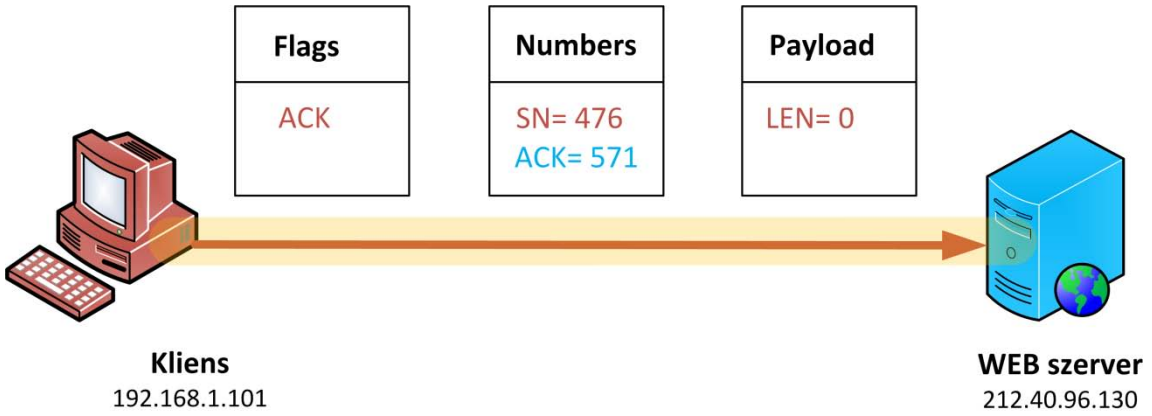
Folytatódik a kapcsolat bontása. A szerver leackolja a kliens bontási kérelmét - majd ő is elkezdi lebontani a tőle kiinduló kapcsolatot. (A kétirányú kapcsolatot két irányból kell bontani is.) Az SN értelemszerűen a kért érték, és mivel az előző csomagban a payload értéke 0 volt, így csak eggyel léptette az ACK értékét.

5.2.1.9 9. LÉPÉS: ACK

No.	Time	Source	Destination	Protocol	Info
8	0.047893	192.168.1.101	212.40.96.130	TCP	51558 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
11	0.060575	212.40.96.130	192.168.1.101	TCP	http > 51558 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 WS=7
12	0.060649	192.168.1.101	212.40.96.130	TCP	51558 > http [ACK] Seq=1 Ack=1 Win=65700 Len=0
13	0.060874	192.168.1.101	212.40.96.130	HTTP	GET /login.php HTTP/1.1
14	0.069587	212.40.96.130	192.168.1.101	TCP	http > 51558 [ACK] Seq=1 Ack=475 Win=6912 Len=0
15	0.086228	212.40.96.130	192.168.1.101	HTTP	HTTP/1.1 302 Found
16	0.087698	192.168.1.101	212.40.96.130	TCP	51558 > http [FIN, ACK] Seq=475 Ack=570 Win=65128 Len=0
18	0.097300	212.40.96.130	192.168.1.101	TCP	http > 51558 [FIN, ACK] Seq=570 Ack=476 Win=6912 Len=0
19	0.097353	192.168.1.101	212.40.96.130	TCP	51558 > http [ACK] Seq=476 Ack=571 Win=65128 Len=0

```
⊞ Frame 19 (54 bytes on wire, 54 bytes captured)
⊞ Ethernet II, Src: AsustekC_ab:37:2e (00:1e:8c:ab:37:2e), Dst: Cisco-Li_f1:34:0a (00:18:f8:f1:34:0a)
⊞ Internet Protocol, Src: 192.168.1.101 (192.168.1.101), Dst: 212.40.96.130 (212.40.96.130)
⊟ Transmission Control Protocol, Src Port: 51558 (51558), Dst Port: http (80), Seq: 476, Ack: 571, Len: 0
    Source port: 51558 (51558)
    Destination port: http (80)
    Sequence number: 476    (relative sequence number)
    Acknowledgement number: 571    (relative ack number)
    Header length: 20 bytes
  ⊟ Flags: 0x10 (ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgment: Set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..0. = Syn: Not set
      .... ...0 = Fin: Not set
    Window size: 65128 (scaled)
  ⊞ Checksum: 0xc413 [correct]
  ⊟ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 18]
      [The RTT to ACK the segment was: 0.000053000 seconds]
```

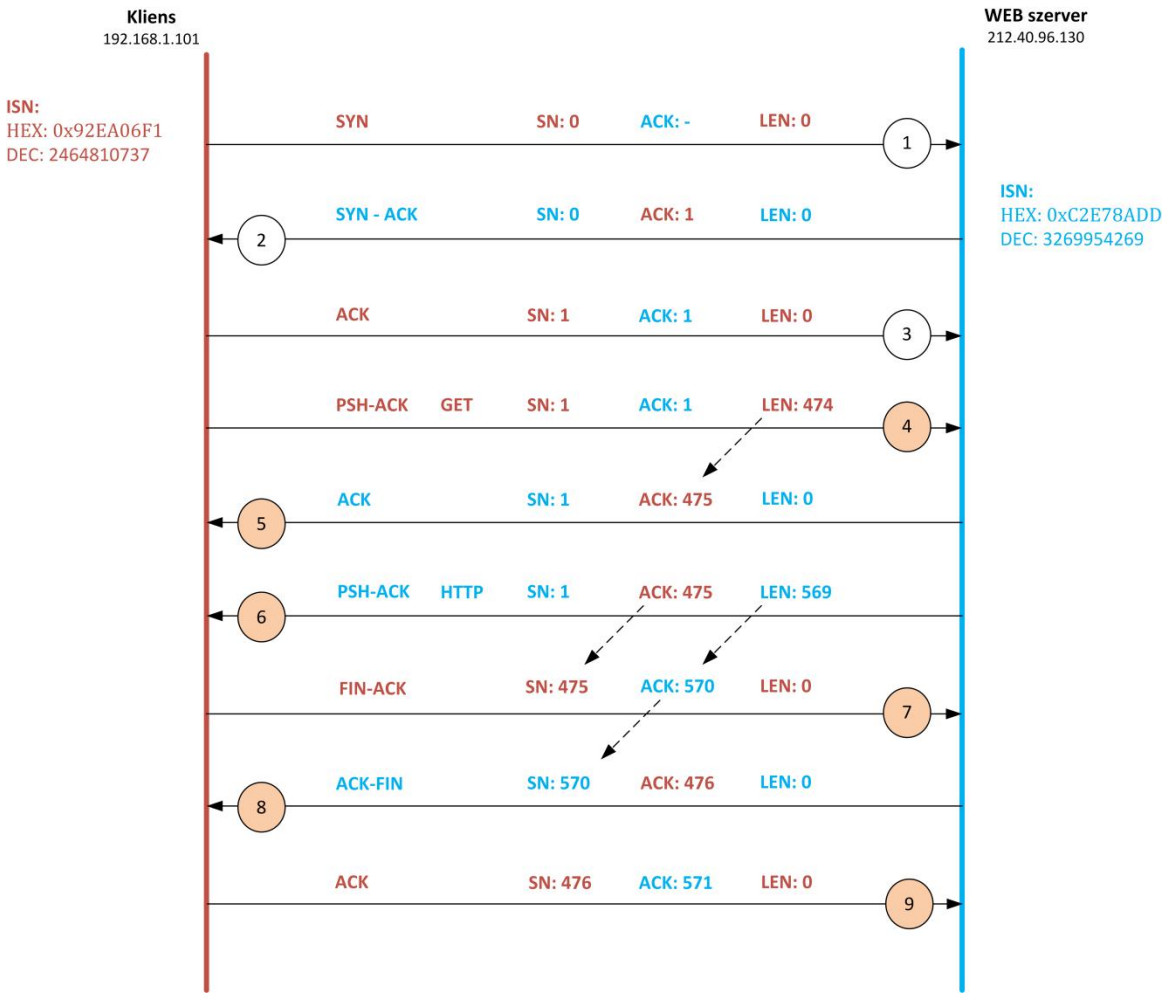
5.30. ÁBRA PACKET #19



5.31. ÁBRA ACK

Itt van vége a történetnek. A kliens visszaigazolja a szerver bontási kérelmét, a kapcsolat elillant. Az SN mezőbe a megkért érték került, sőt, egy kényszeres mechanizmus folytán a kliens még megad egy elvárt ACK-et is (üres payolad, azaz ACK+1), de ez már valójában érdektelen.

Az egész folyamatot megtekinthetjük egy összefoglaló ábrán is.

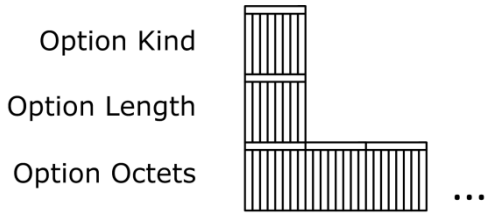


5.32. ÁBRA SN/ACK PATTOGTATÁS ÖSSZEFOGLALÓ ÁBRA

Végül egy megjegyzés:
Senkit ne zavarjon meg, hogy a capture képernyőkön időnként feltűnik egy Next Sequence Number nevű mező. Ez a Wireshark kényelmi szolgáltatása: kiolvassa a láncból és odateszi, hogy lássuk. A csomag keletkezésének pillanatában értelemszerűen ez az érték még nem létezik.

5.2.2 TCP Options

Annyi előnyünk már van, hogy ismerjük az IP Options szerepét, felépítését. Nagyjából hasonlóra számíthatunk itt is.



5.33. ábra TCP Options általános szerkezet

Lesz egy opció azonosító, lesz egy 1 bájtos érték, mely az opció hosszát jelöli - és utána jöhetnek az opcióspecifikus adatok.

5.2.2.1 End Of Option List és No Operation

Mindkettő 1 bájtos TCP opció.

- No Operation: Ez az opció határolja el egymástól a TCP opciókat. (Option Kind=0)
- End Of Option List : Ez pedig zárja a sort. (Option Kind=1)

5.2.2.2 MAXIMUM SEGMENT SIZE OPTION

A maximum segment size (MSS) azt a legnagyobb szegmensméretet jelenti, melyet tördelés nélkül lehet belerakni egy csomagba. A kiszámolása teljesen logikus: fogjuk az MTU-t, levonjuk belőle mind az IP headert, mind a TCP headert - és már meg is van. Vegyünk egy teljesen általános esetet: legyen az MTU 1500 bájt, se az IP, se a TCP headerben ne legyen semmilyen opció - ekkor mindegyik 20-20 bájt - azaz az MSS 1460 bájt lesz.

Egy kapcsolat kiépítésénél a legelső lépésekben történik meg az MSS belövése, méghozzá úgy, hogy amikor a SYN flag magas[80], abban a csomagban kerül átadásra az MSS értéke is - pont az MSS Option révén. Az Option Kind értéke: 2.

Ha valami furcsa véletlen folytán a két fél nem ugyanazt az MSS értéket közölné a másikkal, akkor a kisebb fog nyerni.

[80] Ez ugye az első két csip a csip-csip-csókában.

5.2.2.3 TCP WINDOW SCALE OPTION

Mint a TCP header tárgyalásánál is láttuk, a WINDOW mező mérete 2 bájt. Ez azt jelenti, hogy a receive window maximális mérete 65535 bájt. Ez a szabvány készítésekor maximálisan elegendőnek tűnt[81], de azóta haladt valamennyit előre a világ. Gigabites kapcsolatoknál miért ne lehetne böszme nagy ablakokat használni a folyamatos kommunikáció végett?

Ha nagyobb window méretet szeretnénk, akkor kénytelen leszünk használni a TCP Windows Scale opciót. Az Option Kind értéke: 3. Az opció adatmezőjében egy szám fog utazni, melynek maximális értéke 14 lehet. Valójában ez a szám egy hatványkitevő, ahol a hatványozás alapja 2. Az eredményül kapott számmal pedig meg kell szorozni az eredeti maximális ablakméret értékét, hogy az újat megkapjuk.
Ebből kifolyólag az opciókkal megsegített maximális ablakméret:
$2^{16} * 2^{14} = 2^{30} = 1073741824$ bájt. Közismertebb nevén 1 GB.

[81] *"640 kbyte mindenre elegendő lesz!"*

5.2.2.4 Selective Acknowledgment (SACK) Option

Esett már szó róla, hogy mi van akkor, amikor kimarad egy ACK a címzettől. A feladó küldi a csomagokat, a címzett makacsul nem ackol, aztán feltelik a küldő oldali puffer, végül megmerevednek a frontok. Nagy sokára letelik az ACK várására fenntartott idő, a küldő újra elküldi a le nem ackolt csomagokat.
De mi van akkor ha gigabites vonalon nyomtuk, jó nagy ablakméretekkel (long fat pipe)? Küldjük újra az összes csomagot? Amikor - láttuk - az ablak akár 1 GB méretű is lehet?

Ilyen nagy ablakméreteknél már érdemes használni a selective ACK, azaz SACK opciókat. A többes szám nem véletlen, ilyen opcióból ugyanis kettő is van:

- SACK-Permitted Option (Option Kind=4)
- SACK Option (Option Kind=5)

A SACK-Permitted opció jelzi a partnernek, hogy tudom kezelni, ha SACK visszajelzést kapok. Ezt az információt szintén a SYN csomagokban egyeztetik a felek.

A SACK opcióban pedig konkrét számok utaznak. Itt szerepelnek azok a Sequence Number értékek, melyeket a fogadó nem kapott meg. Ilyenkor a feladó csak ezeket küldi újra.

5.2.2.5 TCP TIMESTAMPS OPTION

Rögtön vezessünk be két fogalmat:

- **Retransmission Time-Out** (RTO): Az az idő, melynek letelte után, ha nem érkezett visszajelzés, akkor újraküldjük a csomagot.
- **Round-Trip Time** (RTT): A csomag elküldése és a visszajelzés megérkezése közti időtartam.

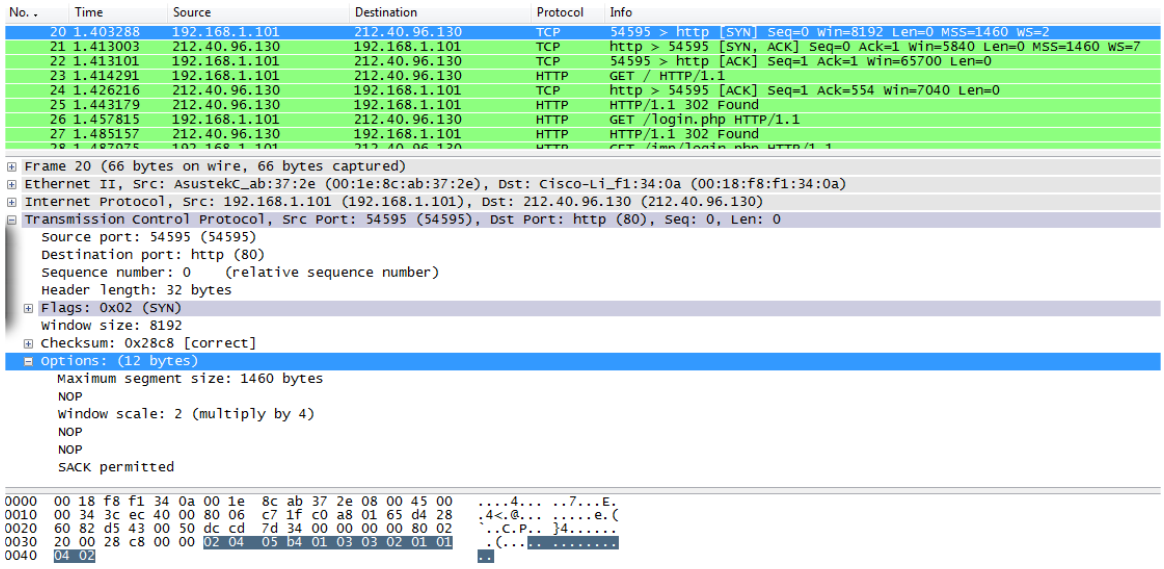
Az RTO finomhangolása érdekében a TCP folyamatosan figyeli az RTT alakulását. A "folyamatosan" kifejezést persze nem kell szószerint érteni, általában egy Send Window-n belül egy RTT mérés történik.

Akadnak olyan kiszámíthatatlanul ingadozó hálózatok, ahol ez kevés. Ekkor kell rákényszeríteni a TCP-t, hogy sűrűbben számolgasson RTT-t. Erre jó a TCP Timestamp opció. (Option Kind=8)

A játékban mindkét félnek részt kell vennie. Bármelyik is küld egy csomagot, a TCP a helyi gépidőből kikalkulál egy értéket (TS Value) és belerakja a TCP Timestamp opció adatmezőjébe, mellé pedig belerakja, hogy mennyi volt a másik félre vonatkozó legutóbbi időérték (TS Echo Reply), azaz az a TS Value, amit legutoljára a másik féltől kapott.

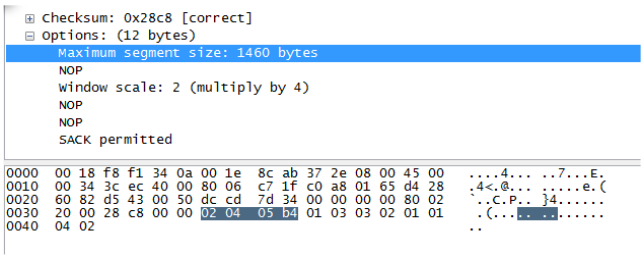
5.2.2.6 PÉLDÁK

Nézzük meg, hogyan is néznek ki a TCP opciók a valóságban.



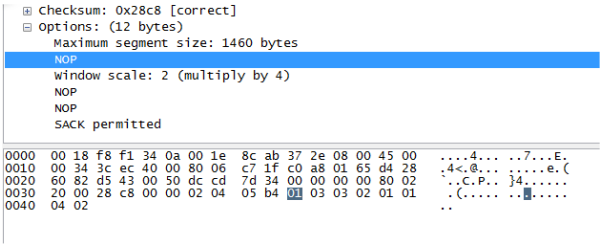
5.34. ÁBRA TCP OPTIONS - SYN

Ez a korábban már kitárgyalt folyamat első csomagja, az, amelyikben a kliens a webszerverhez fordul. A SYN flag szemmel láthatóan magas - és mint utaltam is rá, ebben az első csomagban közöl a kliens a TCP opciókon keresztül egy csomó fontos adatot is a szerverrel.



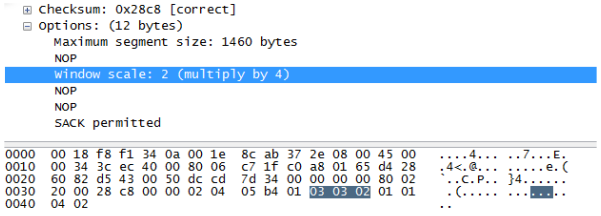
5.35. ÁBRA TCP OPTIONS - MSS

Koncentráljunk a Packet Bytes mezőre. Látható, hogy az OPTION Kind 2, az opció mérete 4 bájt (stimmel), a két utolsó bájt pedig az MSS értéke hexadecimális formában.



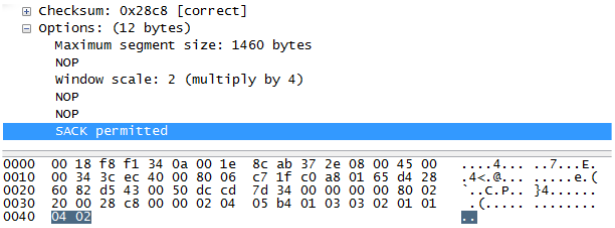
5.36. ÁBRA TCP OPTIONS - NOOP

A jó öreg No Operation TCP opció. Mondtuk róla, hogy elválaszt (stimmel), mondtuk róla, hogy 1 bájt és mondtuk róla, hogy az értéke 1.



5.37. ÁBRA TCP OPTIONS - WINDOWS SCALE

TCP Window Scale opció. Az Option Kind értéke 3, az opció mérete 3 bájt, az adat pedig 2. Mennyi is akkor a kliensem maximális fogadó ablaka? $2^{16} * 2^2 = 2^{18} = 256$ KB.

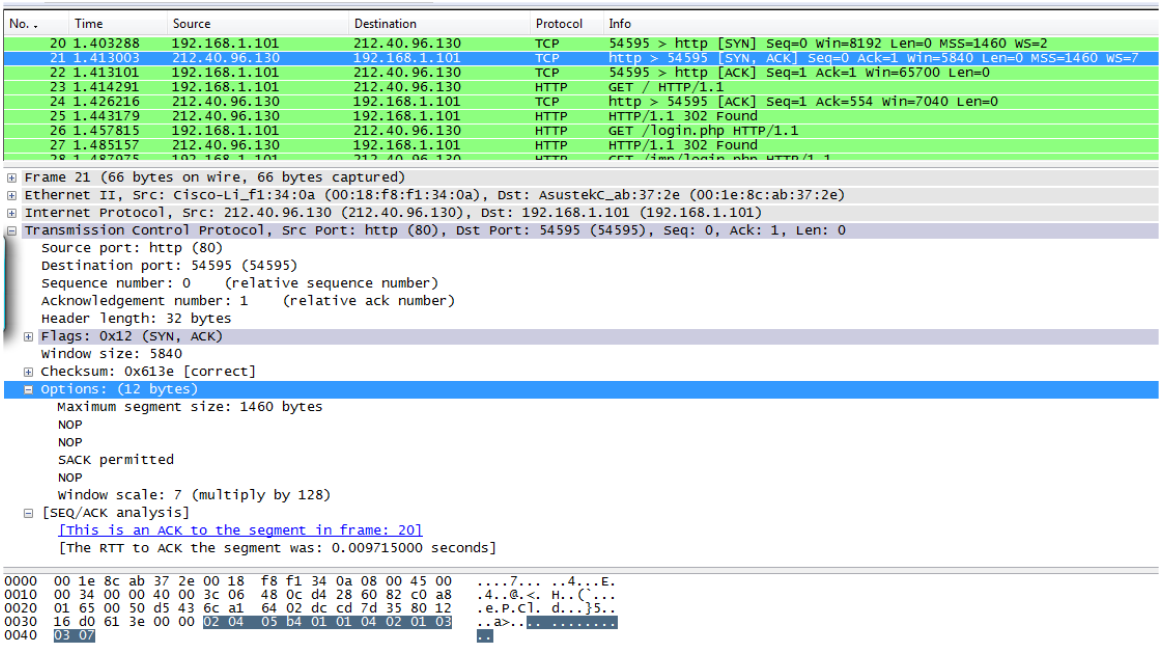


5.38. ÁBRA TCP OPTIONS - SACK

SACK-Permitted opció. Az Option Kind értéke 4, az opció mérete 2 bájt. Adattartalma nincsen, hiszen ez csak jelzi, hogy képes értelmezni a szelektív ACK-ot.

És ennyi. Micsoda? Hogy hiányzik valami? Nem látod az End Of Option List opciót? Hát, azt bizony nem.
Nagyon trükkösen lett megkerülve. Talán feltűnt az ábrán (*5.34. ábra TCP Options - SYN*), hogy egymás után két NOP is jön. Ez a duplázás kellett ahhoz, hogy a TCP opciók által elfoglalt méret néggyel osztható legyen. Ilyenkor nem kell külön lezárni. (Miért fontos a néggyel oszthatóság? Azért, mert a header méretét szószámban tároljuk - lásd a DATA OFFSET mező részletezését.)

Persze a kommunikáció nem csak ennyiből áll: a szerver válaszában is magas a SYN flag, tehát ő is elküldi magáról a fontosabb értékeket.



5.39. ÁBRA TCP OPTIONS - SYN-ACK

Ezt már nem részletezném annyira mélyen. Maximum arra hívnám fel a figyelmet, hogy ebben a csomagban már magas az ACK flag, durván be is jött annyi csomag, amennyi egy Send Window-ban elfér - tehát tudunk RTT értéket visszaküldeni.

5.2.3 TCP FLAG-EK

Kutyafuttában már esett róluk szó, de most menjünk végig részletesebben is a választékon. Azt láttuk, hogy 1 bájt helyet foglalnak - ebből a fejszámoló olvasók már ki is találhatták, hogy nyolc darab flag-ről lesz szó.
Sorrendben:

CWR | ECE | URG | ACK | PSH | RST | SYN | FIN

ECE, AZAZ ECN-ECHO: Habár a második a sorban, de érdemes mégis ezzel kezdeni. Azt jelenti, hogy a küldő ismeri az ECN-t, illetve akkor is magas lesz, ha az ECN ténylegesen be is következik - azaz az IP headerben bejelez az ECN mező. (ECN: Explicit Congestion Notification, azaz határozott visszajelzés túlterhelésről. Ha esetleg nem ugrana be: *4.1.1 Az IP Header*.)

RFC 3168

A TCP opciók egy részéhez hasonlóan az ECN, mint ismert képesség szintén kapcsolatfelvételkor (magas SYN flag) megy át.

CWR, AZAZ CONGESTION WINDOW REDUCED: A feladó visszajelzése, hogy vette az ECE jelzést, le fogja csökkenteni a congestion window méretét. (Később lesz róla szó.)

URG, AZAZ URGENT: Jelzi, hogy a szegmensben sürgős anyag van, tessék komolyan venni az URGENT POINTER értékét.

ACK, AZAZ ACKNOWLEDGMENT: Jelzi, hogy az Acknowledgment mezőbe írt szám nem viccből van ott. (A gyakorlatban az ACK flag szinte mindig magas.)

PSH, AZAZ PUSH: Mit is írtam a megérkezett csomagokról? Mi is történik akkor, amikor megérkezik egy csomag a címzetthez? A TCP rögtön felküldi az alkalmazás rétegnek? Ugyan, nem UDP ez. Szépen gyűjtögeti az adatokat a receive pufferben. Aztán amikor a puffer megtelt, vagy éppen a kapcsolatkezelő eljárás úgy érzi, hogy itt az idő, akkor a köztes tárolóból átkerül az adat az alkalmazáshoz. (Feltéve, hogy nincs hiányzó csomag a sorban.)
Ezt a folyamatot rúgja fel a PSH flag. Ha ilyen csomag érkezik, az olyan, mint HKSz riadó a seregben: mindenki összekapkodja a harci felszerelését és kirohan a szobából. (A 'ne legyen hiányzó csomag' kritérium ettől függetlenül itt is él.)

RST, AZAZ RESET: Köszönjük, ennyi. A kapcsolatnak vége. Méghozzá nem is barátságos módon - az RST sokkal inkább erőszakos bontást jelent. Mind a send, mind a receive pufferből törlődni fognak a kapcsolat adatai. Ugyanezt kapjuk, ha egy kapcsolat már eleve létre sem jött. (Tipikusan amikor egy tűzfal vágja pofán a bimbózó kapcsolatunkat.)

SYN, AZAZ SYNCHRONIZE: Jelzi, hogy egy kapcsolat van kialakulófélben. Amikor ez a flag magas, akkor küldi át az egyik fél a kapcsolat létrehozásához szükséges adatokat a másiknak.

FIN, AZAZ FINISH: Ennek a kapcsolatnak is vége - de azért barátok maradtunk. A kapcsolat normálisan csak akkor szűnhet meg, ha mindkét fél elküldte az elbocsátó szép üzenetet (FIN), majd mind a kettő vissza is igazolta.

5.2.4 TCP KAPCSOLATOK KEZELÉSE

Egy kapcsolatnak alapvetően három fázisa lehet:

- Bimbózó
- Stabil
- Szakító

Jelzem, ez egy nagyon rövid fejezet lesz, hiszen korábban már jócskán kiveséztük a folyamatot. (*5.2.1 A Sequence Number és az Acknowledgment Number pingpongcsata*)

5.2.4.1 EGY TCP KAPCSOLAT KIALAKULÁSA

Amiről ott nem írtam: milyen információk utaznak még a SYN-nel jelzett csomagban?

- Először is a feladó ISN-je. (Ezt tudtuk.)
- A kezdeményező receive windows értéke.
- A kezdeményező MSS (Maximum Segment Size) értéke.
- A lista, hogy a kezdeményező milyen TCP opciókat támogat.

Pusztán csak ismétlésként jegyzem meg, hogy ilyen értelemben a SYN kapcsolatfelvételre jövő SYN-ACK válasz is SYN csomagnak számít: azaz a válaszoló is közli mindezen információkat magáról a háromlépéses kézfogás második lépésében.

Hogy mégse legyen olyan rövid ez a fejezet, meg legyen egy kis pihenő is a száraz tananyagban, említsük meg a SYN Attack néven ismeretes támadási formát. Nem, nem kell idegesen körbenézned, eszembe sem jut megrontani az ifjúságot. A SYN Attack annyira öreg dolog, hogy ma már minden normálisabb rendszer le tudja kezelni.

Szóval. Elindul egy kapcsolat. A kezdeményező küld egy SYN csomagot. A szerver boldogan válaszol egy SYN-ACK csomaggal. A kezdeményező a nyugtázó ACK helyett viszont egy újabb SYN csomagot küld, egy másik portra. A szerver, ha lehet, még boldogabb. Micsoda népszerűség! - gondolja. Természetesen erre is válaszol egy SYN-ACK csomaggal. És így tovább. Gondolom, látható a minta. Aztán mivel a szerver a le nem zárt kapcsolatokat elrakja a memóriájába, előbb-utóbb az betelik. A szerver nyáladzó mosollyal a szája szélén csuklik össze.

5.2.4.2 EGY TCP KAPCSOLAT FENNTARTÁSA

Ha éppen nincs forgalom egy kapcsolaton keresztül, de az nem lett se békésen, se harcosan lezárva, akkor valószínűleg a feleknek még szükségük lesz rá. Ilyenkör jön az, hogy üres ACK csomagokkal jeleznek egymásnak, nehogy azt higyje a másik, hogy valahol elrágta a macska a kábelt.

Persze ez sem ilyen egyszerű. Mi történik közben az SN/ACK számokkal?

Az, hogy aki a keep-alive ACK csomagot küldi, eggyel kisebb SN értéket ír bele, mint amekkorára a másik fél - az általa korábban küldött ACK alapján - számít. A fogadó veszi a lapot, érzékeli, hogy ez csak keep-alive, visszaküldi a korábbi ACK-ot - és most már azt is fogja visszakapni a következő lépésben.

- Miről prédikált a Tisztelendő Atya?
- A bűnről.
- És mit mondott róla?
- Ellenezte.

Azaz eddig írtam a keep-alive folyamatról. Arról a folyamatról, melyet a Windows Server 2008 / Vista alapból tilt[82]. Mert biztos szép és biztos jó dolgok ezek, de leginkább úri huncutságnak tűnnek.
Hadd hulljon a férgese.

[82] Engedélyezni Winsock API függvényhívásból lehet.

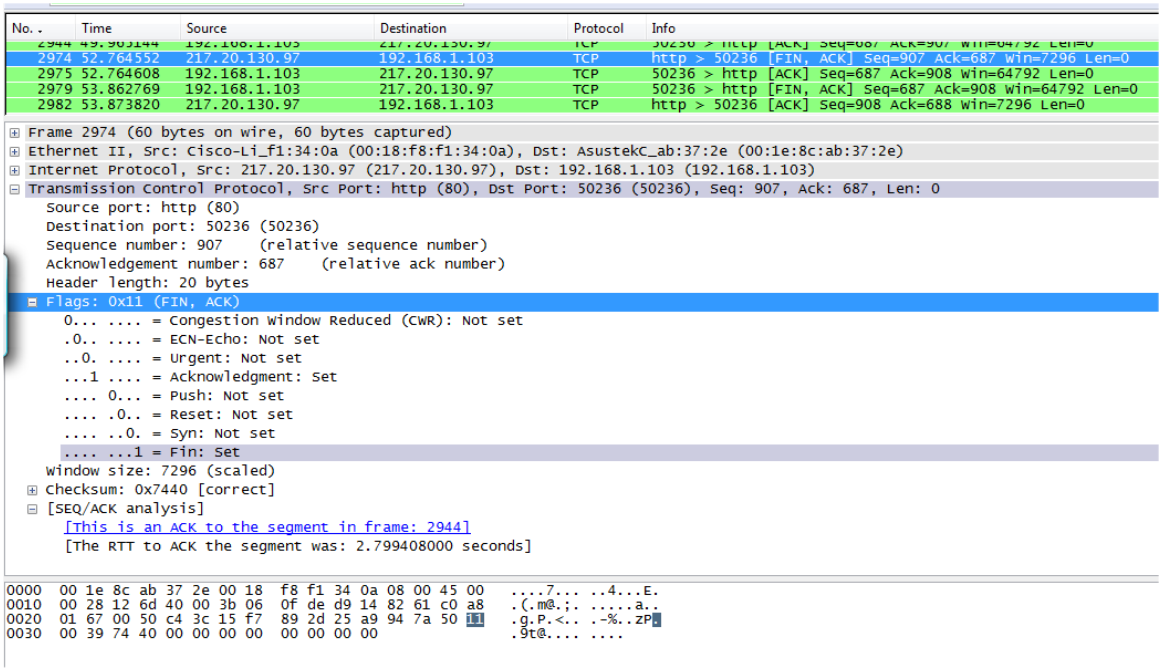
5.2.4.3 EGY TCP KAPCSOLAT FELBOMLÁSA

5.2.4.3.1 A BÉKÉSEBB VERZIÓ: FIN

A FIN flag szerepe nagyban hasonlít a SYN flag szerepére. Míg a SYN kapcsolatfelvételi szándékot jelez, addig a FIN éppen ellenkezőleg: kapcsolatbontási szándékot. Csakhogy itt a kézrázásnak nem három fázisa van, hanem négy:

- A bontani szándékozó jelez: FIN_1-ACK_1
- A partnere visszaigazol: ACK_2
- A partner is lebontja a csatornát: FIN_3-ACK_3
- A kezdeményező visszaigazolja: ACK_4

A fenti folyamatot illusztrálja az alábbi ábra.

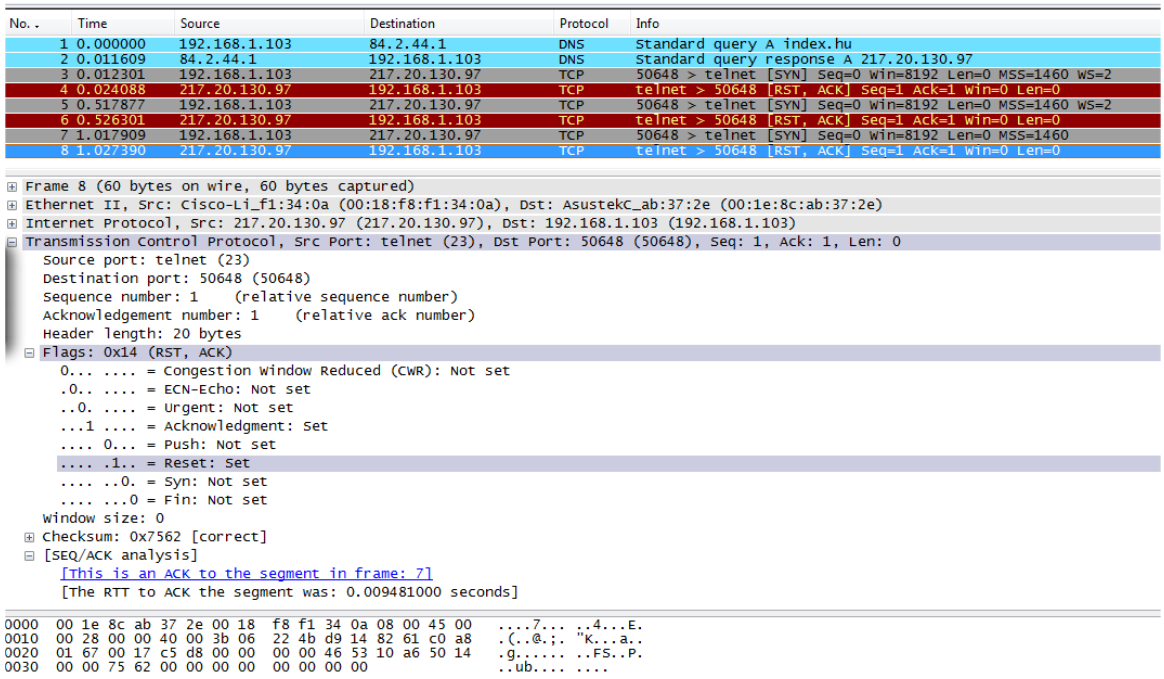


5.40. ÁBRA EGY KAPCSOLAT BÉKÉS VÉGE

Nagyon szépen látszanak az SN/ACK számokkal történő játékok is.

5.2.4.3.2 A VADABB VERZIÓ: RST

Nagy pofont szeretnénk? Telneteljünk rá az index.hu-ra!



5.41. ÁBRA EGY KAPCSOLAT DURVA VÉGE

Szemmel láthatóan nem örültek nekünk. Meg szerettük volna rázni a kezüket, olyan jó háromlépéses formában - de már a kinyújtott kezünket sem fogadták el. Elküldtük a SYN csomagunkat, úgy ahogy kell (SN_{rel}=0, RecWin=8192, szegmensméret=0, MSS=1460 és Window Scale=2) - de csak egy zord RST-ACK jött vissza.
Mivel nagyapánktól azt tanultuk, hogy ne üljünk föl olyan szekérre, ahol nem látnak minket szívesen, tudomásul vettük az RST-t - három próbálkozás után abbahagytuk.

RST-t nem csak kapcsolat létrehozásakor kaphatunk. Ugyanide jutunk, ha egy már meglévő kapcsolaton keresztüli forgalomban értelmezhetetlen paramétert talál a TCP fejlécben valamelyik host. Ilyenkor a kapcsolat összes adata elvész, melyeket az alkalmazás még nem olvasott fel a receive pufferből.

5.2.4.4 A TCP KAPCSOLATOK ÁLLAPOTAI

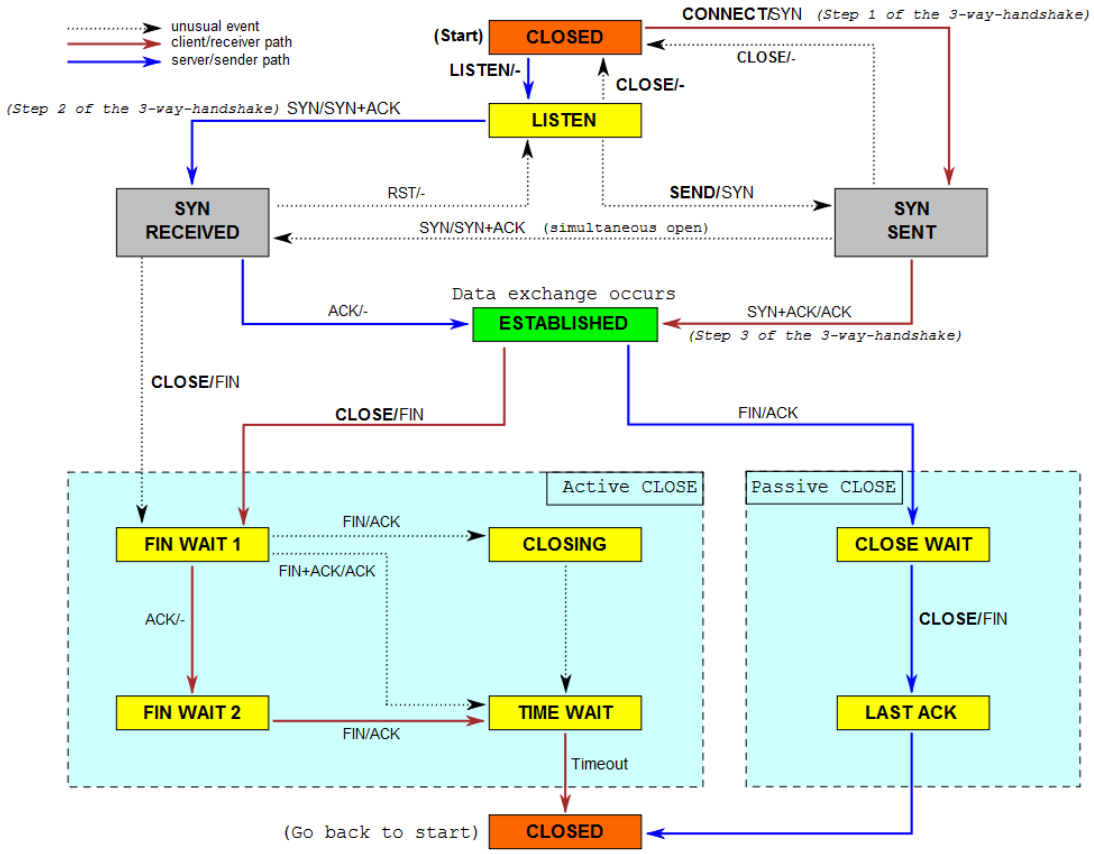
A TCP kapcsolatok egyes lépéseit állapotoknak nevezzük. Ezeket az állapotokat tartalmazza az alábbi táblázat:

5.2. TÁBLÁZAT

Állapot	Leírás
CLOSED	Az égegyadta világon nem történik semmi.
LISTEN	Egy alkalmazás szintű protokoll elkezdett figyelni egy porton.
SYN SENT	Egy alkalmazás szintű protokoll kapcsolatot kezdeményezett, azaz elküldött egy SYN csomagot.
SYN RCVD	A SYN csomag megérkezett, ment is vissza a SYN-ACK.
ESTABLISHED	Megérkezett a SYN-ACK utáni ACK, a kapcsolat létrejött.
FIN WAIT-1	A kapcsolat bontását kezdeményező első FIN-ACK csomag elment.
FIN-WAIT-2	Megjött az ACK a kezdeményező FIN-ACK csomagra.
CLOSING	Ez egy ilyen lebegő állapot, megkaptuk a FIN-ACK csomagot, de még nem küldtük el az ACK-et.
TIME WAIT	Mindenki megkapott minden csomagot, mely a kapcsolat kétirányú lezárásához szükséges volt. Ekkor még eltelik egy bizonyos idő, amíg új kapcsolatot lehet nyitni.
CLOSE WAIT	Megkaptuk a FIN-ACK csomagot és küldünk is egy FIN-ACK csomagot.
LAST ACK	Megkaptuk az ACK csomagot a FIN-ACK csomagunkra.

Akárhogy is nézem, nem teljesen kerek a történet. Például mi lehet a különbség a LAST ACK és a FIN-WAIT2 állapotok között? A táblázat alapján például semmi.

Ezért van itt ez az ábra.



5.42. ÁBRA EGY KAPCSOLAT ÁLLAPOTAINAK TÉRKÉPE (WIKIPEDIA)

Mint látható, a lezárásnál van egy aktív folyamat és van egy passzív. Az aktív folyamat azokból az állapotokból áll, amelyeken a bontást kezdeményező fél megy át, míg a passzív folyamat azokból az állapotokból, amelyeken a partner. Ergo a FIN WAIT-2 állapotban a bontást kezdeményező fél kapja meg az ACK csomagot a FIN-ACK kérésére, míg a LAST ACK állapotban a túloldali fél kapja meg az ACK-et az ő FIN-ACK csomagjára válaszul.

5.2.5 TCP ADATFOLYAM

Tehát, a bitek folynak - mint azt korábban már tisztáztuk. Jönnek befelé, gyűlnek a receive pufferben, majd az alkalmazás felkapkodja. A másik oldalon az alkalmazás tolja bele a send pufferbe, ahonnan a TCP talicskázza kifelé.
De egész pontosan, milyen állapotai is lehetnek egy csomagnak? Hogyan viszonyulnak ehhez a Send/Receive ablakok? És miért áll meg a közlekedés nálunk, ha leesik egy centi hó?
Habár az eddig leírtakból - majdnem - minderre meg lehet találni a választ, úgy gondolom, egy grafikus szemléltetés mindig jól jöhet[83].

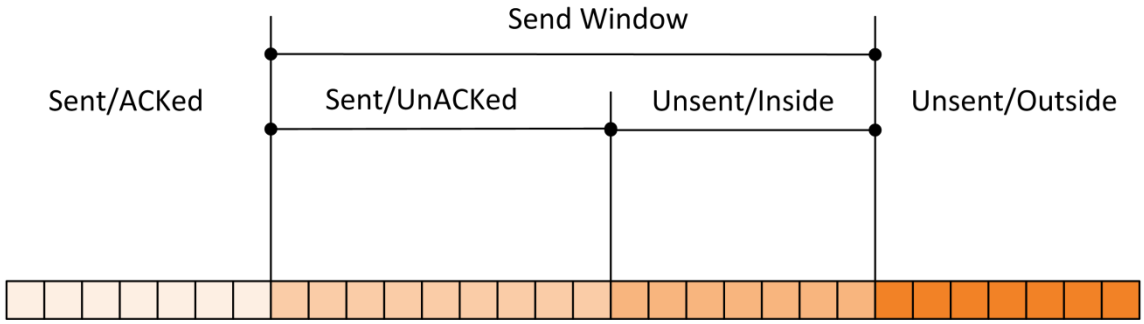
5.2.5.1 SEND WINDOW

Igen, köszönöm, vettem a kérdésedet. Mi is az a Send Window, amelyről eddig csak úgy itt-ott elpötyögtetve esett szó?
Beszéljünk inkább először a Receive Window-ról, mert azt már ismerjük. Ez az az érték, melyet a fogadó fél minden ACK csomagban elküld a feladónak. Ebben mondja meg, hogy barátom, momentán ennyi adatot tudok fogadni egyszerre, ennél több pillanatnyilag nem fér be a szobába[84]. Ad abszurdum, ha a fogadó 0 méretű Receive Window értéket jelez vissza, a feladó teljesen le is áll - addig, amíg nem kap egy nullánál nagyobb értéket.

Most, miután a Send Window fejezetben ilyen jól kitárgyaltuk a Receive Window jelentését, nem ártana magával a Send Window-val is foglalkozni. Nos, a helyzet az, hogy a Send Window a Receive Window hű társa. A teleport állomás, ahonnan a csomagok indulnak a címzett teleport állomásába. Ha nincs forgalom a levegőben - és már korábban beindult a kapcsolat - akkor a Receive és a Send Window mérete megegyezik. Gyakorlatilag a feladó a Send Window-ban adminisztrálja, mi is a helyzet a feladott csomagokkal, a fogadó pedig a Receive Window-ban követi nyomon, hogyan állnak a fogadott csomagok.

Szokták úgy is meghatározni a Send Window-t, hogy az az ablakméret, amíg küldhetünk csomagokat, anélkül, hogy meg kellene várnunk a visszaigazolást. Azaz küldünk, küldünk, küldünk... és ahogy jönnek vissza a visszaigazolások, úgy lépünk egyet-egyet előre az ablakkal. Ha nem jönnek a visszaigazolások, akkor betelik az ablak és megáll - egészen addig, amíg le nem telik az idő és újra nem kezdjük küldeni a vissza nem igazolt szegmenseket.

[83] Különben is, mérnökember mindig rajzol. © Vegyipari Gépészek Kórusa, Veszprém.
[84] My hovercraft is full of eels. (http://www.omniglot.com/language/phrases/hovercraft.htm)



5.43. ÁBRA SEND WINDOW

Mint az ábrán is látható, négy kategóriába soroltuk a csomagokat:

Sent/ACKed: Ezek azok a csomagok, amelyekkel a továbbiakban az égegyadta világon semmi dolgunk sincs. Elküldtük őket, megkaptuk a visszajelzést, a csomagok kicsúsznak a balfenéken.

Sent/UnACKed: Beléptünk a Send Window területére. Itt vannak a már elküldött, de még vissza nem igazolt csomagok.

Unsent/Inside: Ez az a terület, ahová az alkalmazás már lepakolta a csomagokat, azok már bekerültek az elküldendők közé (inside), de még nem lettek elküldve.

Unsent/Outside: Ezek azok a csomagok, melyek csak álmodoznak róla, hogy el lesznek küldve - de egyelőre még csak sorbaállnak a teleport állomás előtt.

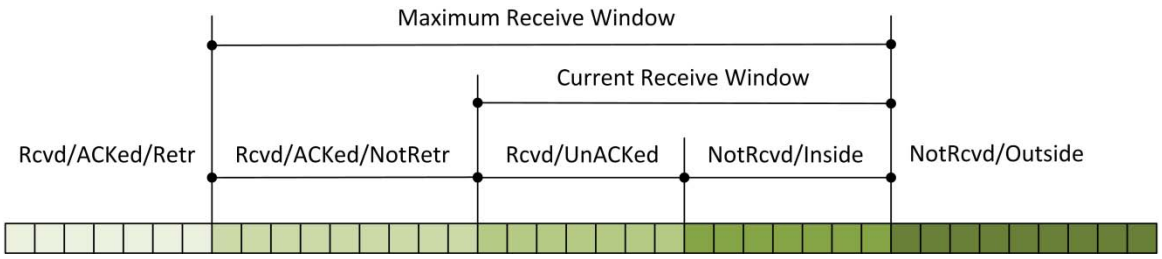
Az már csak szemlélet kérdése, ki hogyan képzeli el:

- Vagy az ablak áll egy helyben és a csomagok vándorolnak jobbról balra.
- Vagy a csomagok állnak egy helyben és az ablak - birodalmi lépegetőként - halad balról jobbra.

Az utóbbi hasonlat már csak azért is szemléletes, mivel tudjuk, hogy az ablak mérete nem állandó. Ez egy imbolygó, dűlöngélő ablak.

5.2.5.2 Receive Window

Most pedig átszökkenünk a fogadó oldalra.



5.44. ábra Receive Window

Received/ACKed/Retrieved: Baloldalt megint azokat a csomagokat látjuk, akik már túl vannak a dolgon. A csomagok megérkeztek, a visszaigazolás elment, a fogadó alkalmazás felkapkodta a dobozokat.

Received/ACKed/Not Retreived: Megérkeztek, visszaigazoltuk - de az alkalmazás még nem ugrott be a csomagokért.

Received/UnACKed: Megérkeztek, de még nem igazoltuk vissza.

Not Received/Inside: Ez egy trükkös meghatározása a semminek. Azt mondja, hogy még az ablakon belül vagyunk, de ide még nem érkezett semmi. Azaz a teleport állomáson az üres üvegcsövek. Ide várjuk a csomagokat feldolgozásra.

Not Received/Outside: A csomagok, melyek majd valamikor meg fognak jönni. Jelenlegi tartózkodási helyük: földön, vízen, levegőben.

Nyilván észrevetted. Itt öt részre osztottuk fel a futószalagot, miközben a Send Window esetében elég volt négy rész. Gondolom, azt is látod, hogy a kavarást a Maximum Receive Window okozza. Mi is ez - és miért van szükség két ablakra?
A megoldás kulcsa a billegés. A Send Window-nál mondtam, hogy állandóan változik a mérete, azaz lépegetőként billeg. Itt ez a viselkedés a Current Receive Window-ra igaz. Ez ugyanis a pillanatnyi méretét jelenti a fogadó ablaknak. (Ezt a pillanatnyi értékét küldi vissza az ACK csomagban.) A Maximum Receive Window mérete ezzel szemben egy fix érték... a maximum, melyet a fogadó egyáltalán be tud fogadni. (Ha a Current Receive Window mérete nem éri el a maximumot, logikus, hogy a feldolgozás utolsó fázisában lévő csomagok helyezkedjenek el a kettő közötti területen.)

Ezt a két, csúszkaként mozgó ablakot (Send/Receive) nevezzük összefoglaló néven TCP Sliding Windows-nak.

Írtam a grafikus szemléltetés hasznosságáról. Próbáltam rajzolgatni is, de ezt a folyamatot igazából egy animáció mutathatná be a legjobban.
Szerencsére van is ilyen a neten.

http://www.osischool.com/protocol/Tcp/slidingWindow/index.php

Szívem szerint bele is illeszteném ebbe a könyvbe, annyira jó. Habár az animáció elsődleges célja a később tárgyalandó Slow Start algoritmus bemutatása, de remekül látszik, hogyan dolgozik párhuzamosan a két TCP Sliding Window. (Egy apró nomenklatúra pontosítás. Az én fogalmaim szerint mind a két ablak - Send Window, Receive Window - a TCP Sliding Windows kategóriába esik, addig az animáció a Receive Window-t nevezi Advertised Window-nak, a Send Window-t meg Sliding Window-nak.)

Illetve érdemes elcsemegézni az oldal többi animációi között is.
http://www.osischool.com/protocol/Tcp/

Jó kérdés lehet, hogy a Receive Window most akkor egy fix méret, vagy dinamikusan változó? Mikor hogy. Valamikor fix érték volt (a Windowsban nyilván registryben tárolva), de a Windows Server 2008 / Vista esetében már úgynevezett autotuning hangolja, a konkrét vonal sávszélességének és forgalmának függvényében. (Emiatt szokták felhívni a figyelmet arra, hogy a fenti operációs rendszereknél tényleg figyeljünk oda a QOS-re: ugyanis innentől a TCP sokkal hatékonyabban tömi ki a drótot - azaz ha van valami alkalmazás, mely fix, minimális sávszéleség igénnyel bír, akkor ezt foglaljuk is le a számára. Hulladékként már nem fog neki annyi megmaradni, mint korábban.)

5.2.6 AZ ÚJRAKÜLDÉSEK RENDSZERE

A TCP működése eddig teljesen logikus volt, még ha egy kicsit bonyolult is.
Na, ez változik meg innentől.
Nem, nem a logikával lesz baj. És egyszerűsödni sem fog a helyzet.

Azt állítottam a TCP-ről, hogy megbízható. Ez bizony nem csak abból áll, hogy sorszámozza a csomagokat és ellenőrzi a megérkezésüket: szükség esetén újra is kell küldenie az elveszett csomagokat.

Istenem, milyen szép dolog is a nyelv. A fenti mondat úgy helyes, ahogy van, mindenki bólogat - pedig két helyen is bele van kódolva a rejtett akna: mi az, hogy '*szükség esetén*'... és mi az, hogy '*elveszett*'?

A TCP kulcskérdéséhez jutottunk el. Mikor küldjünk újra egy csomagot?

- Ha túl sokáig várunk, akkor lassú lesz az átvitel - hiszen ezt az időt minden visszaigazolás előtt ki kell várni.
- Ha túl hamar újraküldjük, akkor borzasztóan rossz lesz a hatékonyság - hiszen anyag megy ugyan át a kapcsolaton, csak sokszor ugyanaz még egyszer.

Igen érzékeny optimumot kell eltalálni - ráadásul szédületesen változó környezetekben. Egyáltalán nem mindegy, hogy gigabites LAN-on vagyunk (ahol minden belefér), műholdas WAN-on (ahol nagy a sávszélesség, de a hiba is rengeteg) vagy éppen egy betárcsázós ISDN-en (ahol nyüszítünk minden egyes kbps-ért). Emellett egyáltalán nem mindegy, hogy a kommunikáló felek a TCP mely huncutságait ismerik és melyeket nem.

Rengeteg trükk, apró ésszerűsítés létezik a TCP-ben, hogy a működése mindenhol optimum közelében legyen. Nem fogom mindegyiket bemutatni ebben a fejezetben, inkább csak mazsolázgatok.

Először is, ismerkedjünk meg az újraküldés alapelemeivel. Az RTO/RTT párosról már ejtettem pár szót, de igazából ebben a fejezetben érzik magukat otthon.

- RTT (Round Trip Time): Az az idő, mely a csomag feladása és a visszaigazolás megérkezése között telik el. Ezt a TCP folyamatosan méri.
- RTO (Retransmission Time Out): Az az időkorlát, mely után egy csomag elveszettnek tekintendő, azaz a feladó újraküldi. A TCP bizonyos rendszerességgel az RTT értékekből számolja ki.

Jó. És hogyan?

RFC 793

Az eredeti RFC szerint így:

Measure the elapsed time between sending a data octet with a particular sequence number and receiving an acknowledgment that covers that sequence number (segments sent do not have to match segments received).
This measured elapsed time is the Round Trip Time (RTT). Next compute a Smoothed Round Trip Time (SRTT) as:

SRTT = (ALPHA * SRTT) + ((1-ALPHA) * RTT)

and based on this, compute the retransmission timeout (RTO) as:

RTO = min[UBOUND,max[LBOUND,(BETA*SRTT)]]

where UBOUND is an upper bound on the timeout (e.g., 1 minute), LBOUND is a lower bound on the timeout (e.g., 1 second), ALPHA is a smoothing factor (e.g., .8 to .9), and BETA is a delay variance factor (e.g., 1.3 to 2.0).

Ez egy szép képlet, kár, hogy a gyakorlatban nem vált be.

RFC 1122

Jött helyette egy másik, mely az RFC 1122-ben lett leírva. Ez egy nagyon szép, több oldalas eljárás, feltételekkel, kiegészítésekkel bőven megtámogatva. Ha nem haragszol, mélyebben nem mennék bele.

Ugyanis ezzel még nincs vége. A fenti eljárás remekül működik akkor, ha alapvetően kicsik az ablak méretek. De a gigabites vonalakon nem ez a jellemző. Viszont ha nagyok az ablakok, akkor kevés az ACK - azaz nem mérvadó az RTT mérése sem. Ekkor jön be a képbe a korábban már említett TCP Timestamp TCP opció. (Csak ismétlésképpen: az opció adatterületén belül mindkét host elküldi a saját belső órájának az értékét.) Ilyen esetekben az RTO-t már nem az RTT-ből számolják a felek, hanem ezekből a belső órák által mutatott értékekből

Szép, mi? És ez még csak az RTO számítás. Hogy a felek mely RFC-ket ismernek? Egyáltalán, mely TCP opciókat? Az eszközöknek nincs más választása: bíznak... és próbálkoznak.

Akkor jöjjenek a különböző megfontolások, illetve trükkök.

TCP Tuning:
http://en.wikipedia.org/wiki/TCP_tuning

Hogyan veszhetnek el útközben csomagok? Elméletileg nem kizárt, hogy éppen arra járt a baltás gyilkos és apróra hasította az ethernet kábelt - de a gyakorlatban inkább a túlterhelt routerek szokták eldobálni a csomagjainkat. Nyilván visszajelzés nélkül, hiszen túlterheltek. Hogyan védekezhetünk ez ellen? Ha felismernénk, hogy router túlterhelésről van szó, akkor például megnövelhetnénk az RTO-t. Ez tulajdonképpen automatikus is, hiszen megnő az RTT, tehát meg fog nőni az RTO. Fog. Csak éppen pár tempóval később, hiszen az RTO a korábbi RTT-kből áll össze. Azaz már nagy az RTT, de az RTO még a korábbi, kisebb RTT-k alapján áll össze -> hirtelen nagyon megugrik az eldobott csomagok száma -> hirtelen beindul egy nagyobb lélegzetű újraküldés. Mindez akkor, amikor a router egyébként is be van havazva.

5.2.6.1 Slow Start algoritmus

RFC 2581, 3465

Erre a problémára az egyik válasz az ún. Slow Start algoritmus. (A Windows Server 2003 / XP a korábbi RFC-t ismeri, a Windows Server 2008 / Vista a későbbit.)
Ehhez bevezetjük az ún. Congestion Window (cwind) változó fogalmát. Gyakorlatilag ezzel a Send Window méretét befolyásoljuk: amennyiben a cwind értéke kisebb, mint az aktuális Receive Window értéke, akkor a feladó inkább a cwind-et használja a Send Window méretének beállításához.
Így finoman tudjuk szabályozni egy kapcsolat beindulását, illetve felpörgetését. Elsőre a cwind-et jóval kisebbre szokták venni, mint a Receive Window. (Tipikus érték: MSS*2.) Ha nem veszett el a csomag, jöhet a duplázás: MSS*4. És így tovább, amíg el nem érjük a Receive Window értékét.

Utalnék megint a korábbi animációra:
http://www.osischool.com/protocol/Tcp/slidingWindow/index.php

5.2.6.2 Congestion Avoidence algoritmus

Kicsit hasonló az előző algoritmushoz, ugyanúgy a cwind változóval operál. Azt mondja, hogy amikor elveszik egy csomag - azaz a feladó nem kapja meg a visszajelzést, akkor a cwind értékét beállítja a pillanatnyi Receive Window értékének a felére, majd nekiáll szép fokozatosan növelni azt. (Lépésenként 1 MSS értékkel.)

Vegyük észre, hogy a két eljárás közötti különbség gyakorlatilag csak a kezdőpont kiválasztása és a növekedés mértéke. Hogy a kettő közül melyiket fogja használni a TCP, az egy sstresh nevű változó értékétől függ - de ennyire mélyen már nem szándékozom belemenni a témába.

5.2.6.3 Az újraküldés lélektana

Tegyük fel, elveszett egy csomagunk. Letelt a várakozási idő, és mi, mint feladók nem kaptuk meg a visszaigazolást. Nyilván újból el fogjuk küldeni. Hányszor? Még egyszer egészen biztosan. És ha arról sem jön meg a visszaigazolás? Türelmes emberek vagyunk, egyszer még megpróbáljuk. Ha arról sem jön meg az ACK, akkor hagyunk fel csak a küldéssel.
Mennyi idő teljen el két küldés közben? Ez nyilván az RTO-tól függ, hiszen annyit várunk, amíg a csomagot elveszettnek nem nyilvánítjuk. Jó nekünk az, ha mindig ugyanannyi RTO-val számolunk? Nem. Az alkalmazott logika az, hogy az újraküldéseknél megduplázzuk a korábbi RTO-t. Konkrétan Windows Server 2008 esetén a tipikus példa kapcsolat felépítésekor: kezdő RTO 3 sec, újranyitás 6 sec, utolsó próbálkozás 12 sec. Azaz egy nem működő kapcsolat felismerése 21 sec összesen.

5.2.6.4 Dead Gateway felismerés

Az előbb nagyon gyorsan lezártuk a szálat, amikor azt mondtuk, hogy abbahagyja a küldést - és kész. Nem egészen.
Ha ugyanis a küldési irány a default gateway felé mutatott, és így veszett el - egymás után háromszor - a csomag, akkor a host úgy fogja kezelni, hogy ez a default gateway megdöglött. És igen, most jön elő az, hogy miért lehet egy hálózati kártya konfigurációjánál több DGW-t is megadni. Nem, nem azért, hogy az egyik irányhoz erre, a másik irányhoz meg amarra menjenek a csomagok[85]. Arra a route tábla való.

[85] MCP vizsgák egyik beugrató kérdése.

Ellenben, ha több DGW-t veszünk fel, akkor ilyenkor, egy Dead Gateway detektálásakor teszi a TCP automatikusan a másodikat az első helyre. Ha ez is meghal, akkor jön a harmadik - amennyiben van.

Mikor jó ez? Ha például van egy központi irodánk és van egy vidéki telephelyünk, amelyhez a biztonság kedvéért két különböző vonalat is kiépítettünk. A telephelyen az egyik router az első DGW, a másik router a második DGW.

A detektálásnak van egy érdekes vonzata is. Ugyanis visszajelzés nem csak akkor nem jöhet, ha a default gateway, azaz a routerünk dobta fel a portjait - hanem akkor sem, ha az útvonal bármelyik eleme állt le, azaz bármelyik host a router mögötti ismeretlen világban. A Dead Gateway felismerés - és a DGW csere ekkor is megtörténik.

5.2.6.5 Forward RTO-Recovery (F-RTO)

Bejelentkezik a tüske. Egy pillanatra elromlik a vonal, egy csomagnál megnő az RTT. Tételezzük fel, hogy éppen egy sok szegmensből álló nagy ablak szegmenseinek küldése közben vagyunk, mondjuk az elején. Az egyik szegmensnél lejár az RTO, így nem kapjuk meg a visszaigazolást - miközben meglehetősen nagy valószínűséggel a csomagok megérkeztek, csak a vonal pillanatnyi belassulása miatt a visszaigazolás nem. Hivatalosan most újra kellene küldenünk az összes csomagot, melyeket a nem visszaigazolt csomagtól egészen a Send Window végéig küldtünk - valószínűleg feleslegesen.
Ilyenkor lép életbe az F-RTO algoritmus.

RFC 4138

A feladó először csak az ablak első olyan szegmensét küldi újra, amelyiknél lejárt az RTO. Megvárja a visszaigazolást. Ha ez megjött, akkor úgy csinál, mintha nem történt volna semmi és elküldi az ablak _utáni_ első szegmenst. (Feltéve, hogy a túloldali Receive Window-ban van még szabad hely.) Ha erre is megjött a visszaigazolás, az csak úgy lehetséges, ha az egész ablak tartalma korábban már megérkezett a túloldalra. (Ugyanis ha lenne benne lyuk, akkor a fogadó nem igazolná vissza az utóbbi csomagot.)
Megköszönjük, mehetünk tovább.

Mire is jó ez? Tipikusan a wifi-re. Amikor megy szépen a hálózat, de kiszámíthatatlanul fordulhatnak elő pillanatnyi lassulások.

5.2.6.6 Selective Acknowledgement (SACK)

Erről már volt szó korábban a TCP opcióknál, a kép teljessége miatt vettem csak bele a felsorolásba. (Ismétlés: ha a címzett nem kapott meg minden szegmenst egy ablakból, vissza tudja jelezni, hogy mely szegmensek hiányoznak. A feladó ezután csak azokat küldi újra és nem az egész ablakot.)

5.2.6.7 Karn algoritmus

Az algoritmus kidolgozása Phil Karn nevéhez kötődik[86]. Ha nagyon tömören akarom összefoglalni, akkor arról van szó, hogy van jó RTT, amelyből lehet RTO-t számolni - és van rossz, amelyből meg nem.

Nézzünk az utóbbira egy példát. Elküldtünk egy csomagot - ezzel le is nyomtuk a stopperóra gombját. Nem kaptunk visszaigazolást az RTO-n belül, így megdupláztuk az RTO-t, elküldtük még egyszer a csomagot. Ez már bevált, megjött a visszaigazolás. Csakhogy. Az RTT-t mérő stopper definíció szerint csak most, az ACK megérkezésekor állt le. Nos? Jó ez az RTT érték RTO számoláshoz? Nem igazán.
Karn azt javasolta ehelyett, hogy ha ilyen szituáció fordul elő, akkor ne használjuk az újraküldött csomag RTT értékét az RTO számításhoz: nemes egyszerűséggel csak hagyjuk ott, ahol volt. (Ez ugye az eredeti RTO*2.)

A Karn algoritmus:
http://en.wikipedia.org/wiki/Karn%27s_Algorithm
http://www.cs.utk.edu/~dunigan/tcptour/javis/tcp_karn.html

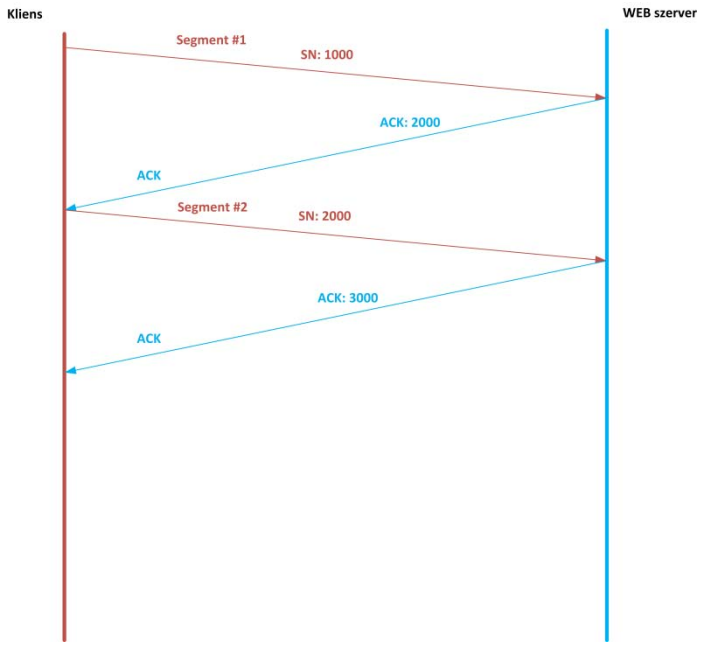
5.2.6.8 Fast Retransmit

RFC 2581

Beesik a várba egy kóbor lovag. Hát te melyik sereghez tartozol?

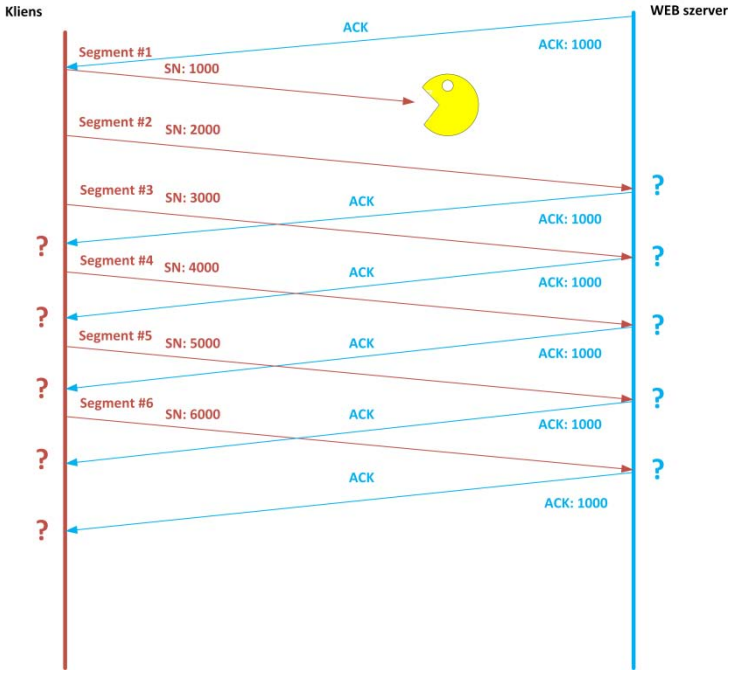
A fogadó egyszer csak kap egy csomagot, melynek az SN értéke nem illik a sorba. Ez egy kóbor csomag, mely eddig bóklászott valahol a hálózaton. A fogadó kínjában visszaigazolja, méghozzá az ACK mezőbe azt a Sequence Number értéket írja, melyre egyébként számított. Ezzel viszont a feladót hozza zavarba: szegény szerencsétlen két ugyanolyan ACK értékű visszaigazolást fog kapni. És ha még csak kettőt.

[86] Karn, P. and C. Partridge, "Improving Round-Trip Time Estimates in Reliable Transport Protocols", (SIGCOMM 87).



5.45. ÁBRA ÍGY KELLENE MENNIÜK A DOLGOKNAK

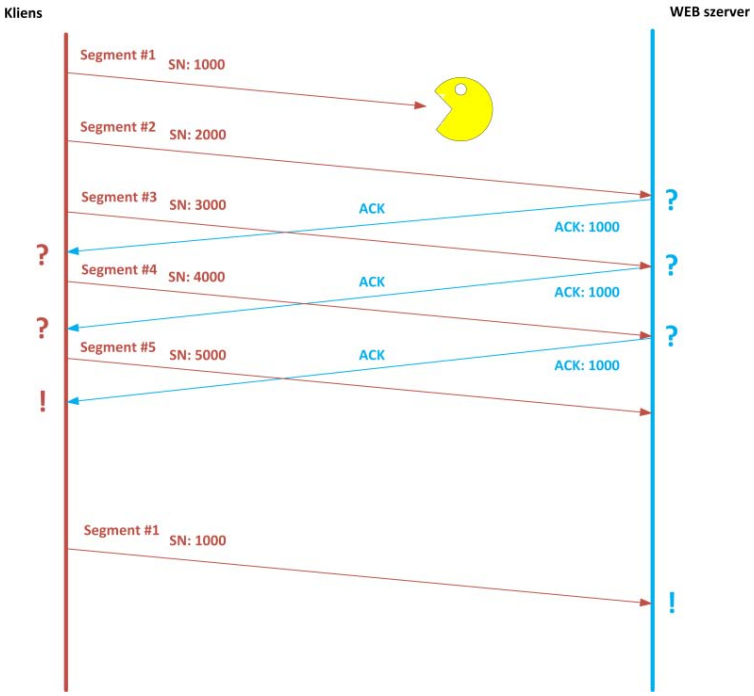
Szépen megy a csomag, jön a visszaigazolás. Mindenki boldog.



5.46. ÁBRA BAJ VAN

Itt történt valami váratlan dolog: megjelent a csomagelkapó manó valamelyik routeren: az 1-es TCP szegmens nem érkezett meg. Emiatt a címzett úgy érzékeli, hogy a 2-es TCP szegmens egy kósza szegmens - hiszen nem őrá számított. Gyorsan küld is vissza egy ACK-ot, méghozzá azzal az értékkel, mely tulajdonképpen a várt

Sequence Number. Ekkor jön meg a 3-as TCP szegmens, mely ugyanolyan kósza csomag elbírálásban részesül... és ez így megy tovább az RTO lejártáig.



5.47. ÁBRA A GYORS MEGOLDÁS

Pedig nem kellene, ugyanis a minta könnyen észrevehető: dőlnek sorban a visszaigazolások, ugyanazzal az ACK értékkel.

Itt jelenik meg a Fast Retransmit. Azt mondja, hogy ha háromszor kapunk vissza ugyanolyan ACK értékkel visszaigazolást, akkor kérés nélkül elküldjük újra az ACK értékhez tartozó TCP szegmenst, jelen esetben az 1-est. Innentől sorfolytonosak lesznek a csomagok a címzett pufferében, mehet tovább az élet.

Megoldottuk a problémát RTO-n belül.

Még egy dolgot kell megmagyaráznom: miért pont 3?
Azért, mert egy, vagy két duplikált ACK akkor is előfordulhat, ha a csomagok nem a megfelelő sorrendben érkeznek. A három viszont már több, mint gyanús.

6 AZ ALKALMAZÁS RÉTEG PROTOKOLLJAI, SZOLGÁLTATÁSOK

Nos, nézzük, hogyan állunk.

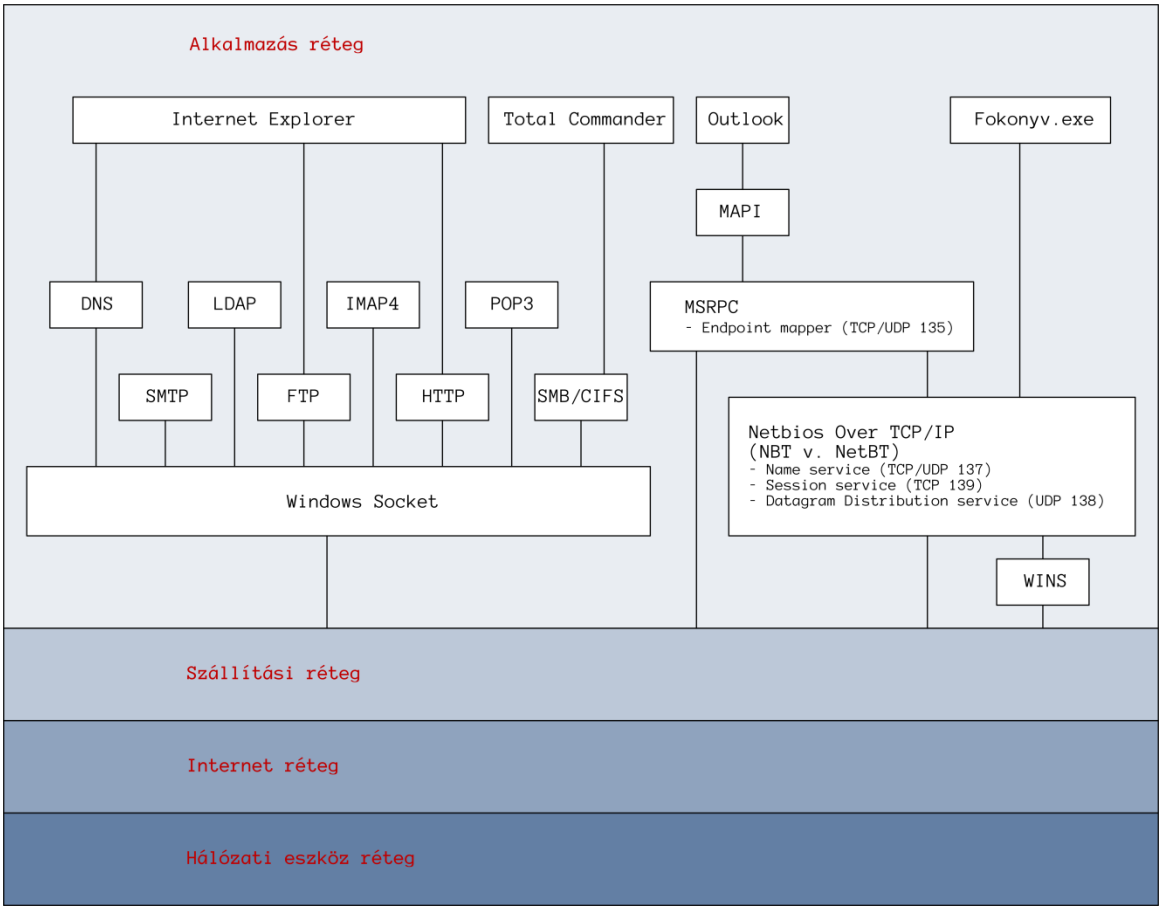
Létrehoztuk a postát. Tömérdek postás áll készenlétben, kempingbiciklin üldögélve, kismotorját brummogtatva vagy zöld furgonban cigarettázva - alig várják, hogy elszállítsák a leveleket.
Kitaláltuk a címeket. Kitaláltuk, hogy legyenek városok, legyenek utcák, a házakon legyenek számok és a bennük lakó embereknek legyenek neveik. Kitaláltuk azt is, hogy a címeket hová írjuk a küldeményeken.
Kitaláltunk szállítási módszereket. Gyorsat, lassút, megbízhatót, megbízhatatlant.

Most már csak azt a nyomorult levelet kellene megírnunk.

Elméletileg ezeket már írhatnánk spontán is. Rengeteg alkalmazás van, mely egyedi protokollokat, egyedi portokat használ információtovábbításra. De az evolúció létrehozott rögzített protokollokat is. Hiszen miért kellene minden egyes feladat megoldásánál újra és újra felfedeznünk a spanyolviaszkot?

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.



6.1. 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.

Ezekből a rögzített, alkalmazás-rétegbeli protokollokból mutatok be ebben a fejezetben kettőt. A lista bőven a teljesség igénye nélkül készült - ezekből a protokollokból ugyanis meglehetősen rengeteg van. (Részletesebben a második kötet fog foglalkozni velük.)

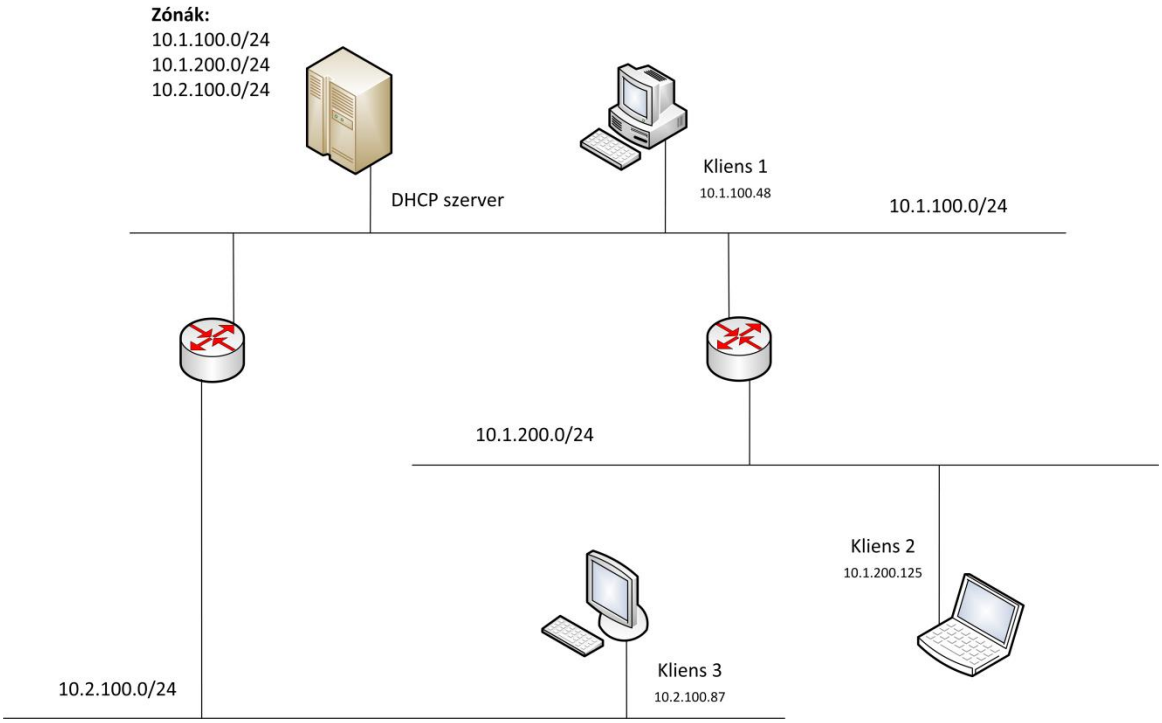
6.1 Dynamic Host Configuration Protocol (DHCP)

RFC 2131, 2132

Tapasztalatom szerint az egyik leginkább misztikusnak tartott alap hálózati szolgáltatás. Háde, hogyanmár... nincs IP címe, oszt mégis dzsal a hálózaton?

Bizony, bizony.

Aztán van itt még homály. Minden hálózati szakkönyv megemlíti, hogy egy DHCP szerveren több zónát is létre tudunk hozni. Na, de... amikor beesik a kliens, mi alapján dől el, hogy melyik zónából kap IP címet? Ha több hálózati kártya van a DHCP szerverben, akkor még érthető is a dolog, hiszen ezek nyilván más subnetbe lógnak bele, tehát a szervernek csak azt kell figyelnie, hogy az egyes hálózati kártyái mely zónákhoz tartoznak. De mi van akkor, ha a DHCP szerverben csak egy kártya van, ennek ellenére alhálózataink számosak?



6.2. ábra DHCP szerver összetett hálózatban

Habár a helyzet válságos, de nem reménytelen. A fejezet során fény derül a titokra.

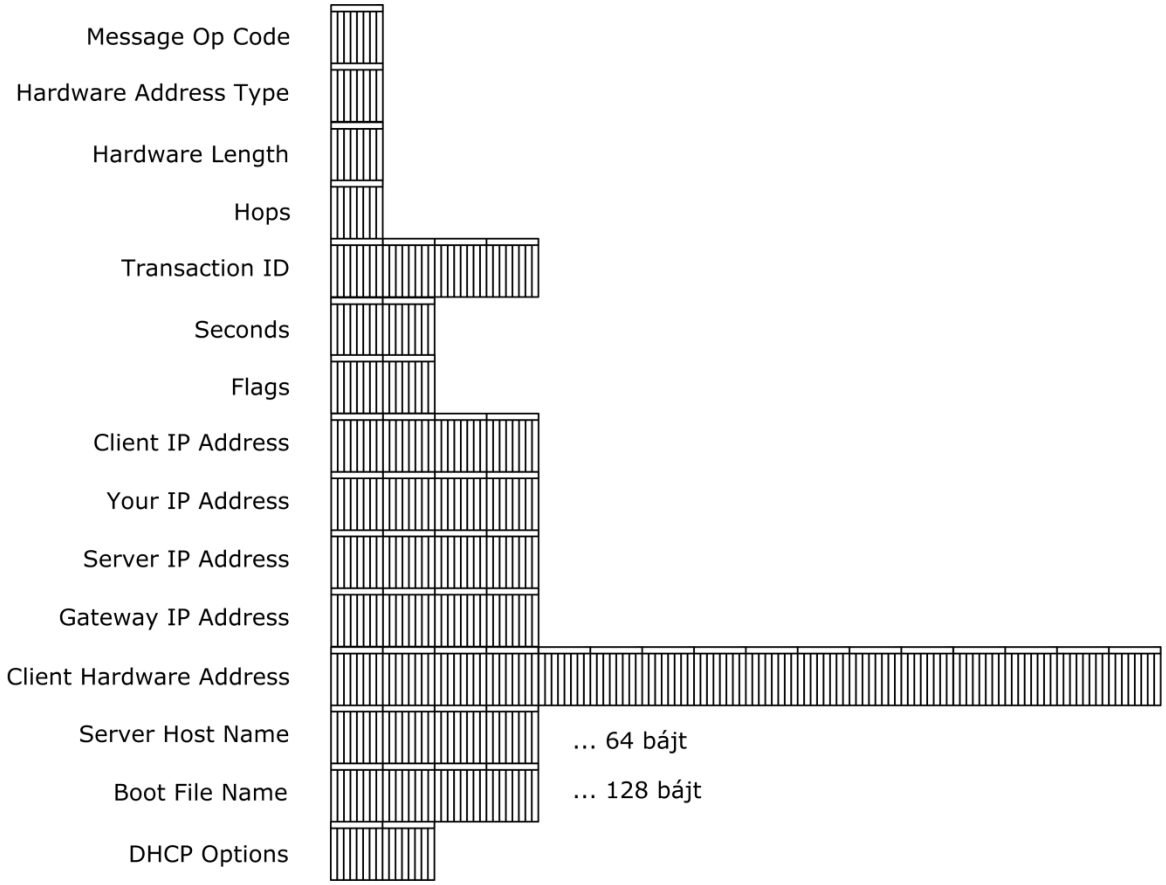
Először fussuk meg a kötelező köröket. Milyen tipusú üzenetekből állnak össze a DHCP folyamatai?

1. DHCPDISCOVER (FELFEDEZÉS): A DHCP kliens küldi, amikor keresi a DHCP szervert.
2. DHCPOFFER (AJÁNLAT): A DHCP szerver küldi, amikor a DHCP kliens felfedezi. Maga az ajánlat egy csomag, benne egy felkínált IP címmel és egyéb hálózati paraméterrel.
3. DHCPREQUEST (IGÉNY): A DHCP kliens küldi, ha elfogadja a szerver ajánlatát. Ezzel egyben vissza is utasítja a többi szerver ajánlatait.
4. DHCPACK (ELFOGADÁS, VISSZAIGAZOLÁS): A DHCP szerver tudomásul veszi a kliens döntését - és adminisztrál.
5. DHCPNAK (NO ACK, NEM ELFOGADÁS): A DHCP szerver küldi a kliensnek a DHCPREQUEST-re válaszul, amennyiben a kliens által igényelt IP címmel balhé van. (Például a kliens meg akarja újítani az IP címét, de az már lejárt vagy maga a kliens másik alhálózatba költözött.)
6. DHCPDECLINE (VISSZAUTASÍTÁS): A DHCP kliens küldi a DHCP szervernek, ha használhatatlan IP címet kap.
7. DHCPRELEASE (ELENGEDÉS): A DHCP kliens küldi a szervernek, amennyiben le akar mondani az eddig használt IP címéről.
8. DHCPINFORM (INFORMÁCIÓ): A DHCP kliens küldi a szervernek, amennyiben az alap hálózati csomaghoz képest plusz beállításokat is szeretne.

Most pedig csapongjunk át a DHCP csomag szerkezetére. Ha netán hiányérzeted lenne, a fenti üzenetek később rendesen is ki lesznek bontva.

DHCP Basics:
http://support.microsoft.com/kb/169289

6.1.1 CSOMAGSZERKEZET



6.3. ÁBRA A DHCP CSOMAG FELÉPÍTÉSE

Szép nagy. Meg tudnád mutatni rajta a jól megszokott tagolást: header és payload?
Bizony, nem.
Nincs. A legfelső szinten vagyunk. Vagy más nézőpontból, ez a legkisebb Matrjoska baba. Itt már minden payload, még a header is. Más szóval a kisérőinformációk is részei a protokollnak.

MESSAGE OP CODE: Request(1) vagy Reply(2)

HARDWARE ADDRESS TYPE: Hirtelen leszaladunk a földszintre. Hardware Address? Utoljára ilyen csúnya szavakat a Network Interface rétegben hallottunk, amikor az ARP folyamatot boncolgattuk. (*3.14. ábra ARP keret*) Itt is ugyanarról van szó: itt azonosítjuk be, hogy egyáltalán milyen hardvert, milyen hálózati csatlakozást használ a kliens. (Az Ethernet kódja 1, a Token Ring kódja 4, az EUI64 kódja meg 27.) Hogy még érthetőbb legyen: ez a kód azt adja meg, milyen tipusú MAC addressre számíthatunk a CLIENT HARDWARE ADDRESS mezőben.

HARDWARE ADDRESS LENGTH: Milyen hosszú a CLIENT HARDWARE ADDRESS mezőben található cím. Ethernet esetén a MAC Address 6 byte, tehát ez az érték 6.

HOPS: Hány darab DHCP Relay Agent-en[87] megy keresztül a DHCP kérés. RFC szerint a maximális érték 16, de a Windows Server 2008 esetében ez lecsökkent 4-re.

TRANSACTION ID: Véletlenszerűen kiválasztott 4 bájtos érték, ez azonosítja az egy folyamathoz tartozó üzeneteket.

SECONDS: Hány másodperc telt el azóta, hogy a DHCP kliens elindította a folyamatot.

FLAGS: Két bájtos mező. A bal szélső flag az úgynevezett broadcast flag. A többi 15 flag kötelezően nulla, értelmük majd valamikor a jövőben lesz definiálva. (Azért látszik, hogy még mindig hullanak a forgácsok.) Kliens oldali kérésnél a broadcast flag azt jelzi, hogy a kliens tud fogadni unicast módon címzett csomagot (0) vagy csak a broadcast (1) jó. A Windows Server 2008 / Vista DHCP kliensként az utóbbi értéket szokta beállítani.

CLIENT IP ADDRESS: A kliens IP címe, ahogy ő tudja.

YOUR IP ADDRESS: A kliens IP címe, ahogy a DHCP szerver tudja.

Server IP Address: A DHCP szerver címe.

GATEWAY IP ADDRESS: A kérést legelőször elkapó[88] DHCP Relay Agent IP címe.

CLIENT HARDWARE ADDRESS: 16 bájt, a DHCP kliens hardver azonosítója, Ethernet esetében ez a 6 bájtos MAC Address. Látható, hogy itt jön be a képbe a korábbi két HARDWARE ADDRESS mező.

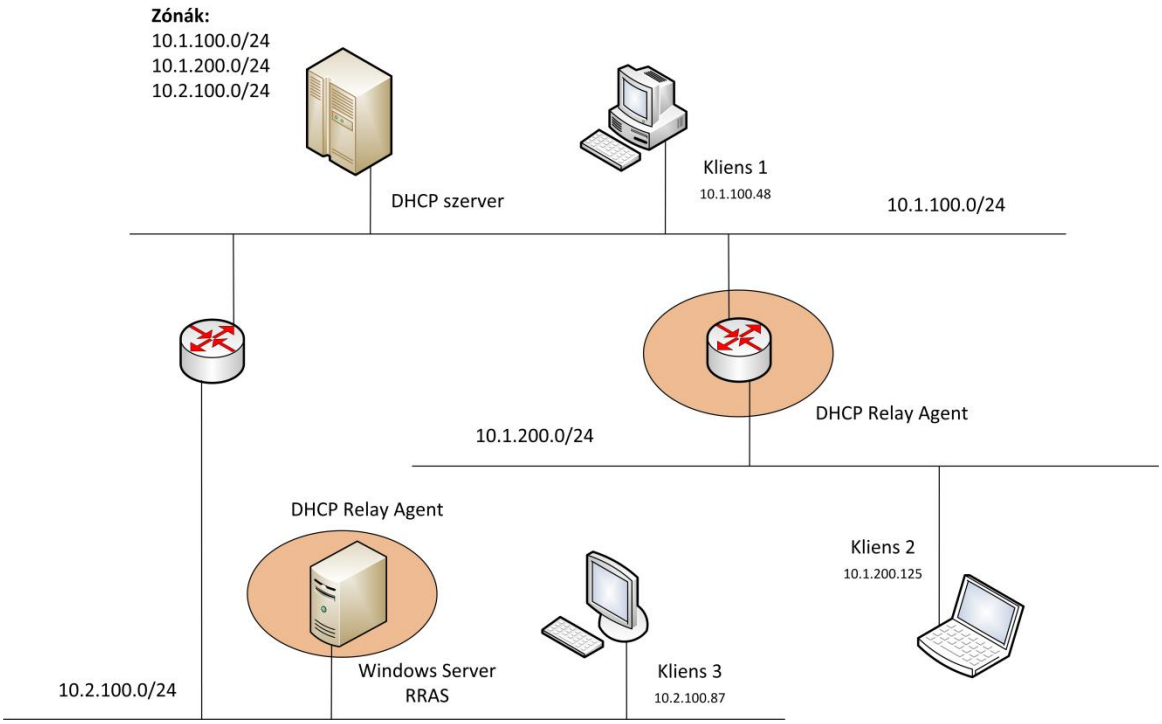
SERVER HOST NAME: A DHCP szerver neve. A Windows Server 2008 nem foglalkozik ezzel a mezővel.

BOOT FILE NAME: 128 bájt, gyakorlatilag tök feleslegesen. Itt lehet tárolni egy BOOTP image elérhetőségi útját. (A BOOTP volt a DHCP elődje.) Manapság a DHCP már nem használja ezt a mezőt.

OPTIONS: A megszokott gumimező. Gyakorlatilag minden, ami nem fért bele az eddig rubrikákba.

[87] Nocsak. Alakul a magyarázat a subnetelt DHCP rendszerre.
[88] Naa??

Remélhetőleg kezd tisztulni a helyzet a sok alhálózatos példával kapcsolatban.



6.4. ÁBRA DHCP RELAY AGENT KONFIGURÁCIÓ

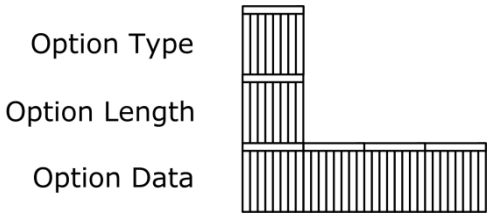
A megoldás kulcsa a DHCP Relay Agent-ek megjelenése. Ehhez nem kell feltétlenül plusz hardver, az ábrán is igyekeztem érzékeltetni, hogy meg lehet oldani a feladatot úgy is, hogy maga a hálózati eszköz - router célhardver - tudja a DHCP Relay funkciót... de meg lehet oldani úgy is, hogy az érintett alhálózaton valamelyik Windows szerverre felrakjuk az RRAS funkciót és azon belül állítjuk be a DHCP Relay Agent szerepkört.

Mi is történik ekkor?

Az alhálózaton a kliens IP címet szeretne kérni, ezért elküld egy DHCPDISCOVER üzenetet. Ezt a Relay Agent kapja el és ő küldi tovább a DHCP szervernek - de közben a HOPS mezőben található értéket megnöveli eggyel, a GATEWAY IP ADDRESS mezőbe pedig beírja a saját IP címét - mégpedig azt, amelyik egy alhálózaton van a kérést küldő klienssel. Ebből fogja tudni a távoli subneten lévő DHCP szerver, hogy melyik zónából adjon IP címet a kliensnek.

6.1.2 DHCP Options

Megint matekozással kezdünk. Tudni kell, hogy a DHCP szigorúan UDP-t használ csomagszállításra. (A kliens a 67-es porton figyel, a szerver a 68-ason.) Mit is írtunk erről a szállítási módról? Azt hogy dobozban szállítunk teljes csomagokat - azaz egy DHCP üzenetnek bele kell férnie az MTU által meghatározott csomagméretbe.
Ha összeadjuk az előbbi ábrán látható értékeket, akkor látszik, hogy 236 bájt biztosan lesz, ehhez jönnek majd a DHCP Options mezőben található adatok.
Konkrét példa: Ethernet csomag esetén az MTU 1500 bájt, a minimális IP header 20 bájt, az UDP header 8 bájt... azaz a DHCP opciók számára maximum 1236 bájt marad.



6.5. ábra DHCP Options

A DHCP opciók szerkezete a jól ismert sémát követi, nem is részletezném jobban. Típus, a blokk hossza, a blokk adatai... aztán jöhet az újabb blokk. Amíg van abból az 1236 bájtból.

DHCP Option Format:
http://www.tcpipguide.com/free/t_DHCPOptionsOptionFormatandOptionOverloading-2.htm

6.1. TÁBLÁZAT

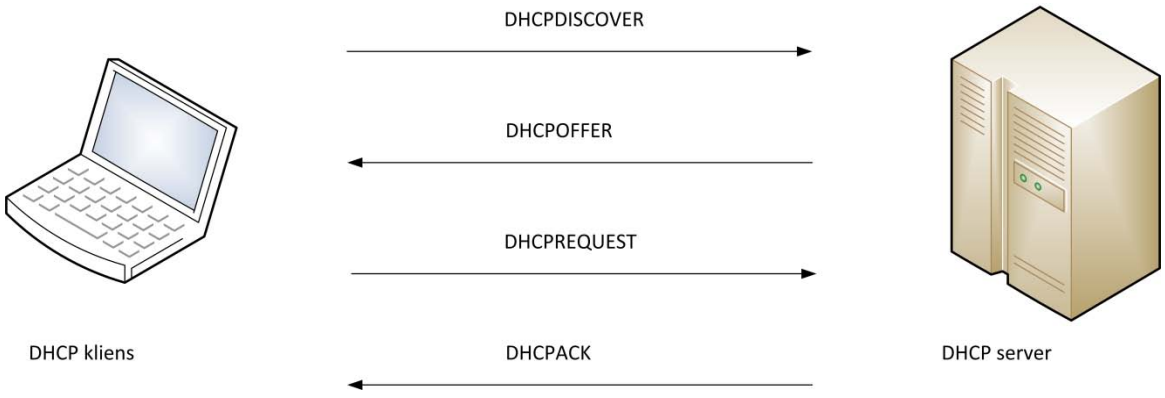
Option Type	Option Name	Option Length	Magyarázat
0	Pad	-	1 bájt, az értéke 0. Elválasztójel.
1	Subnet Mask	4 bájt	A kiajánlott cím alhálózati maszkja.
3	Router	Változó (4*x bájt)	A kliens hálózatán a routerek IP címei. Az első a Default Gateway.
6	Domain Name Servers	Változó (4*x bájt)	A DNS szerverek IP címei
12	Host Name	Változó	A kliens neve
15	DNS Domain Name	Változó	A tartomány neve, amelyikbe a kliens tartozik.
31	Perform Router Discovery	1 bájt	A kliensnek a Router Discovery IP opciót kell használnia a routerek felderítéséhez.
33	Static Route	Változó (8*x bájt)	Classful bejegyzések a kliens lokális route táblájába. 4 bájt IP cím, 4 bájt router IP cím.
43	Vendor-specific information	Változó	Bármi, ami gyártóspecifikus.
44	WINS/NBNS Servers	Változó (4*x bájt)	A WINS szerverek listája.
46	NetBIOS Over TCP/IP Node Type	1 bájt	A kliens NetBIOS tipusa: 1 (0001) - B node (Broadcast) 2 (0010) - P node (Point-to-Point) 4 (0100) - M node (Mixed) 8 (1000) - H node (Hybrid)
47	NetBIOS Scope ID	Változó	RFC 1001/1002
50	Requested Address	4 bájt	A kért, illetve elutasított IP cím
51	Lease Time	4 bájt	A bérlet ideje másodpercben
53	DHCP Message Type	1 bájt	A DHCP üzenet tipusa 1. DHCPDISCOVER 2. DHCPOFFER 3. DHCPREQUEST 4. DHCPDECLINE 5. DHCPACK 6. DHCPNAK 7. DHCPRELEASE 8. DHCPINFORM
54	Server Identifier	4 bájt	A DHCP szerver IP címe
55	Parameter Request List	Változó	Mely DHCP opciókat kéri a kliens. (A Windows Server 2008 / Vista által kért opciókat narancssárgával jelöltem.)
58	Renewal Time (T1)	4 bájt	Az az időtartam (sec), amelyen belül a kliensnek meg kell újítania a bérletét.
59	Rebinding Time (T2)	4 bájt	Időtartam (sec), melynek letelte után a kliens Rebind állapotba kerül.
61	Client Identifier	Változó	A DHCP klienst azonosítja. Ethernet esetben ez a MAC Address.
81	Dynamic DNS Update	Változó	A kliens teljes neve (FQDN). Ezt fogja használni a DHCP, ha be kell regisztrálnia a nevet a DNS-be.
121	Classless Static Route	Változó	Komplett Classless bejegyzések a kliens route táblájába: prefix, subnet mask, next hop.
249	Classless Static Route	Változó	Ugyanaz, mint a 121-es.
255	End	-	A DHCP opciók vége. (Csupa egyesekből álló bájt.)

Ha alaposabban megnézed, láthatod, hogy vannak még lyukak. Ez abból következik, hogy a Windows alapú DHCP szerverek nem támogatják az összes DHCP opciót. A teljes listát az alábbi linken találhatod.

Az összes DHCP opció:
http://www.iana.org/assignments/bootp-dhcp-parameters/

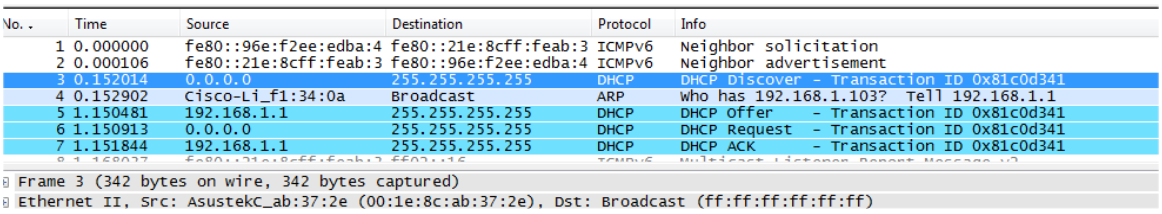
6.1.3 DHCP FOLYAMATOK

6.1.3.1 EGY NORMÁLIS CÍMBÉRLÉS NYÉLBEÜTÉSE



6.6. ÁBRA DHCP CÍMBÉRLÉS

Ha nekem nem hiszel, itt van egy fotó a capture fájlról.



No.	Time	Source	Destination	Protocol	Info
1	0.000000	fe80::96e:f2ee:edba:4	fe80::21e:8cff:feab:3	ICMPv6	Neighbor solicitation
2	0.000106	fe80::21e:8cff:feab:3	fe80::96e:f2ee:edba:4	ICMPv6	Neighbor advertisement
3	0.152014	0.0.0.0	255.255.255.255	DHCP	DHCP Discover - Transaction ID 0x81c0d341
4	0.152902	Cisco-Li_f1:34:0a	Broadcast	ARP	Who has 192.168.1.103? Tell 192.168.1.1
5	1.150481	192.168.1.1	255.255.255.255	DHCP	DHCP Offer - Transaction ID 0x81c0d341
6	1.150913	0.0.0.0	255.255.255.255	DHCP	DHCP Request - Transaction ID 0x81c0d341
7	1.151844	192.168.1.1	255.255.255.255	DHCP	DHCP ACK - Transaction ID 0x81c0d341

Frame 3 (342 bytes on wire, 342 bytes captured)
Ethernet II, Src: AsustekC_ab:37:2e (00:1e:8c:ab:37:2e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

6.7. ÁBRA DHCP CÍMBÉRLÉS

Itt fogunk választ kapni az első kérdésünkre: hogyan tud kommunikálni egy kliens a hálkózaton IP cím nélkül? Az ábrán valami gyanúsat már láthatsz is: egy csupa nulla IP című valami küld egy világméretű broadcast üzenet a subnetre (3. csomag), a DHCP szerver visszabroadcastol (5. csomag), majd újból jön a csupanulla (6. csomag), aztán megint a DHCP szerver (7. csomag) - és a négy üzenetből négy broadcast. Nos, igen. Amíg nincs IP címem, addig csak broadcasttal lehet megtalálni. (Hogyan is? Úgy, hogy habár mindenki felkapja a MAC/IP szintű broadcast csomagokat, de a forrásoldali MAC címek minden csomagban benne vannak, emellett a Transaction ID és a DHCP csomag tartalma egyértelművé teszi, hogy kik a társalgásban résztvevő felek és melyek az éppen aktuális tánclépések.)

Nézzük ezeket a lépéseket. Mindenhol egy-egy csomagot fogunk boncolgatni, méghozzá úgy, hogy az eredeti információkat külön színnel emeltem ki. Ezek közé fogom beirkálni a megjegyzéseimet.

6.1.3.1.1 DHCP DISCOVER

6.2. TÁBLÁZAT

No.	Time	Source	Destination	Protocol	Info
3	0.152014	0.0.0.0	255.255.255.255	DHCP	DHCP Discover - Transaction ID 0x81c0d341

```
Frame 3 (342 bytes on wire, 342 bytes captured)
Ethernet II, Src: AsustekC_ab:37:2e (00:1e:8c:ab:37:2e), Dst: Broadcast
(ff:ff:ff:ff:ff:ff)
Internet Protocol, Src: 0.0.0.0 (0.0.0.0), Dst: 255.255.255.255 (255.255.255.255)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 328
    Identification: 0x5411 (21521)
    Flags: 0x00
    Fragment offset: 0
    Time to live: 128
    Protocol: UDP (0x11)
    Header checksum: 0xe594 [correct]
    Source: 0.0.0.0 (0.0.0.0)
    Destination: 255.255.255.255 (255.255.255.255)
```

Az IP fejlécből látjuk, hogy igen, tényleg broadcast és tényleg UDP. De látunk mást is: az Ethernet frame-ben benne van a MAC címünk, a kérdés-felelet csomagokat pedig összeköti a Transaction ID.

```
User Datagram Protocol, Src Port: bootpc (68), Dst Port: bootps (67)
    Source port: bootpc (68)
    Destination port: bootps (67)
    Length: 308
    Checksum: 0x87a6 [correct]
```

Mint ahogy írtam, a kliens a 68-as portot használja (bootp client), a szerver pedig a 67-est (bootp server). Azaz a forrás a kliens, a cél a szerver.

```
Bootstrap Protocol
    Message type: Boot Request (1)
    Hardware type: Ethernet
    Hardware address length: 6
```

Kérésről van szó. Ethernet kártya, azaz 6 bájtos MAC Address.

```
    Hops: 0
```

Nincs DHCP Relay Agent.

```
    Transaction ID: 0x81c0d341
    Seconds elapsed: 0
    Bootp flags: 0x8000 (Broadcast)
        1... .... .... .... = Broadcast flag: Broadcast
        .000 0000 0000 0000 = Reserved flags: 0x0000
```

Igen, a broadcast flag. A Vista kliensem jelezte, hogy képtelen unicast választ fogadni, csak a broadcast a jó.

```
Client IP address: 0.0.0.0 (0.0.0.0)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 0.0.0.0 (0.0.0.0)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: AsustekC_ab:37:2e (00:1e:8c:ab:37:2e)
```

Ez a MAC Address-em. Minden más érték üres.

```
Server host name not given
Boot file name not given
```

Ez ugye az a nagy büdös (128 bájtnyi) semmi.

```
Magic cookie: (OK)
```

Itt fogtunk valamit. Erről a mágikus sütiről eddig még nem esett szó.

RFC 951, 1048

Az RFC 951 írja le a BOOTP (BootStrap) protokoll működését. Ennek kiegészítése az RFC 1048, amelyben a vendorfüggetlen BOOTP implementációt igyekeznek definiálni. Itt mondják ki, hogy ahhoz, hogy egy BOOTP megvalósítás gyártófüggetlen legyen, az Options mezőnek a következő számsorozattal kell kezdődnie: 99.130.83.99. (63.82.53.63 hexadecimálisan.)

```
    Boot file name not given
    Magic cookie: (OK)
  ⊞ Option: (t=53,l=1) DHCP Message Type = DHCP Offer
  ⊞ Option: (t=1,l=4) Subnet Mask = 255.255.255.0
  ⊞ Option: (t=3,l=4) Router = 192.168.1.1
0110  00 00 00 00 00 00 63 82  53 63 35 01 02 01 04 ff   ......c. Sc5.....
0120  ff ff 00 03 04 c0 a8 01  01 06 08 54 02 2c 01 54   ........ ...T.,.T
0130  02 2e 01 33 04 00 01 51  80 36 04 c0 a8 01 01 ff   ...3...Q .6......
```

6.8. ÁBRA MAGIC COOKIE

Kompatibilitási okból a DHCP-t úgy rakták össze, hogy azt a BOOTP vendorfüggetlennek lássa - értelemszerűen emiatt a DHCP Options blokk kötelezően ezzel a számsorozattal kell induljon. Ez a magic cookie.

```
Option: (t=53,l=1) DHCP Message Type = DHCP Discover
    Option: (53) DHCP Message Type
    Length: 1
    Value: 01
```

53-as opció, az üzenet tipusa. Discovery. Még jó.

```
Option: (t=116,l=1) DHCP Auto-Configuration
    Option: (116) DHCP Auto-Configuration
    Length: 1
    Value: 01
```

Megint fogtunk valamit. Ez az érték nem szerepel a táblázatban.

RFC 2563

Az autokonfigurációról esett már szó, az IPv6-os részben. Ez persze nem jelenti azt, hogy az IPv4-ben semmi ilyesmi nem lenne. Windows környezetben ott van ugye az Automated Private Addressing, azaz APIPA (169.254.0.1-169.254.255.254).
Maga a 116-os DHCP opció arról szól, hogy ápoljuk valahogy a DHCP kliens lelkét, ha nem kapott IP címet, amikor pedig kért.
Mindegyik kliens a DHCPDISCOVER üzenetben jelzi, hogy képes-e az autokonfigurációra, vagy sem. Jelen esetben a 01 érték azt jelzi, hogy az enyém igen. (Naná: APIPA.) Amennyiben a DHCP szerver valamiért nem tud neki IP címet osztani, akkor két dolgot tehet. Amennyiben a kliens már jelezte, hogy képes önmagát konfigurálni, akkor a szerver nem foglalkozik vele. Majd megoldja. Amennyiben viszont a kliens azt jelezte, hogy nincs autokonfigurációs mechanizmusa, akkor a szerver visszaküld neki egy DHCPOFFER csomagot, ahol is a 116-os opcióban az érték 0 - azaz DoNotAutoconfigure. Innentől kezdve a kliensre van bízva, hogy gyorsan fejlesszen ki magának valamilyen önmenedzselő képességet, ha kommunikálni akar - mert a DHCP szerver nem foglalkozik vele. (Amennyiben a DHCP kliensben a 116-os opció értéke DoNotAutoconfigure, azaz 0 volt... és nem kap választ a DHCPDISCOVER kérdésére - az azt jelenti, hogy nincs DHCP szerver a környéken.)

```
Option: (t=61,l=7) Client identifier
    Option: (61) Client identifier
    Length: 7
    Value: 01001E8CAB372E
    Hardware type: Ethernet
    Client MAC address: AsustekC_ab:37:2e (00:1e:8c:ab:37:2e)
```

Jé, megint a MAC címünk.

```
Option: (t=50,l=4) Requested IP Address = 192.168.1.103
    Option: (50) Requested IP Address
    Length: 4
    Value: C0A80167
Option: (t=12,l=2) Host Name = "hq"
    Option: (12) Host Name
    Length: 2
    Value: 6871
```

Régi IP cím, régi név.

```
Option: (t=60,l=8) Vendor class identifier = "MSFT 5.0"
    Option: (60) Vendor class identifier
```

```
        Length: 8
        Value: 4D53465420352E30
    Option: (t=55,l=12) Parameter Request List
        Option: (55) Parameter Request List
        Length: 12
        Value: 010F03062C2E2F1F2179F92B
        1 = Subnet Mask
        15 = Domain Name
        3 = Router
        6 = Domain Name Server
        44 = NetBIOS over TCP/IP Name Server
        46 = NetBIOS over TCP/IP Node Type
        47 = NetBIOS over TCP/IP Scope
        31 = Perform Router Discover
        33 = Static Route
        121 = Classless Static Route
        249 = Classless Static Route (Microsoft)
        43 = Vendor-Specific Information
    End Option
    Padding
```

Itt azt kell látni, hogy a Parameter Request List/Value mezőben egy-egy bájt jelöli a szükséges paraméter azonosítóját. Például a 'NetBIOS over TCP/IP Scope'-hoz a 0x2F azonosító tartozik, mely emberi nyelvre fordítva 47.

6.1.3.1.2 DHCP OFFER

6.3. TÁBLÁZAT

No.	Time	Source	Destination	Protocol	Info
5	1.150481	192.168.1.1	255.255.255.255	DHCP	DHCP Offer - Transaction ID 0x81c0d341

```
Frame 5 (590 bytes on wire, 590 bytes captured)
Ethernet II, Src: Cisco-Li_f1:34:0a (00:18:f8:f1:34:0a), Dst: Broadcast
(ff:ff:ff:ff:ff:ff)
Internet Protocol, Src: 192.168.1.1 (192.168.1.1), Dst: 255.255.255.255 (255.255.255.255)
User Datagram Protocol, Src Port: bootps (67), Dst Port: bootpc (68)
Bootstrap Protocol
    Message type: Boot Reply (2)
```

Igen, ez egy válasz.

```
    Hardware type: Ethernet
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x81c0d341
    Seconds elapsed: 0
    Bootp flags: 0x8000 (Broadcast)
        1... .... .... .... = Broadcast flag: Broadcast
        .000 0000 0000 0000 = Reserved flags: 0x0000
```

Nos, a szerver is csak broadcastot tud fogadni.

```
    Client IP address: 0.0.0.0 (0.0.0.0)
    Your (client) IP address: 192.168.1.103 (192.168.1.103)
```

Hohó! Itt van a szerver által felajánlott IP cím!

```
Next server IP address: 192.168.1.1 (192.168.1.1)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: AsustekC_ab:37:2e (00:1e:8c:ab:37:2e)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) DHCP Message Type = DHCP Offer
    Option: (53) DHCP Message Type
    Length: 1
    Value: 02
Option: (t=1,l=4) Subnet Mask = 255.255.255.0
    Option: (1) Subnet Mask
    Length: 4
    Value: FFFFFF00
Option: (t=3,l=4) Router = 192.168.1.1
    Option: (3) Router
    Length: 4
    Value: C0A80101
Option: (t=6,l=8) Domain Name Server
    Option: (6) Domain Name Server
    Length: 8
    Value: 54022C0154022E01
    IP Address: 84.2.44.1
    IP Address: 84.2.46.1
Option: (t=51,l=4) IP Address Lease Time = 1 day
    Option: (51) IP Address Lease Time
    Length: 4
    Value: 00015180
Option: (t=54,l=4) Server Identifier = 192.168.1.1
    Option: (54) Server Identifier
    Length: 4
    Value: C0A80101
```

Itt pedig az ajánlat többi paraméterei.

```
End Option
Padding
```

6.1.3.1.3 DHCP REQUEST

6.4. TÁBLÁZAT

No.	Time	Source	Destination	Protocol	Info
6	1.150913	0.0.0.0	255.255.255.255	DHCP	DHCP Request - Transaction ID 0x81c0d341

```
Frame 6 (342 bytes on wire, 342 bytes captured)
Ethernet II, Src: AsustekC_ab:37:2e (00:1e:8c:ab:37:2e), Dst: Broadcast
(ff:ff:ff:ff:ff:ff)
Internet Protocol, Src: 0.0.0.0 (0.0.0.0), Dst: 255.255.255.255 (255.255.255.255)
User Datagram Protocol, Src Port: bootpc (68), Dst Port: bootps (67)
Bootstrap Protocol
    Message type: Boot Request (1)
    Hardware type: Ethernet
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x81c0d341
    Seconds elapsed: 0
    Bootp flags: 0x8000 (Broadcast)
        1... .... .... .... = Broadcast flag: Broadcast
        .000 0000 0000 0000 = Reserved flags: 0x0000
    Client IP address: 0.0.0.0 (0.0.0.0)
    Your (client) IP address: 0.0.0.0 (0.0.0.0)
    Next server IP address: 0.0.0.0 (0.0.0.0)
    Relay agent IP address: 0.0.0.0 (0.0.0.0)
    Client MAC address: AsustekC_ab:37:2e (00:1e:8c:ab:37:2e)
    Server host name not given
    Boot file name not given
    Magic cookie: (OK)
    Option: (t=53,l=1) DHCP Message Type = DHCP Request
        Option: (53) DHCP Message Type
        Length: 1
        Value: 03
```

Itt jelezzük, hogy ez már egy határozott kérés, azaz request. Eddig csak érdeklődtünk.

```
    Option: (t=61,l=7) Client identifier
        Option: (61) Client identifier
        Length: 7
        Value: 01001E8CAB372E
        Hardware type: Ethernet
        Client MAC address: AsustekC_ab:37:2e (00:1e:8c:ab:37:2e)
    Option: (t=50,l=4) Requested IP Address = 192.168.1.103
        Option: (50) Requested IP Address
        Length: 4
        Value: C0A80167
    Option: (t=54,l=4) Server Identifier = 192.168.1.1
        Option: (54) Server Identifier
        Length: 4
        Value: C0A80101
    Option: (t=12,l=2) Host Name = "hq"
        Option: (12) Host Name
        Length: 2
        Value: 6871
    Option: (t=81,l=5) Client Fully Qualified Domain Name
        Option: (81) Client Fully Qualified Domain Name
        Length: 5
        Value: 0000006871
        Flags: 0x00
```

```
        0000 .... = Reserved flags: 0x00
        .... 0... = Server DDNS: Some server updates
        .... .0.. = Encoding: ASCII encoding
        .... ..0. = Server overrides: No override
        .... ...0 = Server: Client
        A-RR result: 0
        PTR-RR result: 0
        Client name: hq
```

És ezeket az értékeket szeretnénk.
(Az utolsó opció - 81 - tartalma meglepő egy kissé. A táblázatban viszonylag röviden el lett intézve - itt viszont láthatjuk, hogy van benne infó bőven.)

```
    Option: (t=60,l=8) Vendor class identifier = "MSFT 5.0"
        Option: (60) Vendor class identifier
        Length: 8
        Value: 4D53465420352E30
    Option: (t=55,l=12) Parameter Request List
        Option: (55) Parameter Request List
        Length: 12
        Value: 010F03062C2E2F1F2179F92B
        1 = Subnet Mask
        15 = Domain Name
        3 = Router
        6 = Domain Name Server
        44 = NetBIOS over TCP/IP Name Server
        46 = NetBIOS over TCP/IP Node Type
        47 = NetBIOS over TCP/IP Scope
        31 = Perform Router Discover
        33 = Static Route
        121 = Classless Static Route
        249 = Classless Static Route (Microsoft)
        43 = Vendor-Specific Information
    End Option
```

6.1.3.1.4 DHCP ACK

6.5. TÁBLÁZAT

No.	Time	Source	Destination	Protocol	Info
7	1.151844	192.168.1.1	255.255.255.255	DHCP	DHCP ACK - Transaction ID 0x81c0d341

```
Frame 7 (590 bytes on wire, 590 bytes captured)
Ethernet II, Src: Cisco-Li_f1:34:0a (00:18:f8:f1:34:0a), Dst: Broadcast
(ff:ff:ff:ff:ff:ff)
Internet Protocol, Src: 192.168.1.1 (192.168.1.1), Dst: 255.255.255.255 (255.255.255.255)
User Datagram Protocol, Src Port: bootps (67), Dst Port: bootpc (68)
Bootstrap Protocol
    Message type: Boot Reply (2)
    Hardware type: Ethernet
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x81c0d341
    Seconds elapsed: 0
    Bootp flags: 0x8000 (Broadcast)
        1... .... .... .... = Broadcast flag: Broadcast
        .000 0000 0000 0000 = Reserved flags: 0x0000
    Client IP address: 0.0.0.0 (0.0.0.0)
    Your (client) IP address: 192.168.1.103 (192.168.1.103)
    Next server IP address: 192.168.1.1 (192.168.1.1)
```

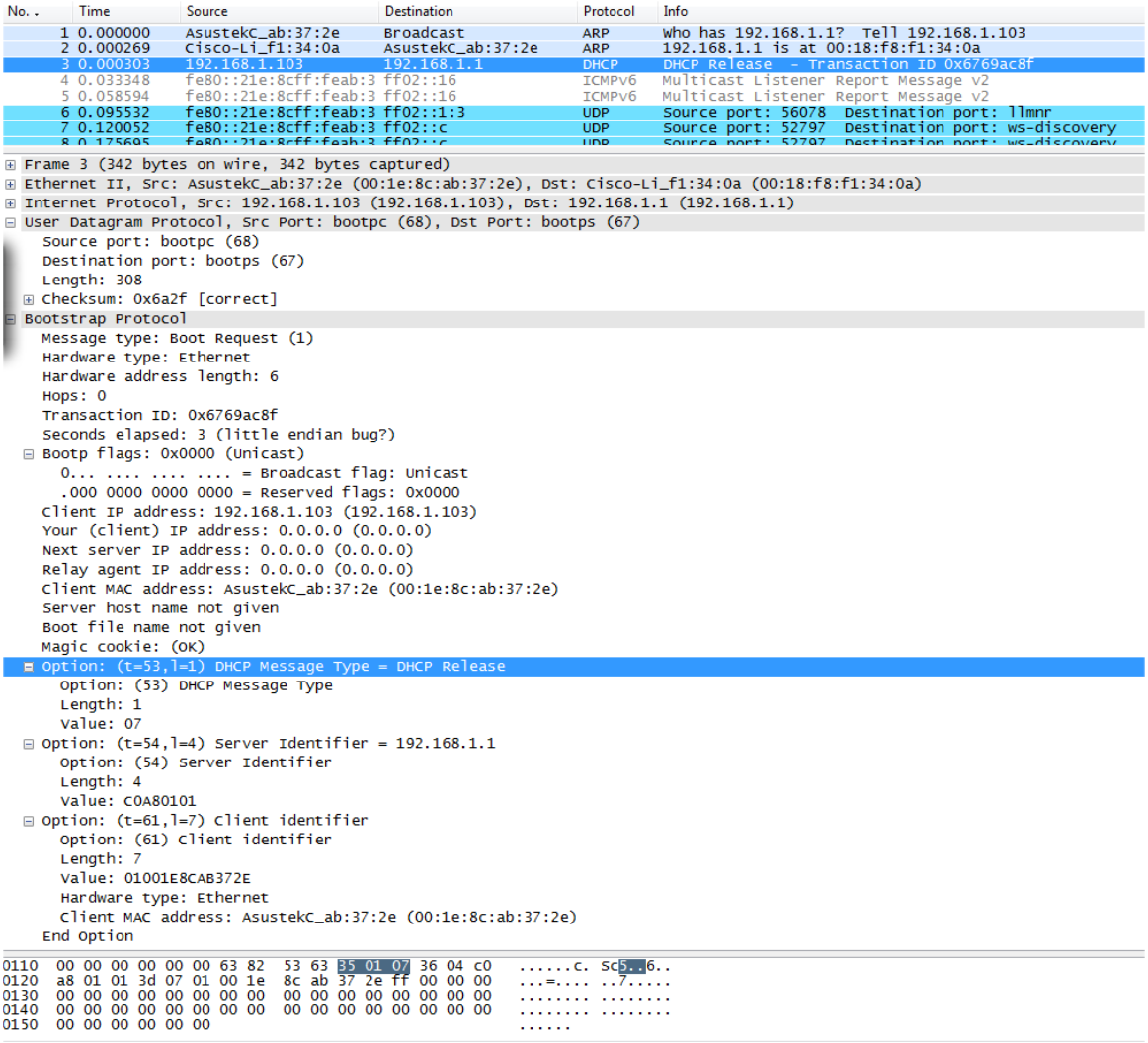
```
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: AsustekC_ab:37:2e (00:1e:8c:ab:37:2e)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) DHCP Message Type = DHCP ACK
    Option: (53) DHCP Message Type
    Length: 1
    Value: 05
```

Visszaigazoltuk. Tiéd a cím.

```
Option: (t=1,l=4) Subnet Mask = 255.255.255.0
    Option: (1) Subnet Mask
    Length: 4
    Value: FFFFFF00
Option: (t=3,l=4) Router = 192.168.1.1
    Option: (3) Router
    Length: 4
    Value: C0A80101
Option: (t=6,l=8) Domain Name Server
    Option: (6) Domain Name Server
    Length: 8
    Value: 54022C0154022E01
    IP Address: 84.2.44.1
    IP Address: 84.2.46.1
Option: (t=51,l=4) IP Address Lease Time = 1 day
    Option: (51) IP Address Lease Time
    Length: 4
    Value: 00015180
Option: (t=54,l=4) Server Identifier = 192.168.1.1
    Option: (54) Server Identifier
    Length: 4
    Value: C0A80101
End Option
Padding
```

6.1.3.2 EGY CÍM ELENGEDÉSE

Ez már nem lesz olyan bonyolult koreográfiájú társastánc, mint az előző. A kliens kijelenti, hogy el akar szakadni az IP címétől - majd elszakad.



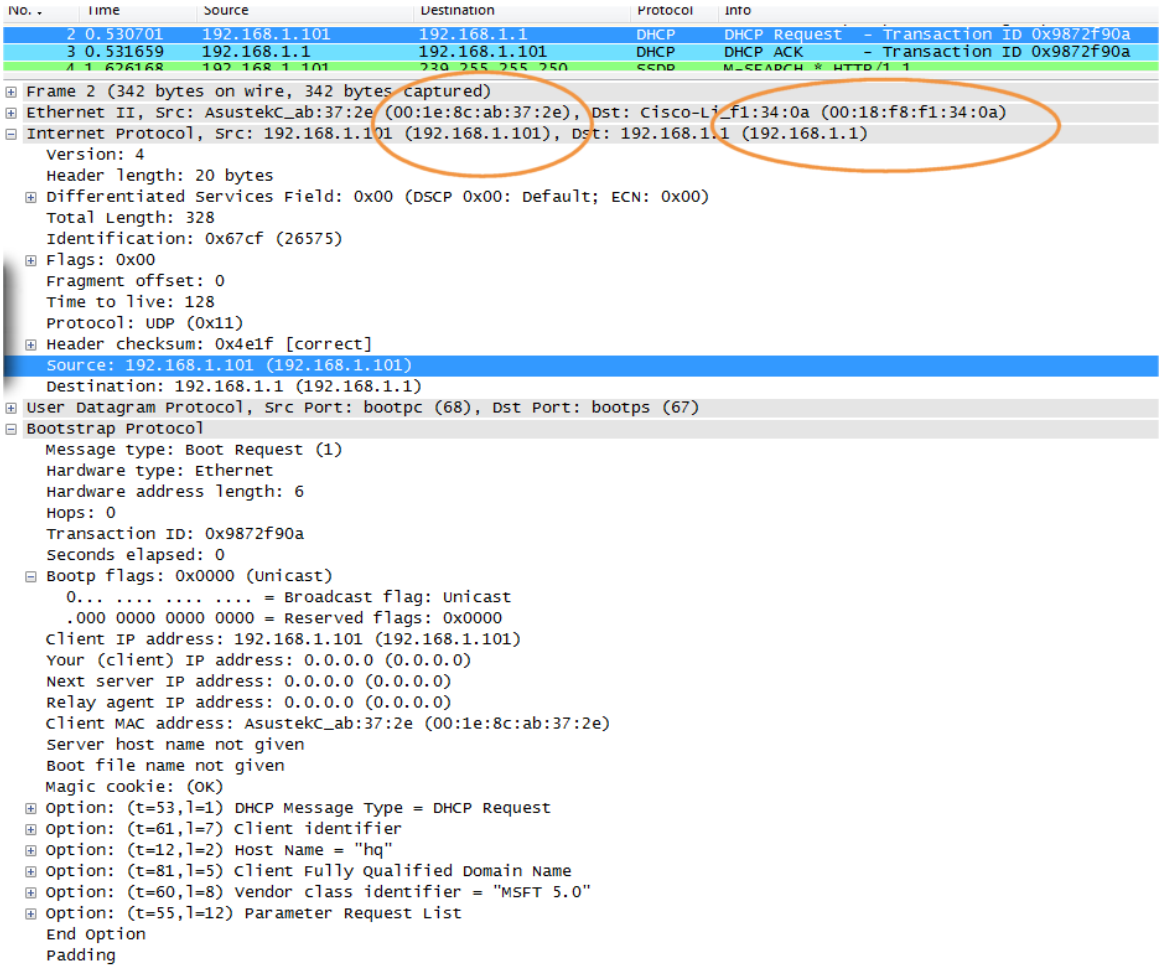
6.9. ÁBRA DHCP RELEASE

6.1.3.3 Renew és Rebuild

Kezdjük ott, hogy mi a különbség?

6.1.3.3.1 Renew

Általában a bérleti idő felénél - egész pontosan az 58-as DHCP opcióban megadott T1 idő eltelte után - a kliensnek meg kell újítania az IP címét. Ez egy gyors ütésváltással történik: jön egy DHCPREQUEST a klienstől, amire jön egy katonás DHCPACK a szervertől. Ráadásul ekkor még van a kliensnek IP címe is, azaz nem kell broadcastolni. (Legalábbis nem mindig.)

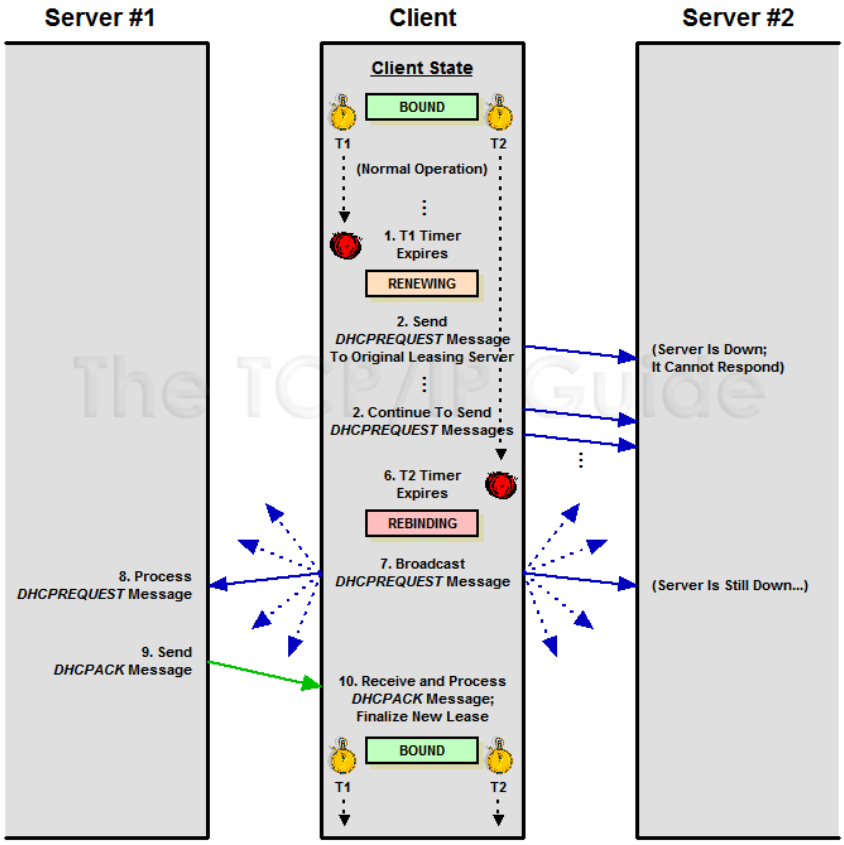


6.10. ábra DHCP Renew - IP címmel

Szépen látható a DHCP Request és a DHCP ACK válasz - ahol mind a két csomagnak rendes címzettje és feladója van. Szó sincs semmiféle broadcastról.

6.1.3.3.2 REBUILD

Egy kicsit cifrább dologról van szó. Letelt a T1 időtartam, a kliens eszeveszetten szeretné megújítani a bérletét - de a DHCP szerver nem érhető el. Terroristák felrobbantották. Leöntötték kakaóval. Elvitte a cica. Bármi. A kliens próbálkozik, próbálkozik... majd ha letelt az 59-es opcióban megadott T2 időtartam, akkor feladja: újrakezd egy DHCPDISCOVERY üzenettel, hátha talál másik DHCP szervert.



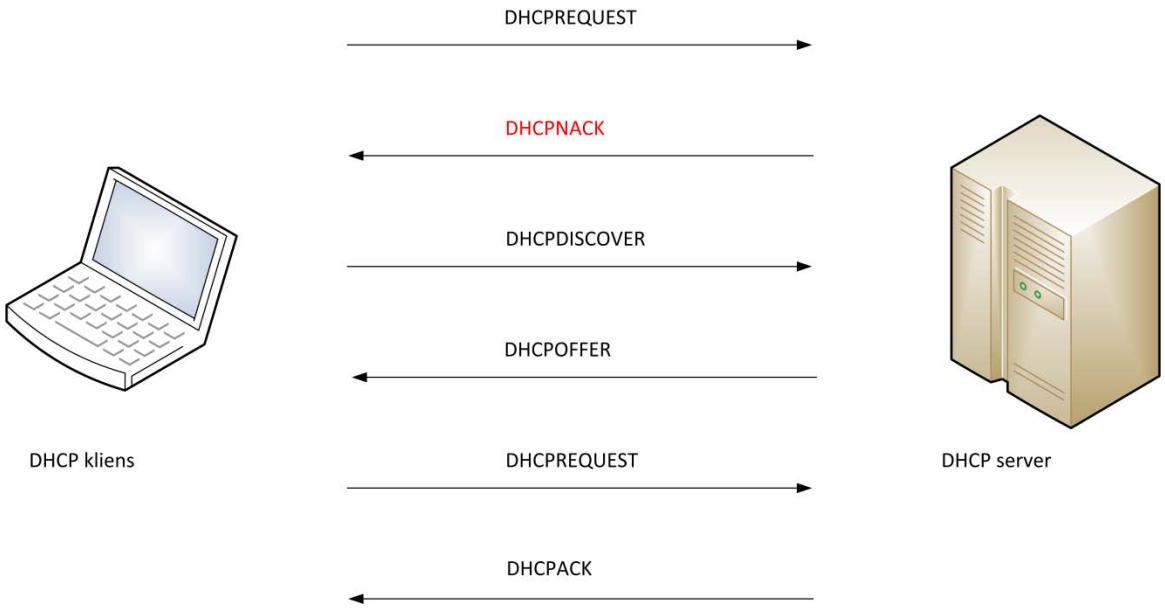
6.11. ÁBRA DHCP REBUILD (FORRÁS: THE TCP/IP GUIDE, WWW.TCPIPGUIDE.COM)

Renew and Rebind:
http://www.tcpipguide.com/free/t_DHCPLeaseRenewalandRebindingProcesses-2.htm

6.1.3.4 SUBNET VÁLTÁS

Láttuk, mi történik, ha a szerver hagyja cserben a kliensét. De mi van akkor, ha a bérlés kellős közepén a klienst átdugjuk egy másik subnetbe? (Azok a kollégák, akik menedzselnek állandóan úton lévő, telephelyek között cikázó, laptopos ügynöki hálózatot, gondolom, most megértően bólogatnak.)

Nos, ez történik.



6.12. ÁBRA DHCP ÉS SUBNET VÁLTÁS

A csóka T1 után szeretné megújítani az IP címét. El is küldi a DHCPREQUEST kérést, benne az igényelt, de már nem jó IP címmel - amire a szerver pofánvágja egy DHCPNAK üzenettel. Ettől a kliens megcondolodik és békülékeny hangnemben elindít egy klasszikus DHCPDISCOVER üzenettel kezdődő bérletkérési folyamatot.

6.1.4 DHCP v6

RFC 3315

Az IPv6 esetében már egy kicsit cifrább a helyzet. Itt ugyanis a kliensek már egész jól el tudnának boldogulni DHCP szerver nélkül is, hiszen láttuk, megvannak az algoritmusaik ahhoz, hogy a semmiből IP címeket kreáljanak maguknak, routereket fedezzenek fel, megéljenek a jég hátán.

Nyilvánvaló, hogy itt valahogyan rendet kell vágni, ne keveredjenek az eljárások.

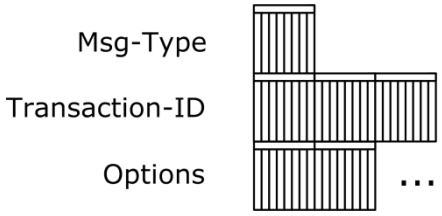
Erre a célra két flag-et fogtak be, mely flag-ek a Router Advertisement üzenetben utaznak. (*4.2.1.6 Autokonfiguráció*)

- M FLAG (MANAGED ADDRESS CONFIGURATION FLAG): Ha magas, akkor arra utasítja a hostot, hogy az elosztó központból kérjen magának IP címet.
- O FLAG (OTHER STATEFUL CONFIGURATION FLAG): Ha magas, akkor arra utasítja a hostot, hogy az elosztó központból kérje le a kiegészítő hálózati konfigurációs adatokat.

A két flag-ből logikusan négy szituáció következik:

- MINDKETTŐ ALACSONY. Ekkor nincs DHCP a hálózaton, boldoguljon mindenki, ahogyan tud.
- MINDKETTŐ MAGAS. A DHCP a király, mindent ő oszt. Ezt a figurát hívják *DHCPv6 Stateful* állapotnak.
- O MAGAS, M ALACSONY. A hostok az IP címeket a routerektől tudják meg. Minden egyéb kiegészítő paramétert a DHCP szerverektől. Ezt nevezik *DHCPv6 Stateless* állapotnak.
- O ALACSONY, M MAGAS. A DHCP osztja az IP címeket, az egyéb paraméterek pedig... hát, azok a fasorban sincsenek. Ennek az opciónak inkább csak matematikai lehetősége van, a gyakorlatban értelmetlen.

Menjünk bele a DHCPv6 üzenetek szerkezetébe. Elöljáróban mit vársz? Mondjuk, kiindulva a korábbi ábrából? (*6.3. ábra A DHCP csomag felépítése*)
Meg fogsz lepődni. Ugyanis ennyi:



6.13. ÁBRA A DHCPV6 ÜZENET SZERKEZETE

A változás oka, hogy a DHCPv4 igyekezett kompatibilis lenni a korábbi BOOTP szerkezettel. (Lásd tök felesleges, óriási mezők.) A DHCPv6 már nem tűzött ki maga elé ilyen célokat - így lehetőség adódott áramvonalasítani a modellt, egészen addig, amíg ilyen cseppforma nem lett belőle.

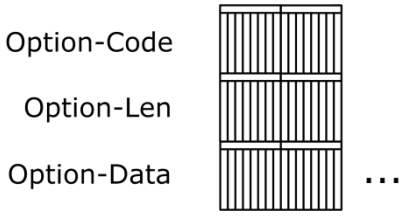
MSG-TYPE: Itt jelezzük, hogy milyen tipusú DHCPv6 üzenetről van szó.

6.6. TÁBLÁZAT

Msg-Type	Üzenet	Leírás	DHCPv4 megfelelőség
1	Solicit	A kliens keresi a szervert	DHCPDiscover
2	Advertise	A szerver válaszol a Solicit kérésre	DHCPOffer
3	Request	A kliens konkrétan elkéri az adatokat	DHCPRequest
4	Confirm	A kliens körbeérdeklődik az összes DHCP szervernél, hogy amit kapott, az jó-e	DHCP Request
5	Renew	Cím megújítás	DHCPRequest
6	Rebind	Címeldobás, amennyiben nem sikerült a Renew	DHCPRequest
7	Reply	A szerver visszaigazolja a kéréseket (request, confirm, rebind, reply)	DHCPAck
8	Release	A kliens üzeni, hogy pá, kisaranyom.	DHCPRelease
9	Decline	A kliens üzeni a szervernek, hogy az rossz anyagot szállított - ilyen cím már van.	DHCPDecline
10	Reconfigure	A szerver üzeni a kliensnek, hogy változtak a konfigurációs adatai, nézzen be újra. A kliens ilyenkor vagy Renew, vagy Information-Request üzenettel próbálkozik.	-
11	Information-Request	A kliens konfigurációs információkat kér. (De IP címeket nem.)	DHCPInform
12	Relay-Forward	A DHCP Relay Agent küldi az eredeti DHCP kérést, DHCPv6 Relay-Message üzenetbe csomagolva.	-
13	Relay-Replay	A DHCP szerver válaszol a Relay Agent-nek, szintén Relay-Message üzeneten keresztül.	-

TRANSACTION-ID: A szokásos folyamat azonosító.

OPTIONS: Ebben a változó méretű mezőben található a lényeg. A megszokott módon, - azaz blokkokra bontva - itt van minden, amit a felek közölni szeretnének egymással.



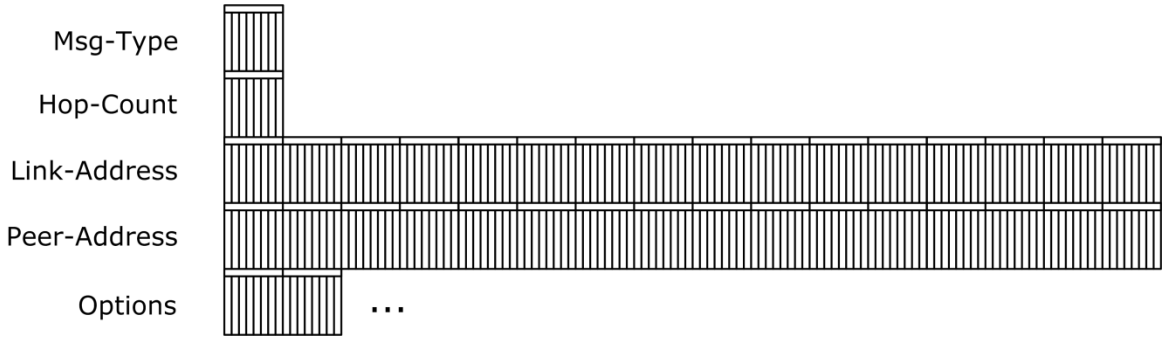
6.14. ÁBRA DHCPV6 OPTIONS

OPTION CODE: Az opció azonosítója

OPTION-LEN: Az opció mérete bájtban.

OPTION-DATA: Ebben a változó méretű mezőben jönnek az adatblokkok.

A *6.11. táblázatban* megemlítettem egy eddig még nem tárgyalt üzenettípust, a DHCP Relay-Message üzenetet. Valamiért úgy gondolták az RFC alkotói, hogy a Relay Agent-en keresztüli kommunkációt nem DHCPv6 opciókon keresztül valósítják meg, hanem külön üzenettípust vezetnek be erre a célra.



6.15. ÁBRA DHCPV6 RELAY-MESSAGE

MSG-TYPE: Ez ugyanaz, mint a másik tipusnál.

HOP-COUNT: Hány Agent-en keresztül érkezett meg a kérés a szerverhez.

LINK-ADDRESS: A Relay Agent azon IP címe, amelyikkel arra a hálózatra néz, ahol a kérést kezdeményező DHCP kliens lakik.

PEER-ADDRESS: Vagy a DHCP kliens, vagy az előző Relay Agent IP címe.

OPTIONS: A Relay-Message opciók adatblokkjai.

Végezetül fussuk át, hogyan is történik egy stateful és egy stateless DHCP címkiosztási folyamat.

Stateful, azaz az M és az O flag-ek is magasak, mindent a DHCP szerver oszt:

- A kilens küld egy Solicit üzenetet, hogy rátaláljon a szerverre.
- A szerver küld egy Advertise üzenetet, miszerint nála van olyasmi, amit a kliens keres.
- A kliens visszaküld egy Request üzenetet, immár a konkrét szervernek, miszerint kell neki az anyag.
- A szerver egy Reply üzenetben elküldi a kért adatokat.

Stateless, azaz csak az O flag magas, a DHCP szerver IP címet nem, csak egyéb konfigurációs adatokat (domain név, netbios névfeloldás tipusa, stb...) oszt:

- A kliens küld egy Information-Request üzenetet, konkrétan a szervernek.
- A szerver egy Replay üzenetben elküldi a kért adatokat.

6.2 Domain Name System (DNS)

Az előző fejezetben a MAC Address - IP cím összerendelésen pörögtünk, a mostani fejezetben feljebb lépünk a hierarchiában: az 'IP cím - mindenféle tartományi nevek' (FQDN) összerendeléssel fogunk foglalkozni.
Ezt az összerendelést nevezik DNS-nek - az adminisztrálását pedig DNS szerverek végzik. DNS kliensnek nevezünk mindenkit, akinek valamilyen folyó ügye van a DNS szerverekkel: címet szeretne beregisztrálni, törölni, lekérni... vagy elkérni egy teljes zóna adatait. Ilyen értelemben a DNS szerver is lehet más irányban DNS kliens.

A kommunikáció DNS üzenetek formájában megy, az 53-as porton. Ez alkalomtól függően vagy UDP vagy TCP szállítási protokollon keresztül történhet.

De mi is ez az alkalom?

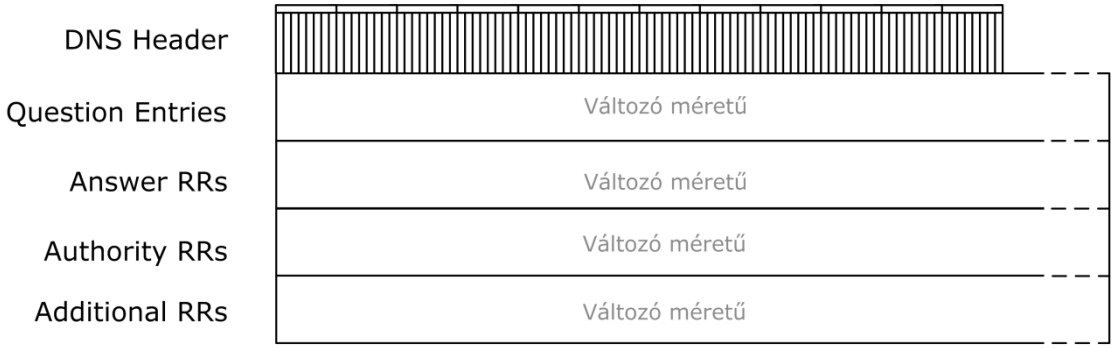
Nos, megint a méret a lényeg.
Tudjuk, hogy ha egy üzenetet az UDP-re bízunk, akkor annak kötelezően bele kell férnie egy csomagba. Ez a DHCP-nél nagyjából tartható is volt - ezzel szemben a DNS-ben a mezők többsége rugalmas méretű, ráadásul a kommunikáció jellegéből következően nagy mennyiségű adat is utazhat a csomagban - gondoljunk bele például egy zónatranszferbe.
Innentől viszonylag egyszerű az algoritmus: ha beleférünk egy csomagba, akkor UDP, ha nem férünk bele, akkor TCP. Ebből jön az, hogy a zónatranszfer nagy eséllyel TCP-n keresztül megy, míg a lekérdezésekhez elég az UDP is.

Négy üzenettípust különböztetünk meg:

- DNS Name Query Request: A DNS kliens küldi a szervernek, gyakorlatilag egy névfeloldási kérelem.
- DNS Name Query Response: A DNS szerver küldi a kliensnek, válaszul a névfeloldási kérelemre.
- DNS Update: A DNS kliens küldi a szervernek, be szeretne jegyezni egy rekordot.
- DNS Update Response: A szerver küldi a kliensnek, mi történt a bejegyzési kérelmével.

6.2.1 DNS NAME QUERY REQUEST ÉS NAME QUERY RESPONSE ÜZENETEK

Nagy vonalakban mindkét üzenettípus így néz ki:

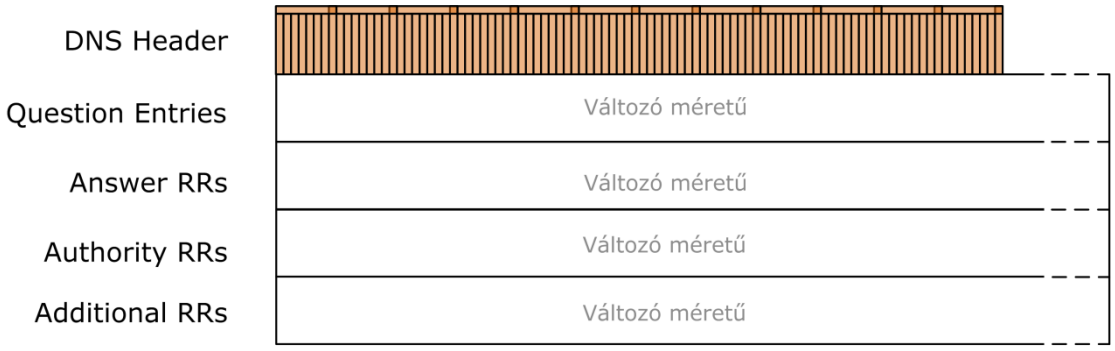


6.16. ÁBRA DNS NAME QUERY REQUEST ÉS NAME QUERY RESPONSE

Habár itt már nem kellene header+payload formában kettészedni a csomagot, de nem is tilos. A protokoll tervezői úgy gondolták, a fix méretű részt header-nek nevezik, a változó méretűeket meg... sehogy. Azok egyszerűen csak mezők.

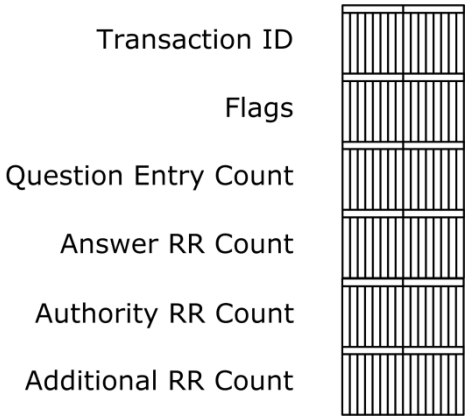
Még valami. Mit is takar az RR rövidítés? Resource Record - azaz egy bejegyzés a DNS szerveren. Nagyon sokszor fog szerepelni a fejezetben.

6.2.1.1 DNS Query Header



6.17. ábra DNS Header a fókuszban

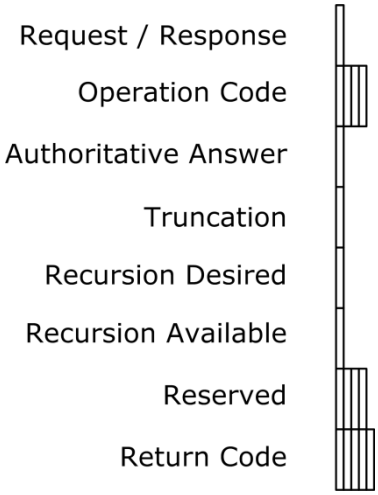
A 12 bájtnyi rész a következőképpen tagozódik tovább.



6.18. ábra DNS Header (Name Query Request / Name Query Response)

Transaction ID: A DHCP-nél megismert azonosító. A kliens generálja, mindenki más már csak bemásolgatja.

FLAGS: A szokásos tudatmódosító zászlócskák.



6.19. ÁBRA A FLAGS MEZŐ

REQUEST / RESPONSE: Request(0) vagy Response (1)

OPERATION CODE: Pontosan mit is csinál a szerver a csomaggal. Jelen esetben az értéke 0, ez a query kódja.

AUTHORITATIVE ANSWER: Értéke 1, amennyiben a válasz a zónára nézve autoritatív szervertől jött. (Azaz a szerver kezeli a zónát, nem kellett továbbmennie más DNS szerverekhez kérdezősködni.)

TRUNCATION: Ezen a flag-en keresztül jelezzük, hogy balhé van. Nem fértünk bele az UDP csomagba. Az üzenet első adagja még elment UDP-n, de innentől a felek TCP kapcsolatot nyitnak és azon folytatják tovább a kommunikációt.

RECURSION DESIRED: Amikor a kérdező eldönti, hogy rekurzív query-t szeretne (1) - azaz ha a DNS szerver nem autoritativ a kérdezett név zónájára nézve, akkor menjen utána a kérésnek és keresse meg a zónát valamelyik egyéb DNS szerveren - vagy csak dobja vissza azon DNS szerverek listáját (0), amelyeken jó eséllyel ott lesz a keresett zóna.

RECURSION AVAILABLE: A szerver jelzi vissza a Query Response-ban, hogy egyáltalán képes-e rekurzív keresést végrehajtani. (1 az igen, 0 a nem.)

RESERVED: Elspájzoltunk némi bitet rosszabb időkre.

RETURN CODE: A Query Response-ban játszik szerepet. Jelzi, hogy mi is történt tulajdonképpen a kéréssel.

A teljesség igénye nélkül.

6.7. TÁBLÁZAT

Kód	Leírás
0	Siker
1	Format Error
2	Szerverhiba
3	Nemlétező domain
4	Not implemented
5	Query elutasítva
6	Létező név, mely nem létezhetne
7	RR, mely nem létezhetne
8	RR, melynek léteznie kellene
9	A szerver nem autoritatív a zónára
10	A név nincs a zónában

Oké, kijöttünk a FLAGS mezőből, folytassuk tovább a DNS header boncolgatását.

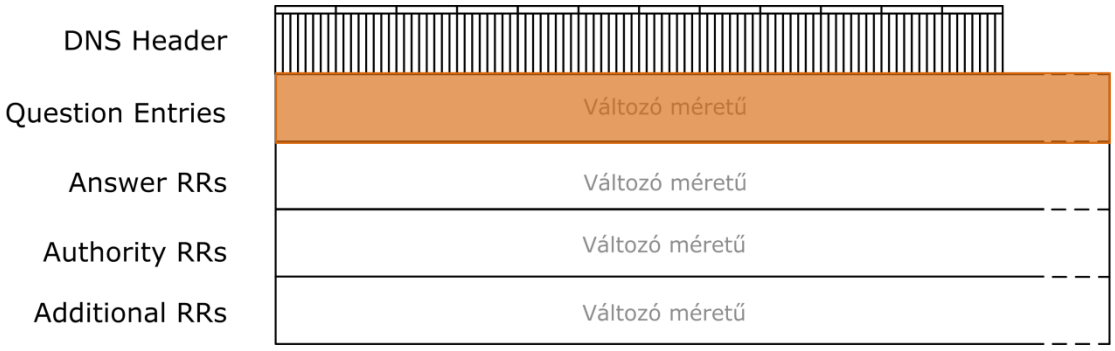
QUESTION ENTRY COUNT: Hány bejegyzés lesz a Question Entry mezőben.

ANSWER RR COUNT: Hány bejegyzés lesz az Answer RR mezőben.

AUTHORITY RR COUNT: Hány bejegyzés lesz az Authority RR mezőben.

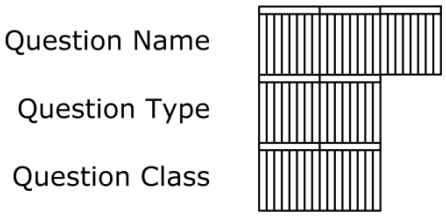
ADDITIONAL RR COUNT: Hány bejegyzés lesz az Additional RR mezőben.

6.2.1.2 Query Question Entries



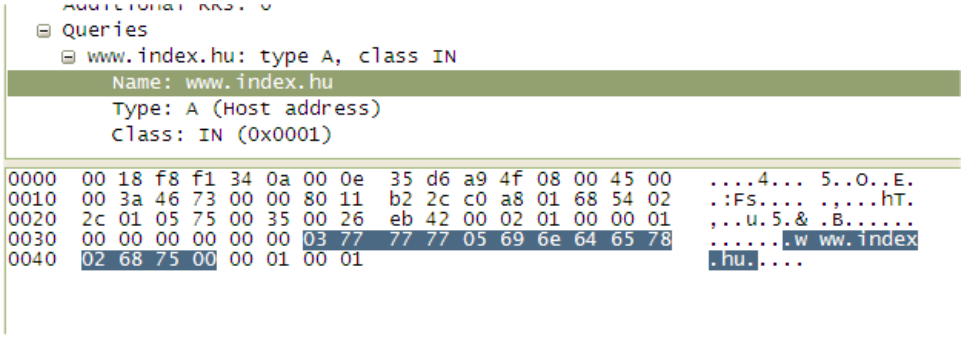
6.20. ábra Fókuszban a Question Entry

Valahogy így néz ki a mezőn belüli adatszerkezet.



6.21. ábra Question Entries

Question Name: Változó méretű mező. Itt utazik maga a kérdés. A cím, aminek a feloldására választ szeretnénk kapni. A kódolása... minimum bájos.



6.22. ábra Question Name

A fenti képen az látszik, hogy megkérdeztem a DNS szerveremet, vajon mi a www.index.hu host IP címe. Eddig rendben is van... de vessünk már egy pillantást a Packet Bytes részre! Hogy is van ez?

Így. A sorozat első bájtja az első cimke hosszát jelzi. Jelen esetben ez a www, azaz három bájt. Láthatod is: 03 77 77 77, ahol a hexadecimális 77 (decimálisan 119) a 'w' betű ASCII kódja. A következő cimke öt betű - index - , tehát 05 69 6e 64 65 78 lesz a kód. Végül hasonló logikával jön a vége: kétbetűs cimke - hu - azaz 02 68 75.

QUESTION TYPE: Két bájt. A Resource Record tipusa.

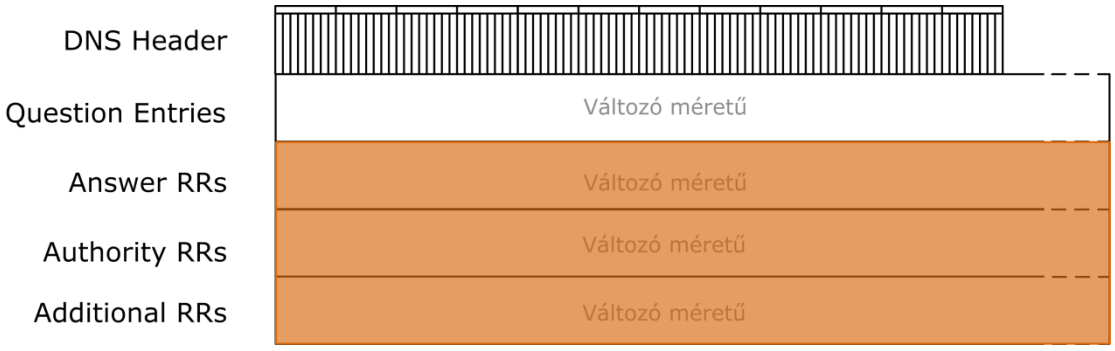
6.8. TÁBLÁZAT

Tipus	Név
0x01	Host Address (A) rekord
0x02	Name Server (NS) rekord
0x05	Alias (CNAME) rekord
0x06	Start of Authority (SOA) rekord
0x0C	Reverse-lookup (PTR) rekord
0x0F	Mail Exchanger (MX) rekord
0x1C	IPv6 host (AAAA) rekord
0x21	Server Selection (SRV) rekord
0xFB	Incremental Zone Transfer (IXFR) rekord
0xFC	Standard Zone Transfer (AXFR) rekord
0xFF	All records

A fenti képen (*6.22. ábra Question Name*) látható (00 01), hogy 'A' rekordot kértem le.

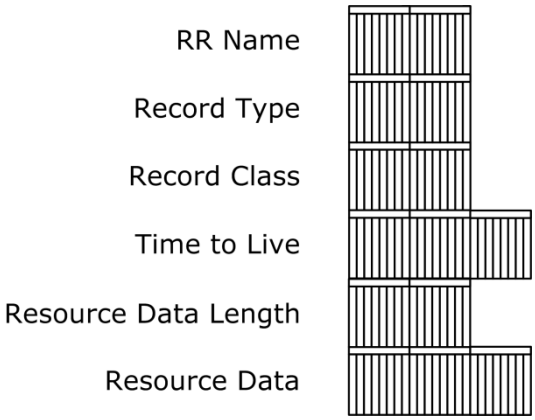
Question Class: Két bájt. DNS esetében mindig 1. Ez jelzi az (IN) class-t.

6.2.1.3 Query Erőforrás rekordok



6.23. ábra Fókuszban az erőforrás rekordok

Mivel formailag teljesen megegyeznek, így nincs értelme külön tárgyalni ezt a három mezőt.

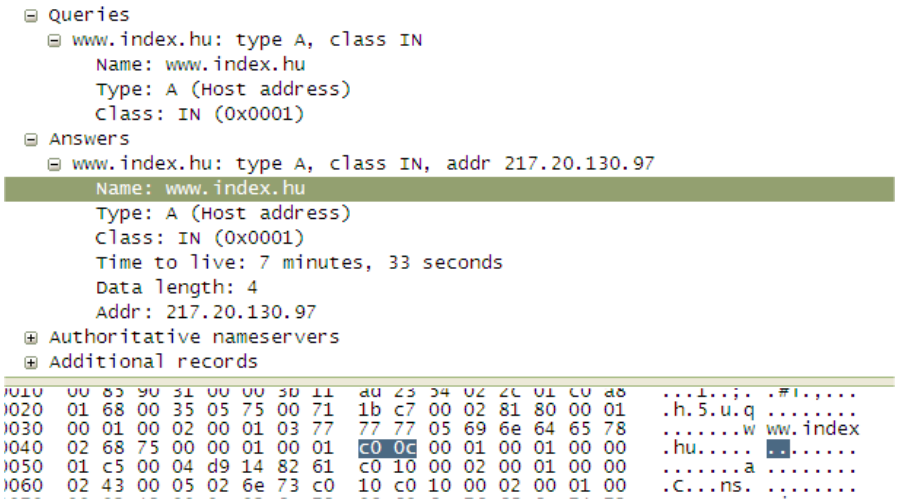


6.24. ábra DNS RR formátum a Query Response csomagban

RR Name: A feloldott név FQDN-je. Bár van, amikor kicsit trükkös maga az adat tárolása.

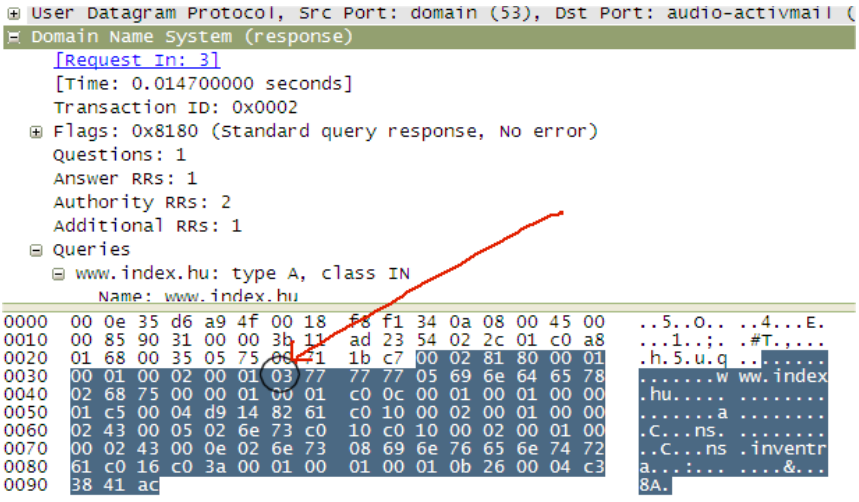
Nézzük csak meg az alábbi ábrát! Mutatni azt mutatja a Wireshark, hogy a visszaadott FQDN a www.index.hu - de a Packet Bytes részben csak annyit látunk, hogy C0 0C. Barátom, micsoda tömör tárolás már!

De hogyan lehet ezt visszakódolni? Önmagában sehogy. Ez ugyanis jelen esetben egy pointer és azt mutatja meg, hogy csomagon belül hol szerepel már egyszer ez a név. Hol is? Nem sokkal felette. A Query blokkján belül a Question Name mezőben. Méghozzá úgy, hogy a hosszát sem kell ismernünk, hiszen minden cimke előtt ott van a konkrét bájtok száma.



6.25. ÁBRA RR NAME MEZŐ TÁROLÁSA

Akik szeretnek matekozni, azokat valószínűleg érdekelni fogja, miért pont c0 0c lett ez a pointer.
Ez ugye két bájt. Írjuk fel bináris formában:
11000000 00001100
Az első két bit jelzi, hogy ez pointer lesz. Azokat le is kaphatjuk a tábláról.
000000 00001100
Mennyi is ez decimálisban? 12.



6.26. ÁBRA RR NAME TÖMÖRÍTVE

Kezdjünk el számolni. Jelen esetben a bemeszelt rész a DNS csomag, azon belül kell megkapnunk a Question Name kezdő bájtjának a számát. Aszongya, 1, 2, 3, ..., 13.
Izé.
Nekünk meg 12 jött ki.

Számoljuk újra, de most már úgy, hogy a nulla is játszik[89]: 0, 1, 2, 3, ..., 12.
Így már jó.

RECORD TYPE: A *6.8. táblázat* alapján megadott kód.

RECORD CLASS: Ugyanúgy 1, azaz (IN), mint a kérdésnél.

TIME TO LIVE: Ennyi ideig lehet cache-ben tárolni a feloldott értéket.

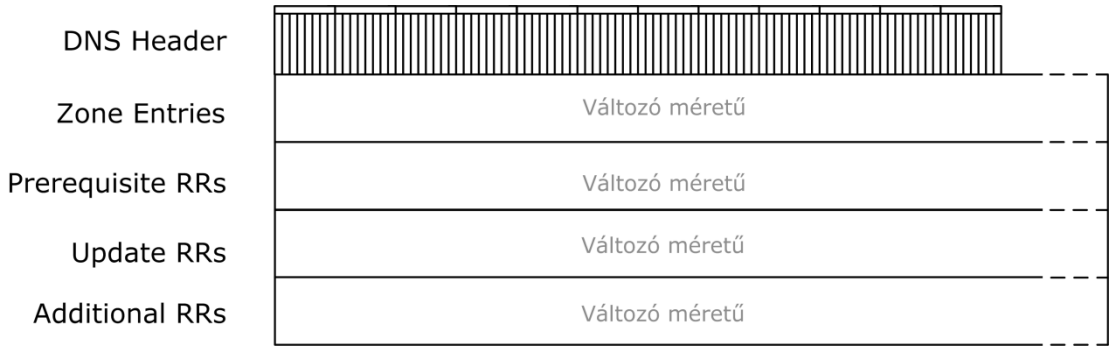
RESOURCE DATA LENGTH: A válaszként visszaadott érték mérete.

RESOURCE DATA: A válaszként visszaadott érték.

[89] Informatikusok vagyunk, vagy mi a szösz?

6.2.2 DNS UPDATE ÉS UPDATE RESPONSE ÜZENETEK

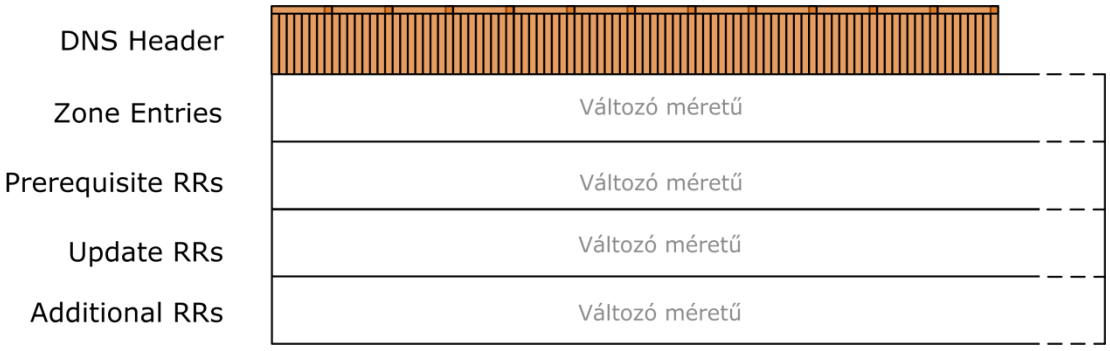
Eddig kérdezgettünk és válaszolgattak. Csakhogy azokat az értékeket be is kell rakni a DNS adatbázisába (zónafájl), sőt, karban is kell tartani.
Erről fog szólni a fejezet.



6.27. ÁBRA DNS UPDATE ÉS UPDATE RESPONSE CSOMAGSZERKEZETEK

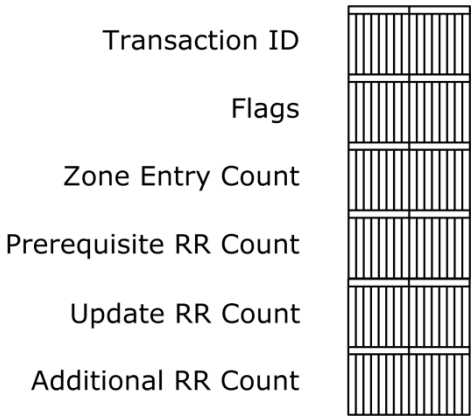
Egy dézsa vü, mi? Dehát jól bevált formán ne változtass, gondolhatták a tervezők.

6.2.2.1 DNS UPDATE ÉS UPDATE RESPONSE HEADER



6.28. ÁBRA DNS HEADER A FÓKUSZBAN

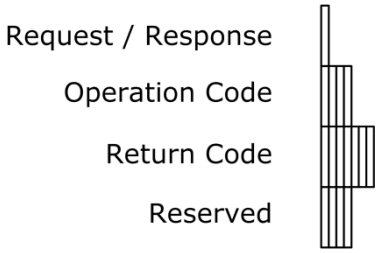
A fejléc szerkezete szintén nagyon hasonló.



6.29. ÁBRA DNS UPDATE ÉS UPDATE RESPONSE HEADER

TRANSACTION ID: Őt már ismerjük.

FLAGS: Itt már nincs olyan sok.



6.30. ÁBRA A FLAG MEZŐ SZERKEZETE A DNS UPDATE ÉS UPDATE RESPONSE FEJLÉCBEN

REQUEST / RESPONSE: Update (0), Response (1).

OPERATION CODE: Hasonló, mint a DNS Query Flag mezőjében, csak az Update kód értéke 5.

RESERVED: A jövőnek tartogatva.

RETURN CODE: A Response üzenetben jelzi, mi lett az akció végkimenetele.

6.9. TÁBLÁZAT

Return Code	Üzenet
0	Az update sikeres volt.
1	Format error: a DNS szerver nem tudott mit kezdeni a kéréssel
2	A DNS szerver begolyózott.
3	A név, amelyiknek léteznie kellene, nem létezik.
4	A DNS szerver nem tudja értelmezni a megadott Operation Code-ot.
5	A DNS szervernek nincs kedve elvégezni a kért névfeloldást.
6	A név, melynek nem lenne szabad léteznie, létezik.
7	Egy RR set, amelynek nem lenne szabad léteznie, létezik.
8	Egy RR set, amelynek léteznie kellene, nem létezik.
9	A DNS szerver nem autoritatív a kért zónára nézve.
10	A név, melyet megadtunk a Prerequisite illetve Update mezőkben, nem található meg a Zone Entries mezőben.

A FLAGS mező ismertetése után térjünk vissza a header mezőihez.

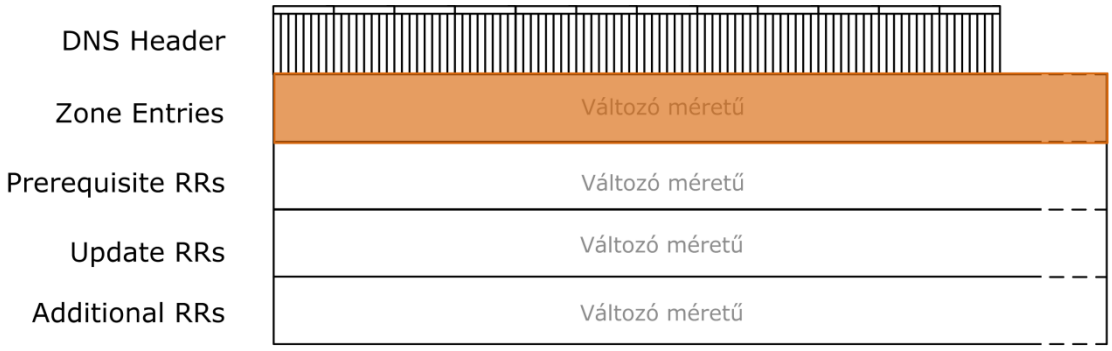
ZONE ENTRY COUNT: A Zone Entry mezőben az elemek száma.

PREREQUISITE RR COUNT: A Prerequisite RR mezőben az elemek száma.

UPDATE RR COUNT: Az Update RR mezőben az elemek száma.

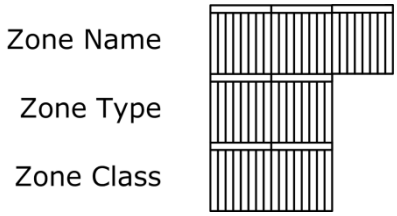
ADDITIONAL RR COUNT: Az Additional RR mezőben az elemek száma.

6.2.2.2 UPDATE ZONE ENTRIES MEZŐ



6.31. ÁBRA FÓKUSZBAN A ZONE ENTRIES MEZŐ

A mező adatszerkezete a következőképpen néz ki.



6.32. ÁBRA ZONE ENTRY ADATSZERKEZET

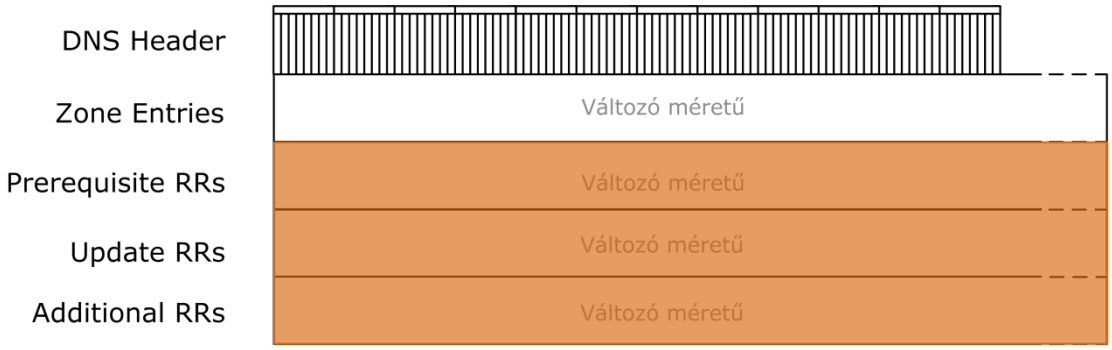
ZONE NAME: Változó méretű mező, a zóna teljes neve. Az adatok ábrázolása a korábban megismert módon (szám, cimke, szám, cimke, stb...) történik.

ZONE TYPE: Fix 6-os. A *6.8. táblázat* szerint ez a SOA kódja.

ZONE CLASS: 1, azaz (IN).

6.2.2.3 UPDATE ERŐFORRÁS REKORDOK

Ez a pálya.



6.33. ÁBRA ERŐFORRÁSREKORDOK A FÓKUSZBAN

RFC 2136

A Prerequisite Resource Records mezőben tulajdonképpen erőforrás rekordokra vonatkozó előfeltételeket adhatunk meg. Ezt úgy kell érteni, hogy az update csak akkor történhet meg, ha ezen feltételek mindegyike igaz.
A feltételek konkrét megadása nem könnyen látható át. Ebbe a mezőbe ugyanis Resource Record-okból álló set-ek kerülnek, méghozzá a normálistól némileg eltérő feltöltéssel. (A Resource Record felépítését az alábbi ábra mutatja: *6.24. ábra DNS RR formátum a Query Response csomagban*.) Attól függően, hogyan variálom meg a feltöltési szintaxist, öt feltételtipust használhatunk:

- Az erőforrás rekord csomagban lévő erőforrás rekordok közül legalább az egyik létezik a szerveren.
- Az erőforrás rekord csomagban lévő összes erőforrás rekord létezik a szerveren.
- Az erőforrás rekord csomagban lévő egyik erőforrás sem létezik a szerveren.
- A megadott névvel legalább egy erőforrás rekord létezik a szerveren.
- A megadott névvel egy erőforrás rekord sem létezik a szerveren.

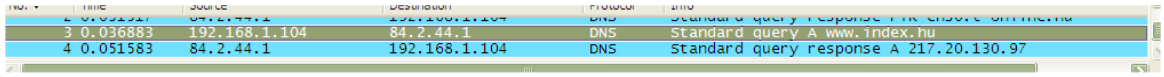
Az Update Resource Records mezőben azokat az erőforrás rekordokat találjuk, melyeket vagy hozzá kell adni az érintett zónákhoz, vagy ki kell törölni belőlük.

Az Additional Resource Records mezőben pedig az update-hez tartozó egyéb erőforrás rekordok utaznak.

6.2.3 DNS FOLYAMATOK

A száraz felsorolások után jöjjenek a jóval izgalmasabb példafolyamatok. Kezdjük a legsűrűbben használt folyamattal, a névfeloldással.

6.2.3.1 NÉVFELOLDÁS



6.34. ÁBRA DNS QUERY

Ez ilyen egyszerű dolog. Egy csomag oda, egy csomag vissza.

6.10. TÁBLÁZAT

No.	Time	Source	Destination	Protocol	Info
3	0.036883	192.168.1.104	84.2.44.1	DNS	Standard query A www.index.hu

```
Frame 3 (72 bytes on wire, 72 bytes captured)
Ethernet II, Src: Intel_d6:a9:4f (00:0e:35:d6:a9:4f), Dst: Cisco-Li_f1:34:0a
(00:18:f8:f1:34:0a)
Internet Protocol, Src: 192.168.1.104 (192.168.1.104), Dst: 84.2.44.1 (84.2.44.1)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 58
    Identification: 0x4673 (18035)
    Flags: 0x00
    Fragment offset: 0
    Time to live: 128
    Protocol: UDP (0x11)
    Header checksum: 0xb22c [correct]
    Source: 192.168.1.104 (192.168.1.104)
    Destination: 84.2.44.1 (84.2.44.1)
```

Az IP fejlécben túl sok érdekesség nincs, az én IP címem 192.168.1.104, a DNS szerveremé pedig 84.2.44.1.

```
User Datagram Protocol, Src Port: audio-activmail (1397), Dst Port: domain (53)
    Source port: audio-activmail (1397)
    Destination port: domain (53)
    Length: 38
    Checksum: 0xeb42 [correct]
```

Ezt egy kicsit benézte a Wireshark, de nem ezért szeretjük. Tudjuk, hogy a kliens hogyan kapcsolódik a szerverhez: forrásként ún. kliens portot (1024-65535) használ, címzettként meg az ún well-known portot, mely a protokoll RFC-jében kerül ajánlásra. Az UDP 53 a szerver oldalon rendben is van - de a kliens oldalon lévő, véletlenszerűen kiválasztott user port pont megegyezik egy ritkán használt

protokoll ajánlott portszámával. Így "ismerte" fel a program audio-activmail kommunikációnak a névfeloldási kérelmet.

```
Domain Name System (query)
    [Response In: 4]
    Transaction ID: 0x0002
```

Ezt generálta a kliensem kétbájtos tranzakcióazonosítónak. Nem erőltette meg magát.

```
    Flags: 0x0100 (Standard query)
        0... .... .... .... = Response: Message is a query
        .000 0... .... .... = Opcode: Standard query (0)
        .... ..0. .... .... = Truncated: Message is not truncated
        .... ...1 .... .... = Recursion desired: Do query recursively
        .... .... .0.. .... = Z: reserved (0)
        .... .... ...0 .... = Non-authenticated data OK: Non-authenticated data is
unacceptable
```

Semmi extra, a korábbi fejezet alapján minden flag könnyen érthető.

```
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
```

Egy címet szeretnék feloldani. Az RR mezők logikusan üresek, hiszen ezek a válaszok számára vannak fenntartva.

```
    Queries
        www.index.hu: type A, class IN
            Name: www.index.hu
            Type: A (Host address)
            Class: IN (0x0001)
```

A jó hosszú bevezető után lássuk végre magát a kérdést is. A feloldandó név a www.index.hu (emlékszünk még arra a bájos kódolásra), ez egy 'A' rekord, méghozzá az IN osztályban.

Jöhet a válasz.

6.11. TÁBLÁZAT

No.	Time	Source	Destination	Protocol	Info
4	0.051583	824.2.44.1	192.168.1.104	DNS	Standard query response A 217.20.130.97

```
Frame 4 (147 bytes on wire, 147 bytes captured)
Ethernet II, Src: Cisco-Li_f1:34:0a (00:18:f8:f1:34:0a), Dst: Intel_d6:a9:4f
(00:0e:35:d6:a9:4f)
Internet Protocol, Src: 84.2.44.1 (84.2.44.1), Dst: 192.168.1.104 (192.168.1.104)
User Datagram Protocol, Src Port: domain (53), Dst Port: audio-activmail (1397)
    Source port: domain (53)
    Destination port: audio-activmail (1397)
    Length: 113
    Checksum: 0x1bc7 [correct]
```

Igen, még mindig audio-activmail.

```
Domain Name System (response)
    [Request In: 3]
    [Time: 0.014700000 seconds]
    Transaction ID: 0x0002
```

Azonosító: stimmel.

```
    Flags: 0x8180 (Standard query response, No error)
        1... .... .... .... = Response: Message is a response
        .000 0... .... .... = Opcode: Standard query (0)
        .... .0.. .... .... = Authoritative: Server is not an authority for domain
        .... ..0. .... .... = Truncated: Message is not truncated
        .... ...1 .... .... = Recursion desired: Do query recursively
        .... .... 1... .... = Recursion available: Server can do recursive queries
        .... .... .0.. .... = Z: reserved (0)
        .... .... ..0. .... = Answer authenticated: Answer/authority portion was not
authenticated by the server
        .... .... .... 0000 = Reply code: No error (0)
```

Megint nincs semmi érdekes a zászlócskák környékén.

```
    Questions: 1
    Answer RRs: 1
    Authority RRs: 2
    Additional RRs: 1
```

Itt viszont hirtelen megszaporodtak az adatok.

```
    Queries
        www.index.hu: type A, class IN
            Name: www.index.hu
            Type: A (Host address)
            Class: IN (0x0001)
    Answers
        www.index.hu: type A, class IN, addr 217.20.130.97
            Name: www.index.hu
            Type: A (Host address)
            Class: IN (0x0001)
            Time to live: 7 minutes, 33 seconds
            Data length: 4
            Addr: 217.20.130.97
```

Egyfelől láthatjuk az eredeti kérdésünket a Queries mezőben, de láthatjuk a választ - mit választ, válaszrengeteget - is az RR mezőkben. A legfontosabb: itt van a kért IP cím. Méghozzá precíz használati utasítással: egész pontosan 7 perc 33 másodpercig tarthatom bent a DNS gyorstáramban. (Emlékszünk? Itt szokták a nevet pointerrel kiváltani.)

```
Authoritative nameservers
    index.hu: type NS, class IN, ns ns.index.hu
        Name: index.hu
        Type: NS (Authoritative name server)
        Class: IN (0x0001)
        Time to live: 9 minutes, 39 seconds
        Data length: 5
        Name server: ns.index.hu
    index.hu: type NS, class IN, ns ns.inventra.hu
        Name: index.hu
        Type: NS (Authoritative name server)
        Class: IN (0x0001)
        Time to live: 9 minutes, 39 seconds
        Data length: 14
        Name server: ns.inventra.hu
```

Jönnek az extrák. Az Authority RR mezőből például megtudhatom, mely DNS szerverek tekinthetők autoritatívnak az index.hu zónára nézve.
(Ezeket lehet DOSolni[9091].)

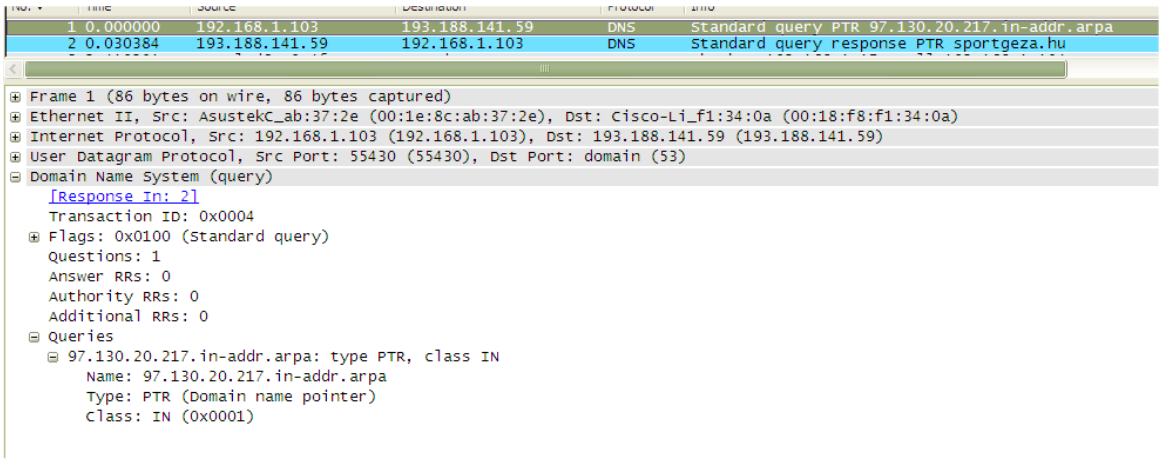
```
Additional records
    ns.index.hu: type A, class IN, addr 195.56.65.172
        Name: ns.index.hu
        Type: A (Host address)
        Class: IN (0x0001)
        Time to live: 18 hours, 59 minutes, 50 seconds
        Data length: 4
        Addr: 195.56.65.172
```

Ez pedig már az abszolút extra: az index.hu DNS szerverének 'A' rekordját is láthatjuk.

[90] Vicc volt.
[91] Rossz vicc.

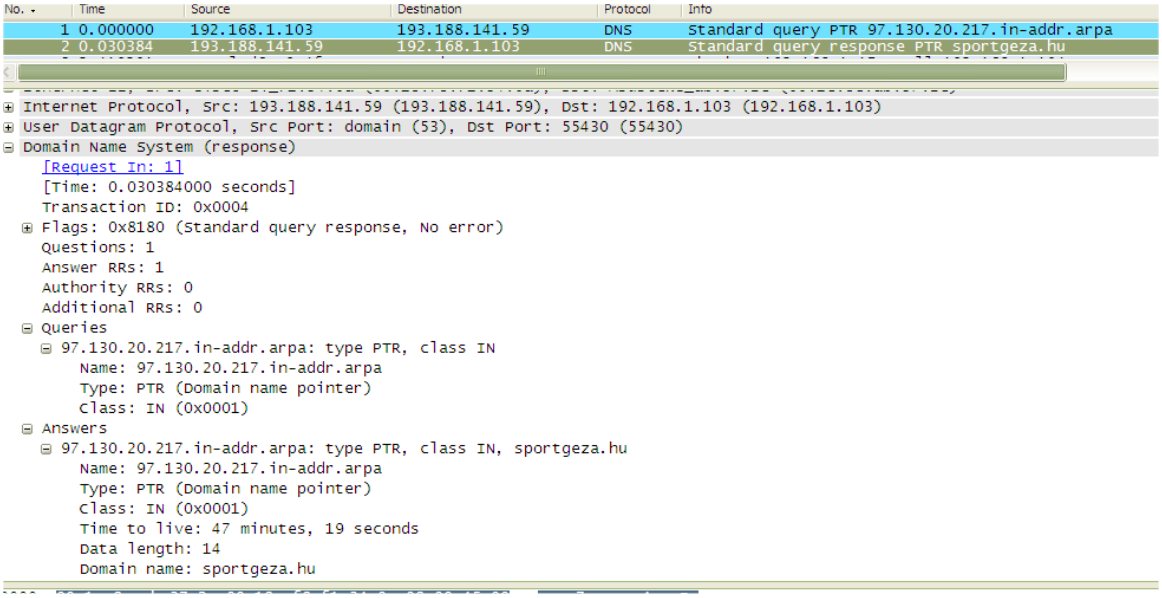
6.2.3.2 REVERZ NÉVFELOLDÁS

Ez az előző névfeloldás fordítottja: tudunk egy IP címet, de nem tudjuk, milyen névhez tartozik. Természetesen ezt is megkérdezhetjük. A folyamat teljesen hasonló, nem is mennék túlzottan részletesen bele.



6.35. ÁBRA REVERZ NÉVFELOLDÁS, QUERY

Látható, hogy név helyett a Question Name mezőben most egy IP cím, pontosabban egy IP címből képzett elnevezés áll. (Az egyes cimkék továbbra is ASCII kódokkal tárolódnak.) A típus is más egy kicsit, most nem 'A' rekordra vadászunk, hanem 'PTR'-re. De minden más stimmel.



6.36. ÁBRA REVERZ NÉVFELOLDÁS, QUERY RESPONSE

Szépen vissza is kaptuk, hogy az IP címhez tartozó reverz bejegyzés a sportgeza.hu. Némileg fura lehetne, mivel mi az index.hu címre számítottunk... de végülis nem az. Ilyesmi simán előfordul még a jobb családokban is.

6.2.3.3 HOGYAN CSINÁLJUNK HÜLYÉT MAGUNKBÓL?

A vége felé közeledünk, itt már egy kicsit elengedhetjük magunkat. Mindenki dőljön hátra a székében[92], helyezze magát kényelembe. Mesélni fogok.

Egyik ügyfelünk frissen beüzemelt alternatív levelezési kijáratával baj volt. Ha azon ment ki egy levél, akkor nagy valószínűséggel nem érkezett meg a címzetthez, ráadásul hibaüzenet sem érkezett vissza. Nem kicsit volt kínos, az ügyfél üzleti tevékenysége ugyanis erősen emailfüggő.

Általában mikor nem érkezik hibaüzenet? Ha a túloldalon lévő levelezőszerver szpemnek tekinti a küldeményt. Hülye lenne a szpemmert információkkal tömni.

De miért tekintettek minket szpemmernek? És miért csak néhányan?

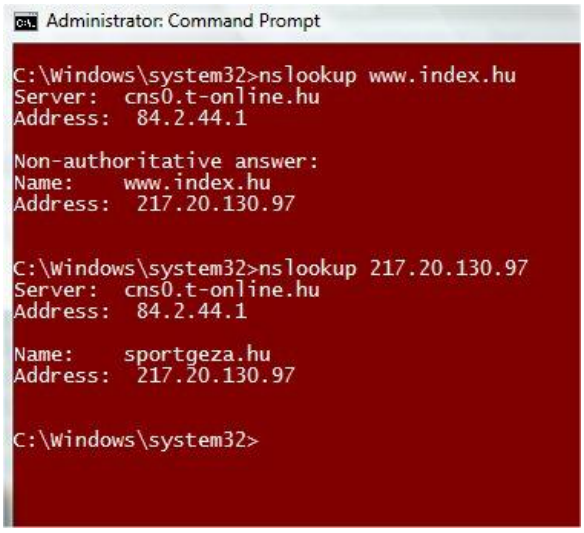
Némi nyomozás után kiderült, hogy azokkal a címzettekkel volt baj, ahol az ügyfelünk levelezési kijáratát reverz DNS vizsgálattal ellenőrizték. Ez abból áll, hogy megnézték, milyen IP címről érkezett meg a levél, rákérdeztek, hogy ehhez az IP címhez milyen FQDN tartozik - és ha találtak ehhez tartozó MX rekordot az ügyfél zónájában, akkor a levél nem volt szpem. Feltehetőleg.
Innentől már sínen voltam: elkezdtem tesztelni a kijáratra vonatkozó névfeloldást. Elkeserítő eredményt kaptam: mind a nagyvilágban, mind a magyar neten teljesen véletlenszerű volt, hogy mely DNS szervereknél működik a reverz lekérdezés és melyeknél nem.

Jobb híján összeállítottam egy terjedelmes leírást a jelenségről, elküldtem az ügyfél DNS szolgáltatójához. Nézzék meg, oldják meg... csináljanak valamit.

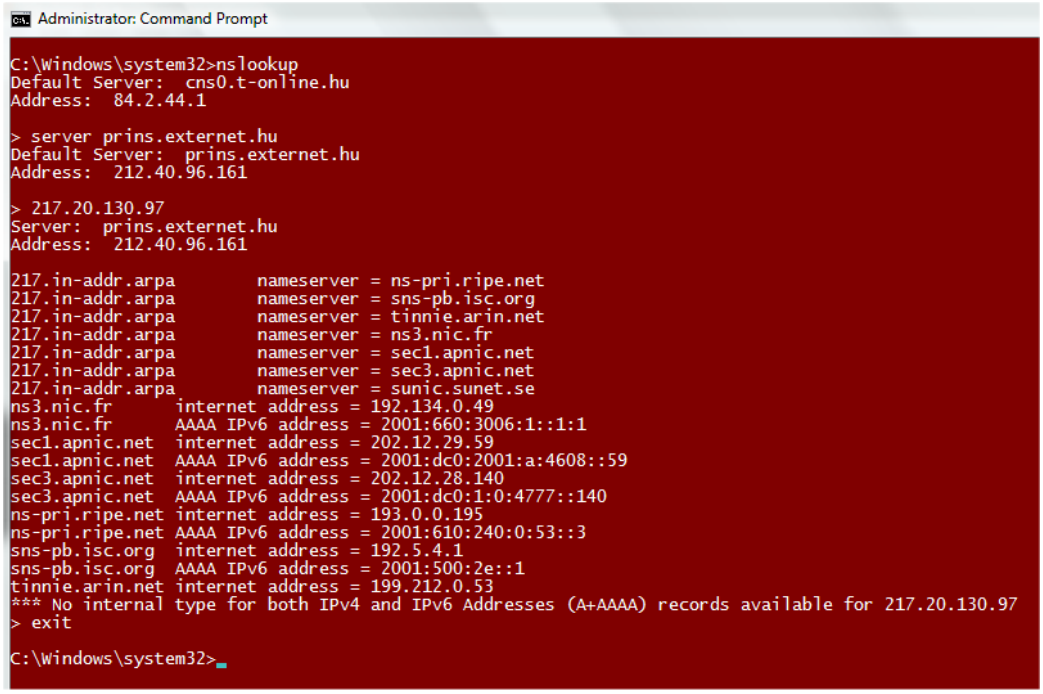
(A következő oldalon az index.hu példáján mutatom be a hülyeségemet. Természetesen az eredeti ügyfél _nem_ az index.hu volt. Jól is néznék ott ki a Windows-os tudásommal.)

[92] Aki ilyen derékgyógyász böszme gumilabdán ül, az inkább ne.

Idemásolom, hogy én mit láttam hibának.



6.37. ÁBRA EGY NORMÁLIS, ODA-VISSZA NÉVFELOLDÁS



6.38. ÁBRA EZ MEG MI?

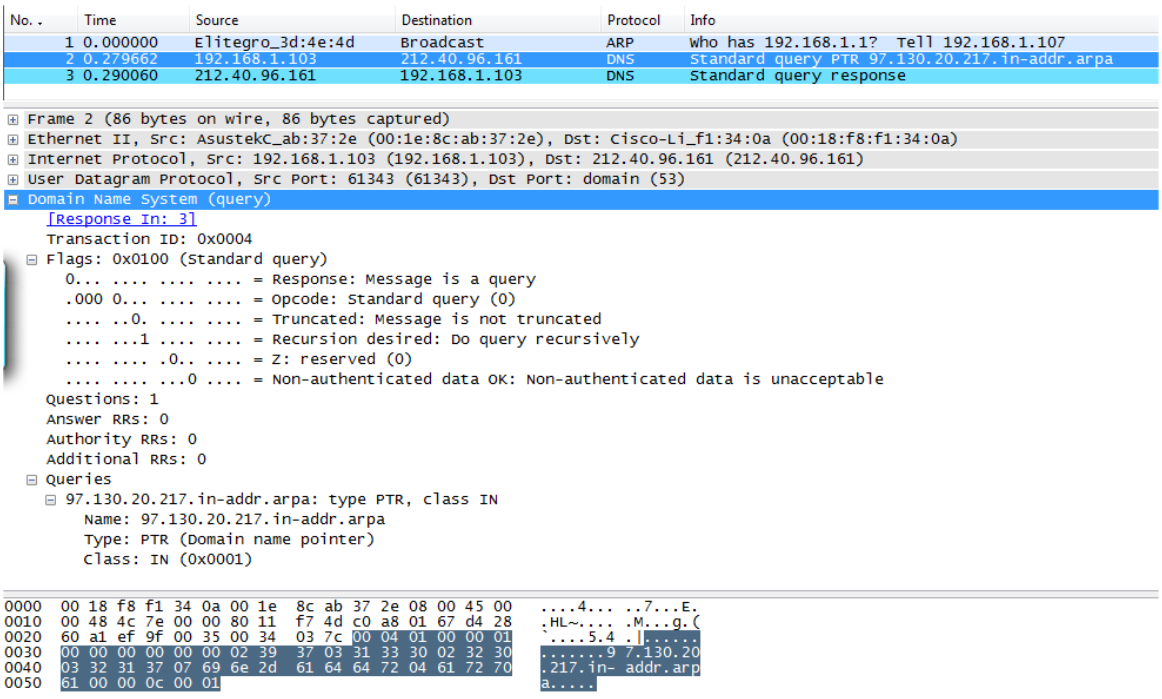
Véletlenszerűen kiválasztottam az Externet DNS szerverét, elkezdtem a reverz névfeloldást - és azt kaptam, hogy a DNS szerver szerint nincs ilyen rekord.

- Ez az! - gondoltam - Most megvagy!

A szolgáltató visszahívott, hogy ő nem látja a problémát. Mit nem lát? - jöttem zavarba - világosan ott van, hogy "no records"? Az nem számít, legyintettek rá. Aztán turkáltak valamit, javítgattak valamit[93], majd azt mondták várjak pár hetet, oszt jó lesz.

Eltelt pár hét. Lefuttattam a fenti próbát, ugyanúgy elhasalt. Ekkor írtam egy kedves, de azért kicsit vitriolos levelet, hogy ez még mindig nem jó. Erre a szolgáltató kedvesen ugyan, de elküldött a wikipedia DNS-sel foglalkozó oldalára. Engem. Aki éppen hálózati szolgáltatásokról is szóló könyvet írt.
Forrt bennem a harag. A levelezőkliensem Draft folderében már formálódott a válaszlevél, mely már egyáltalán nem volt kedves, cserébe egy kicsivel több vitriol jutott bele.

Itt jártunk, amikor elkezdtem mélyebben foglalkozni jelen fejezettel. És megtaláltam. Amikor megértettem, nem győztem szégyenemben a pokróc alá bújni.

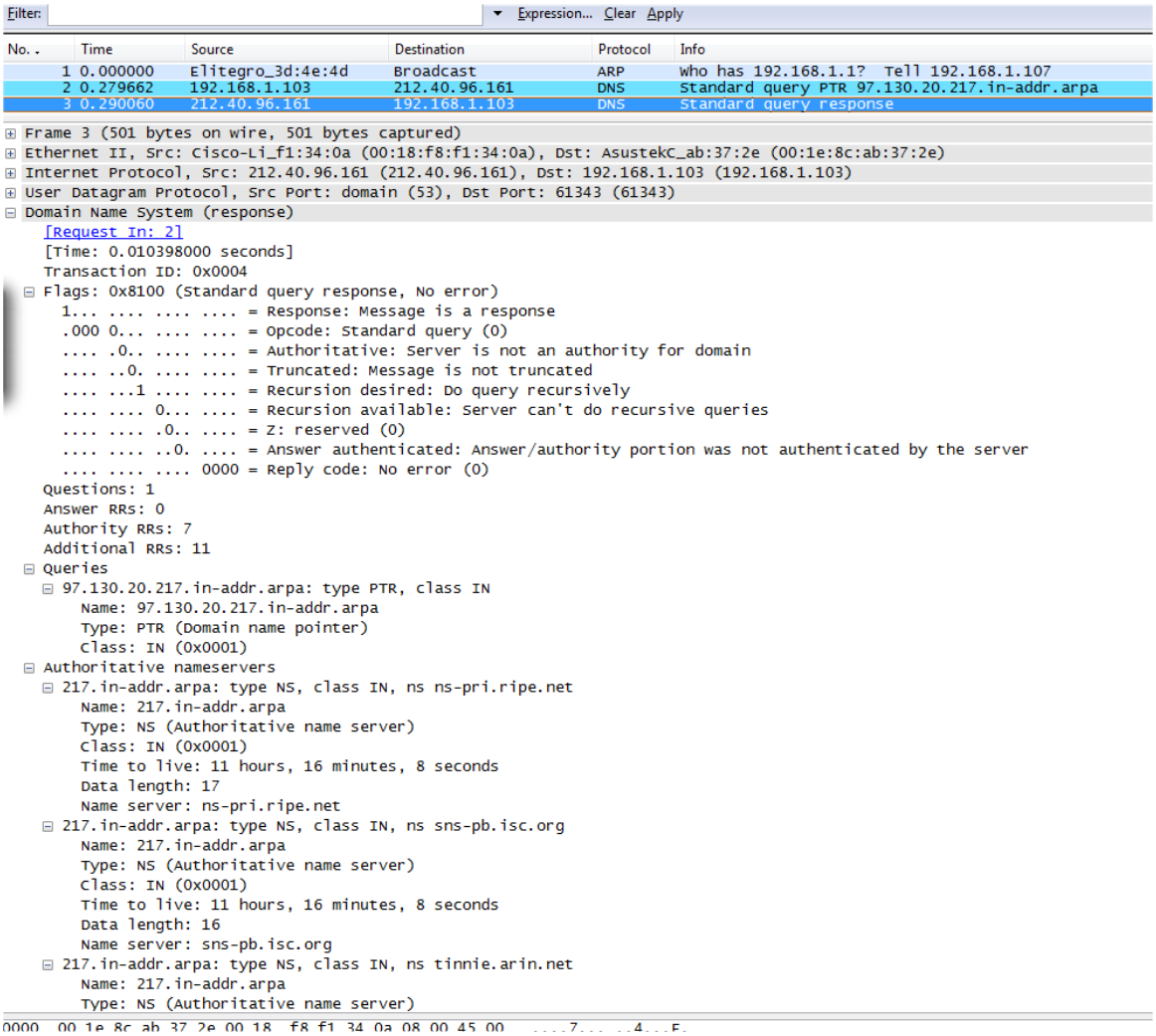


6.39. ÁBRA 2. PRÓBÁLKOZÁS, QUERY

Tessék megnézni az ábrát. Ez ugye az ún. sikertelen feloldáshoz vezető folyamat első lépése, a lekérdezés. Az IP cím, akinek küldtem, az a prins.externet.hu, a lekérdezett IP cím, amelyhez a nevet kerestem, az a 217.20.130.97. A flag-ek alapján rekurziót kértem (recursion desired), azaz azt, hogy ha a feloldandó címhez tartozó zóna nincs

[93] Pontosan tudom, mit javítottak, de ennyire nem akarom kiadni az ügyfelet.

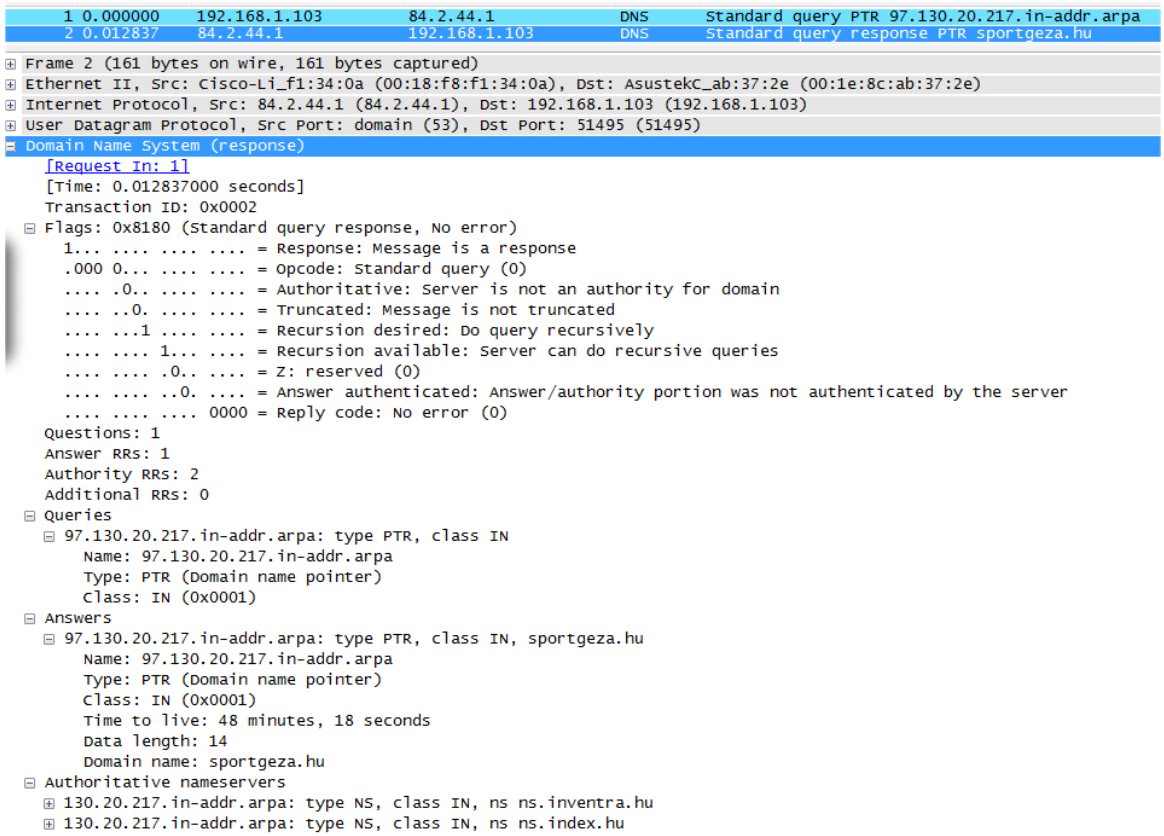
a megcélzott DNS szerveren, akkor legyen kedves, járjon már utána, hogy hol van és kérje el a nevemben a nevet.
Ehhez képest mi történt?



6.40. ÁBRA 2. PRÓBÁLKOZÁS, QUERY RESPONSE

Látsz a válaszban feloldott nevet? Nem. A Queries mező után rögtön az Authority RR mező jön, nincs sehol Answer RR. Miért nincs? A flag-eket kell megnézni. Ott áll feketén-fehéren a Recursion Available flag-ben, hogy "Server can't do recursive queries". Ezt a DNS szervert úgy konfigurálták, hogy ne dolgozzon más helyett. Ha rekurzív lekérdezésre utasítják, akkor visszadobja, hogy ő merre indulna el - és ennyi. Ez az infó van az Authority RR mezőben és ezt az infót dobja fel az ábrán (*6.38. ábra Ez meg mi?*) is.

Nézzünk egy ellenpróbát. Látható, hogy a nagy magyar interneten volt olyan DNS szerver is, mely képes volt feloldani az ügyfél címét. (*6.37. ábra Egy normális, oda-vissza névfeloldás*)



6.41. ÁBRA A SIKERES QUERY RESPONSE

Amikor ezt a szervert kérdeztem, akkor már teljesen más lett a helyzet. A Recursion Available flag értéke 1, azaz "Server can do recursive queries". És meg is csinálta, láthatod, ott van a válasz az Answer RR mezőben.

Nos, ennyi. Amikor én elkezdtem játszogatni akár a magyar, akár a külföldi DNS szerverekkel, akkor attól függően, hogy a helyi rendszergazda hogyan konfigurálta azokat, vagy kaptam jó választ (ekkor a szerver jó fiú volt és megkereste helyettem az értéket a neten) vagy nem (ekkor csak az elinduláshoz kaptam segítséget). Nyilván ha én DNS szerver lettem volna, akkor az utóbbi esetben nekem kellett volna utánamennem az információnak. De mivel nem az voltam, hanem csak egy hétköznapi, szomorú Command prompt, így a történet azzal végződött, hogy "no records".

Baromi nagy mázli, hogy azt a durván paprikás levelet végül nem küldtem el. A szolgáltatónak ugyanis maximálisan igaza volt.

6.2.4 DNS AZ IPV6-BAN

Tudom, vannak olyan rendszergazdák, akik fejből tudják a cégüknél előforduló összes IP címről, hogy melyik kié. Amikor kicsi (200-250 fős) cégnél dolgoztam, ahol mindenkinek fix IP címe volt, mi is tudtuk, mekkora prioritással kellett kirontani az IT szobából, ha IP conflict üzenet ugrott fel.

De... belegondoltál már? Egy IPv6 cím négyszer olyan hosszú.
Mondjuk a jelenlegi gépemé ilyen: *fe80:0:0:0:21e:8cff:feab:372e*. Meg se próbálom decimálisban leírni. Ki fogja ezeket fejből tudni?

A DNS szerver. És rajta kívül más senki.
Ezért kiemelten fontos, hogy jól működjön.

Tegyük fel, hogy jól működik. Akkor hogyan fog kinézni benne egy IPv6 bejegyzés? Hogyan fog beleférni 16 bájt a 4 bájtnyi rubrikába?
Sehogy.

RFC 1886

Be lett vezetve az IPv6 címek számára egy új rekordtípus, a quadA, azaz AAAA rekord. (Azért 4 darab 'A' mert a mérete pont négyszerese az IPv4-ben lévő 'A' rekord méretének.)
Ha nálam lenne itthon IPv6-ra felokosított DNS szerver, így nézne ki a bejegyzésem:

```
hq.petrenyi.local   IN   AAAA      fe80::21e:8cff:feab:372e
```

Oké. És mi a helyzet a reverz névfeloldással. Belegondoltunk? Szürkül már a tekintetünk?

Hogy is nézett ez ki IPv4 alatt?

IP cím	: 192.168.1.103
A reverz zóna neve	: 1.168.192.in-addr.arpa
Bejegyzés	: `103 IN PTR hq.petrenyi.local`
A gépnév	: 103.1.168.192.in-addr.arpa

Az a reverz zóna név, az fog itt problémákat okozni. Először is, IPv6 esetében nem fogunk visszaváltani decimálisba. Az olyan gyerekes.

Először felírjuk az IP címet, teljesen kibontva:
`F.E.8.0.0.0.0.0.0.0.0.0.0.0.0.0.2.1.E.8.C.F.F.F.E.A.B.3.7.2.E`
Aztán keresünk egy fix pontot, majd kiforgatjuk a sarkából:
`E.2.7.3.B.A.E.F.F.F.C.8.E.1.2.0.0.0.0.0.0.0.0.0.0.0.0.8.E.F`
És akkor már csak az ismerős farkat kell a végére csapni:
`E.2.7.3.B.A.E.F.F.F.C.8.E.1.2.0.0.0.0.0.0.0.0.0.0.0.0.8.E.F.IP6.ARPA`
Ez lesz a számítógépem ~~perverz~~ reverz neve IPv6-ban. Barátságos, mi?

Még egy apróságról szeretnék szót ejteni. Ha én egy DNS szerverhez fordulok, feloldandó egy nevet, a szerver honnan fogja tudni, hogy én A vagy AAAA rekordot szeretnék visszakapni? Úgy, hogy külön jelzem neki a Question Entries mezőben, ha kell az AAAA rekord is. Méghozzá a megfelelő helyen (*6.8. táblázat - Question Type értékek*) egy pontosan beszúrt 0x1C (AAAA) illetve 0xFF (Any) értékkel.

7 KIVEZETÉS

Minden normális könyvben arra szokta buzdítani a szerző a kedves olvasót, hogy ha kérdése van, akkor ne hezitáljon, tegye föl bátran.
Ez ilyen szempontból is rendhagyó könyv. Én azt mondom, hogy eszed ágában se legyen tőlem bármit is kérdezni. Nem azért, mert én egy undok vadmalac vagyok... hanem azért, mert ez egy alulról korlátos könyv.

Elmagyarázom.

Aki van annyira bátor, hogy könyvet ír, annak nagyon kell tudnia. Ezt a tudást legtöbbször nem is tudja teljesen kiírni magából. Már nyomdában van a könyv, amikor eszébe jut, hogy ez is kimaradt, meg az se lett teljesen kibontva. Azaz ha bárki kérdez valamit, ami nem került bele a könyvbe, a szerző magától értetődően el tudja magyarázni.
Ez a felülről korlátos könyv. Ilyen volt az Exchange 2007-ről szóló könyvem.

Ennek a mostani könyvnek már a legelején jeleztem, hogy rendhagyó kísérletre készülök. Olyan területről írok, amelyhez nem értek. Pontosabban értek valamennyire, de messze nem annyira, mint amilyen mélyen el akarok merülni a témában. Menetközben tanulok... és a megértett dolgokat gyermeki örömmel adom át, mint megannyi újdonságot.
Leírtam mindent, amit tudok... sőt valójában többet is írtam le, mint amit ténylegesen tudok. Hiszen a keretek és datagramok szerkezetét, a szabványok lényegét én is kézikönyvekből, weblapokról szedtem össze.
Ergo ez egy alulról korlátos könyv.

Kérdezni kérdezhetsz, persze... de én is csak annyit tudok tenni, amennyi neked is rendelkezésedre áll: gugli vagy bing.

Ellenben ha te a téma szakértője vagy és csak a viccek miatt olvastad végig a könyvet, aztán úgy találtál benne ordító hibákat - nos, te *ne tétovázz*, írd meg ezeket egyből a jpetrenyi@gmail.com címre. Az online megjelenésnek ugyanis pont az az óriási előnye, hogy a tartalom módosítható, aktualizálható.

A végére egy jó hír: a könyv működik. Amikor gyűjtöttem az anyagot, szép lassan kezdett érthető lenni a TCP/IP működése. Amikor írtam a könyvet, már azt hittem, hogy értem - csak ekkor még darabokban láttam mindent, hiszen ebben a fázisban a részletek kidolgozása volt a jellemző. És amikor lektoráláskor olvastam el egyben az egészet, akkor állt csak össze a teljes kép. De összeállt.

8 FORRÁSOK, LINKEK

Ez a könyv úgy készült, hogy elolvastam hozzá RFC-ket, izmos kéziratokat - majd ahol úgy éreztem, hozzáolvastam a netről is. Rengeteg link gyűlt össze a végére. Ezeket a linkeket találjátok meg a ebben a fejezetben, minimálisan strukturálva.

SZAKKÖNYVEK:

Alapvetően ezekre az írott forrásokra támaszkodtam:

- Joseph Davies: Windows Server 2003 TCP/IP Fundamentals for Microsoft Windows
- Joseph Davies: Windows Server 2008 TCP/IP Fundamentals for Microsoft Windows
- Joseph Davies, Tony Northrup with the Microsoft Networking Team: Windows Server 2008 Networking and Network Access Protection (NAP)
- Joseph Davies: Windows Server 2008 TCP/IP Protocols and Services
- Joseph Davies: Understanding IPv6

RFC

A tiszta forrás, ahol minden megvan:

- http://tools.ietf.org/html/
- http://www.rfc-editor.org/rfc.html
- http://www.networksorcery.com/enp/
- http://en.wikipedia.org/wiki/Request_for_Comments

KIEMELT LINKEK:

TCP/IP Guide:

http://www.tcpipguide.com

Internetworking Technology Handbook:

http://www.cisco.com/en/US/docs/internetworking/technology/handbook/ito_doc.html

TCP/IP Networks:

http://www.citap.com/documents/tcp-ip/tcpip001.htm

Protocols Directory:

http://www.protocols.com/protocols.htm

Internetworking Guide:

http://technet.microsoft.com/en-us/library/cc951210.aspx

WINDOWS SERVER 2008 ÉS VISTA

New Networking Features in Windows Server 2008 and Windows Vista
http://technet.microsoft.com/en-us/library/bb726965.aspx
Windows Server 2008 Networking:
http://technet.microsoft.com/en-us/library/cc753940(WS.10).aspx
Windows Vista Networking Tecnologies:
http://en.wikipedia.org/wiki/Windows_Vista_networking_technologies

ÖMLESZTETT LINKEK

Végül ömlesztve egy csomó link abból a segédfájlból, ahová menetközben szórtam a hasznos infókat tartalmazó találatokat.

- http://www.itu.int/rec/T-REC-X/en
- http://en.wikipedia.org/wiki/Ethernet
- http://en.wikipedia.org/wiki/Frame_(telecommunications)
- http://www.iana.org/assignments/ieee-802-numbers
- http://www.erg.abdn.ac.uk/users/gorry/course/lan-pages/llc.html
- http://www.citap.com/documents/tcp-ip/tcpip007.htm
- http://www.techfest.com/networking/lan/token.htm
- http://www.javvin.com/protocolFDDI.html
- http://www.laynetworks.com/FDDI.htm
- http://en.wikipedia.org/wiki/Point_to_Point_Protocol
- http://www.cisco.com/en/US/docs/internetworking/technology/handbook/PPP.html
- http://www.tcpipguide.com/free/t_PointtoPointProtocolPPP.htm
- http://en.wikipedia.org/wiki/802.11
- http://hu.wikipedia.org/wiki/Wi-Fi
- http://hu.wikipedia.org/wiki/WPA
- http://wifi.cs.st-andrews.ac.uk/wififrame.html
- http://en.wikipedia.org/wiki/MAC_address
- http://en.wikipedia.org/wiki/Media_Access_Control
- http://www.leapforum.org/published/internetworkMobility/split/node12.html
- http://www.wireless-center.net/Wireless-Internet-Technologies-and-Applications/1923.html
- http://www.iana.org/assignments/arp-parameters/
- http://www.javvin.com/protocolARP.html
- http://blogs.technet.com/networking/archive/2009/01/15/unable-to-connect-to-windows-server-2008-nlb-virtual-ip-address-from-hosts-in-different-subnets-when-nlb-is-in-multicast-mode.aspx
- https://www.netacademia.net/tudastar/default.aspx?upid=2836
- http://en.wikipedia.org/wiki/Address_Resolution_Protocol
- http://sunsite.nus.sg/pub/slip/slip-vs-ppp.html
- http://www.iana.org/assignments/ppp-numbers
- http://en.wikipedia.org/wiki/Cryptographic_hash_function
- http://en.wikipedia.org/wiki/Challenge-handshake_authentication_protocol

- http://www.windowsnetworking.com/articles_tutorials/Windows-Server-2008-Networking-Services.html
- http://www.szabilinux.hu/trendek/trendek422.html
- http://www.protocols.com/pbook/frame.htm
- http://en.wikipedia.org/wiki/Frame_Relay
- http://en.wikipedia.org/wiki/Maximum_transmission_unit
- http://www.iana.org/assignments/ip-parameters
- http://en.wikipedia.org/wiki/Routing
- http://technet.microsoft.com/en-us/library/cc750576.aspx
- http://technet.microsoft.com/en-us/library/bb727001.aspx
- http://en.wikipedia.org/wiki/Routing_Information_Protocol
- http://en.wikipedia.org/wiki/Distance-vector_routing_protocols
- http://en.wikipedia.org/wiki/Open_Shortest_Path_First
- http://www.iana.org/assignments/icmp-parameters
- http://en.wikipedia.org/wiki/Ping_of_death
- http://www.tcpipguide.com/free/t_ICMPv4DestinationUnreachableMessages-3.htm
- http://www.networkcomputing.com/netdesign/1107icmp3.html?ls=NCJS_1107bt
- http://en.wikipedia.org/wiki/Path_MTU_discovery
- http://www.netheaven.com/pmtu.html
- http://en.wikipedia.org/wiki/ICMP_Redirect_Message
- http://www.networkdictionary.com/protocols/irdp.php
- http://www.javvin.com/protocolESIS.html
- http://tldp.org/HOWTO/Multicast-HOWTO-2.html
- http://en.wikipedia.org/wiki/Distance_Vector_Multicast_Routing_Protocol
- http://www.dataconnection.com/multicast/pimprotocol.htm
- http://www.javvin.com/protocolMOSPF.html
- http://en.wikipedia.org/wiki/Internet_Group_Management_Protocol
- http://technet.microsoft.com/en-us/library/cc957916.aspx
- http://technet.microsoft.com/en-us/library/cc957911.aspx
- http://www.mbone.net/
- http://en.wikipedia.org/wiki/Mbone
- http://acs.lbl.gov/OldMisc/mbone/
- http://www.cisco.com/en/US/tech/tk828/tech_brief09186a00800a4415.html
- http://en.wikipedia.org/wiki/Multicast_address
- http://www.networksorcery.com/enp/protocol/igmp.htm
- http://www.javvin.com/protocolIGMP.html
- http://www.iana.org/assignments/ipv6-multicast-addresses
- http://www.tcpipguide.com/free/t_UDPMessageFormat-2.htm
- http://en.wikipedia.org/wiki/User_Datagram_Protocol
- http://en.wikipedia.org/wiki/Transmission_Control_Protocol
- http://www.tcpipguide.com/free/t_TCPConnectionEstablishmentSequenceNumberSynchroniz.htm
- http://www.firewall.cx/tcp-analysis-section-2.php
- http://en.wikipedia.org/wiki/Windows_Vista_networking_technologies
- http://technet.microsoft.com/en-us/library/bb726965.aspx
- http://en.wikipedia.org/wiki/TCP_tuning
- http://en.wikipedia.org/wiki/Karn%27s_Algorithm

- http://www.cs.utk.edu/~dunigan/tcptour/javis/tcp_karn.html
- http://www.iana.org/assignments/bootp-dhcp-parameters/
- http://support.microsoft.com/kb/169289
- http://www.tcpipguide.com/free/t_DHCPLeaseRenewalandRebindingProcesses-2.htm
- http://www.tcpipguide.com/free/t_DHCPOptionsOptionFormatandOptionOverloading-2.htm
- http://www.tcpipguide.com/free/t_TCPIPAddressResolutionForIPMulticastAddresses.htm
- http://av.c3.hu/multicast/
- http://www.firewall.cx/multicast-intro.php
- http://www.tcpipguide.com/free/t_ProxyARP.htm
- http://www.cisco.com/en/US/tech/tk648/tk361/technologies_tech_note09186a0080094adb.shtml
- http://www.osischool.com/protocol/Tcp/slidingWindow/index.php
- http://www.osischool.com/protocol/Tcp

9 JAVÍTÁSOK

Habár az lenne a helyénvaló, hogy egyenként említsek meg mindenkit, aki észrevett valami hibát, vagy fogalmi zavart, de attól jelentősen olvashatatlan lenne ez a javításokat felsoroló rész. Így egyszerre szeretnék köszönetet mondani mindenkinek, aki hozzájárult ahhoz, hogy a könyv korrektebb és érthetőbb legyen.

9.1 2.0 VERZIÓ, 2010 MÁRCIUS

Na, itt aztán rendesen belenyúltam a könyvbe.

- Az összes link kattintható lett.
- A könyv tartalomjegyzéke fa struktúrában megjelenik a pdf fájl bal oldalán - és kattintható.
- Rengeteg átfogalmazás, olvashatóbbá tétel, modorosságok irtása.
- Elgépelések javítása.
- Az irodalmi idézet lecserélése.
- Bevezetésben arc visszavétele.
- A 2. fejezet átstrukturálása.
- Teljesen új 2.2 fejezet, ahová az alapfogalmak ismertetése került. Ezeket a szövegeket a korábbi előfordulási helyükről kiszedtem.
- A teljes szövegben javítottam a laza fogalomhasználatot, a 2.2 fejezetnek megfelelően.
- Ahogy fel is hívták rá a figyelmemet, a Wireshark alsó mezője minden, csak nem bináris. A hivatalos elnevezése Packet Bytes. Végig javítva.
- 21. oldal: A hub nem L2, hanem L1.
- 31 és 163 oldalak: A MAC Address I/G bitjének részletezése, a broadcast átsorolása.
- 36. oldal: Lábjegyzet: sávszélesség vs. adatátviteli sebesség.
- 37. oldal: Fontos lábjegyzet a broadcast és a collision domainekről.
- 60. oldal: A BOOTP megkövetése.
- 60-62. oldalak: A Proxy ARP mélyebb kifejtése.
- 117. oldal: Volt egy kis elmatekolás. Csodálkozom, hogy eddig senki nem vette észre.
- 131-132 oldalak: A NAT fajtáinak (NAT, PAT, SNAT, DNAT) rövid bemutatása.
- 132. oldal: Az APIPA hozzárakása a privát tartományokhoz.
- 178. oldal: NAT hátrányai kiegészítve.
- 180. oldal: Megint az a fránya matek. A 128 bájt az nem 2^{128} lehetőség.

- 187. oldal: A unique-local IPv6 cím képzésének bővebb kifejtése.
- 191. oldal: IPv6 címek összefoglaló táblázata. Szvsz meglehetősen fontos.
- 207. oldal: Megjegyzés a solicited IPv6 címhez.
- 210. oldal: Egyértelműbb fogalmazás az IPv6 stateless autokonfigurációnál.
- 215. oldal: A 4-63 ábra cseréje.
- 215-225. oldalak: Az IPv4-IPv6 konverziók gyökeres újraírása, konkrét példák, részletesebb magyarázatok.
- 225. oldal: A Teredo alfejezet bővítése a Windows 2008 Server R2 miatt.
- 239-287 oldalak: Már az oldalszámokból is láthatod, hogy ez volt az egyik legnagyobb ívű változtatás a könyvben. Célozgattam rá korábban is, hogy a legnehezebben megértett rész számomra a szegmensek ackolása, az adatfolyam szabályozása és az újraküldések optimalizálása volt. Ez be is jött, mert kaptam egy levelet, melyben szelíden elmagyarázták, mi mindent értettem meg rosszul ebben a témakörben. Innentől az én dolgom már csak annyiból állt, hogy a pontosabb megértés fényében újrafogalmazzam az érintett részeket. (A poénok maradtak.) Ezzel nem azt akarom mondani, hogy újraírtam 48 oldalt - csak azt, hogy elég sok helyen kellett belenyúlnom a szövegbe ahhoz, hogy a változtatásokat egyenként már ne akarjam kirészletezni ebben a listában.
- 289. oldal: Térkép az alkalmazás réteghez.
- És nem utolsósorban a második kötet megjelenése.

9.2 1.1 VERZIÓ, 2009 DECEMBER

- 12. oldal : Rétegek fogalmi zavarai javítva
- 23. oldal : Az FCS, ami ugye nem hash.
- 79. oldal : Az SSL - TLS átmenet tisztázása
- 86. oldal : PPPoE Discovery Phase pontosítása
- 86-87. oldal : A PPP, a PPPoE és az Ethernet struktúrák egymáshoz való viszonyának tisztázása, a 3.29 ábra módosítása
- Végül számtalan elgépelés javítása

10 A SZERZŐ

Itt szeretnék adni az érzésnek. Mármint az exhibicionizmus nevűnek.

Valamikor vegyészmérnöknek készültem, sőt, meglepő módon még diplomát is kaptam belőle. Pár évnek kellett csak eltelnie és társaságban már azt bizonygattam, hogy a kén-monoxid sokkal veszélyesebb, mint a kén-dioxid. Ha jót akarsz magadnak, nem engem kérdezel meg arról, mely anyagok mérgezőek.

Valójában soha nem is tekintettem magamat vegyésznek: az utolsó két évet a veszprémi egyetem kibernetika tanszékén töltöttem - és aki ismeri a helyet, tudja, hogy akkoriban arrafelé tanyáztak az égőszeműek, az elhivatott rendszeresek... azaz a Rendszermérnök szakos hallgatók. Gyakorlatilag nekünk csak a testnevelés és a nyelvi óráinkon nem volt matek - az összes többin tisztán. Modelleztünk (matek), optimalizáltunk (lineáris/nemlineáris algebra), mindezeket számítógépre vittük (numerikus matek), elemeztük (statisztika, valószínűségszámítás)... kikapcsolódásképpen ipari/általános számítástechnikát, ipari/kisgépes/nagygépes programozási nyelveket tanultunk. A hab a tortán a fizikai kémia volt, melyet gondolom az egyetem vegyipari jellege miatt csempésztek bele álcázásképpen a tanrendbe, de nálunk azt is statisztikus alapon oktatták.

Ehhez képest az első munkahelyemen, a veszprémi IKV távfűtési részlegén mindösszesen egy darab C128-as számítógép volt. Arra kellett ügyviteli szoftvereket. fejlesztenem. Mondjuk, még jó, hogy nem a főnök Casio menedzserkalkulátorára[94].

[94] A geek vonal egyébként már korán kiütközött. Az első programozható zsebszámológépem - Stylandia, egy szégyentelen tajvani Sharp koppintás - 48 lépést ismert. Erre gyártottam különböző programokat, úgy, hogy a gombnyomásokat cetlikre írtam fel. Különösen büszke voltam a másodfokú egyenlet megoldóképletére - hé, mondtam már, hogy 48 lépés? - és a Stirling formulával megvalósított faktoriális számolásra. Ez utóbbi ugyanis képes volt a 69-nél nagyobb számok esetében is faktoriálist számolni.

Aztán jött a rendszerváltozás, a tanácsi rendszer felbomlott, az IKV szintén - és ahogy a távfűtés önálló cég lett, megszabadultunk az összes kontraszelektált idiótától. Innentől kezdve tényleg informatikával foglalkoztam, rendszerépítés, ügyvitelszervezés, üzemeltetés, programozás (Clipper/Pascal), hálózat. Ahogy egy kis cégnél szokás az ilyesmi.

Az OOP már az üzemeltetési oldalon talált[95]. Nem szándékosan álltam át, egyszerűen a következő munkahelyemen erre a tudásra volt szükség. Ha fejlesztőt kerestek volna, akkor ma fejlesztenék.

A szakmai karrierem 2002-ben indult be igazán, amikor egy csoportos leépítés előjátékaként kirúgtak a munkahelyemről. Jó tíz hónapig kerestem új munkahelyet. Ekkor fogadtam meg, hogy ilyesmit többször nem engedélyezek a sorsnak. Rágyúrtam a szakmára, tanultam, mint az őrült, sorra nyomtam a - valódi - vizsgákat, a cégemnél is szépen haladtam előre, kaptam az egyre izgalmasabb feladatokat. 2005 körül vágtam bele egy szakmai/civil blogba, az itteni szakmai írások keltették fel később a Microsoft figyelmét. Így lettem a Technet Magazin egyik rendszeres szerzője. Innentől szépen apránként épültek egymásra a dolgok: cikkek, blogbejegyzések, oktatási anyagok... közben egy MVP cím... majd egy Exchange 2007 könyv, mely eredetileg Netacademia produkciónak indult, de a kézirat végül a Microsoft Magyarországnál landolt. Erre a mostani könyvre már a Microsofttól kaptam a megbízást - és ahogy most kinéznek a dolgok, valószínűleg nem ez lesz az utolsó.

Végül álljanak itt az elérhetőségek:

Szakmai blog:

EMAIL ÉS A DETEKTÍVEK	:	http://emaildetektiv.hu
MICROSOFT TECHNET PORTÁL	:	http://tinyurl.com/ydmbgdk

Privát blog

MI VAN VELEM?	:	http://mivanvelem.hu

Email:

jpetrenyi@gmail.com

[95] Ha csak nem számítjuk a kicsit bénácska objektumorientált Clippert, a CA-VO-t.