TCP/ IP Tutorial

Deze tutorial zal een overzicht geven van de TCP/ IP stack in het algemeen en IP nummering in het bijzonder. Er zal worden ingegaan op zaken als classful en classless IP adressering, subnetting, VLSM, DNS en routering.

Alle screenshots en datagrammen die zijn gesniffed zijn gebaseerd op communicatie tussen een computer met Windows 2000 Advanced Server en een computer met Windows 2000 Professional enerzijds, communicatie met computers op het Internet anderzijds. Dumps zijn gemaakt met de ingebouwde network sniffer van Advanced Server en met `windump.exe`. Grafische *ondersteuning* van uitgewerkte voorbeelden van subnetting zullen worden gegeven aan de hand van de IP Subnet Calculator van WildPackets, te downloaden van de volgende URL:

ftp://ftp.wildpackets.com/public/goodies/ipcalc321.exe

Dit programma zal echter alleen als ondersteuning worden gebruikt om de handberekende resultaten toe te lichten. Begrip gaat boven het intoetsen van wat nummers.

Veel leesplezier toegewenst.

Inhoudsopgave

1 Inleiding tot TCP/IP

1.1 Protocol

Als mensen met elkaar willen communiceren, dan kunnen ze dat alleen als ze dezelfde communicatieregels en taal gebruiken (al dan niet via een derde persoon die voor de vertaling zorgt). Hetzelfde geldt voor computers: computers begrijpen elkaar alleen als ze dezelfde communicatieregels gebruiken en dezelfde taal gebruiken (of als er hardware aanwezig is die de communicatieregels en taal die beide partijen hanteren succesvol kan vertalen). Neem als voorbeeld een telefoongesprek tussen twee mensen. Hierbij worden meestal de volgende afspraken gehanteerd:

- Degene die wordt gebeld zegt zijn naam;
- Degene die belt zegt zijn naam en begint zijn verhaal (wellicht in het Nederlands);
- Degene die gebeld wordt geeft antwoord, waarna de beller weer gaat praten;
- De personen praten niet door elkaar heen;
- De personen verbreken de verbinding als ze beiden klaar zijn met praten.

Alhoewel de vergelijking met menselijke communicatie natuurlijk vastloopt, kan een verzameling vergelijkbare stappen worden onderscheiden bij communicatie tussen computers. Er wordt een verbinding opgezet, data uitwisseling vindt plaats, er wordt alleen data verstuurd als de andere partij niet stuurt, en de verbinding wordt aan het einde van de sessie weer verbroken.

De taken die komen kijken bij het mogelijk maken van communicatie tussen computers, zoals het opzetten, coördineren en verbreken van de verbinding, het adresseren van de te versturen data etc., kunnen in aparte en onafhankelijke modulen worden ondergebracht. De functionaliteit van zo'n module wordt geïmplementeerd via een gestandaardiseerde verzameling regels en wordt met een duur woord een **protocol** genoemd. Nu zijn er, net zoals bij menselijke communicatie, verschillende protocollen. Zo nemen de mensen in andere landen de telefoon aan met "Ja?", of zeggen ze hun eigen telefoonnummer. Die mensen gebruiken dus een ander protocol.

1.2 Het OSI model

Om een goed beeld te krijgen van de communicatie protocollen die worden gebruikt door computers, is door de **International Organisation for Standardization** (ISO) een referentie model opgezet, waarbinnen protocollen een specifieke plaats krijgen al naar gelang de functie die het protocol uitvoert. Niet alle protocollen doen immers hetzelfde. De verzameling protocollen die is geïmplementeerd in een bepaald systeem wordt ook wel de **protocol stack** of **protocol suite** genoemd. De lagen waaruit het OSI model bestaat (en dus de protocollen in iedere laag) zijn onafhankelijk van zowel elkaar als het besturingssysteem waarop het model is geïmplementeerd. Iedere laag communiceert dus alleen met de laag direct boven of direct onder hem (uitzonderingen daargelaten). Ook maak het dus niet uit of de installatie op Linux zijn werk doet, of Windows XP. Door deze modulaire structuur wordt voorzien in verminderde complexiteit, een betere onderhoudbaarheid en standaardisatie waar iedereen gebruik van kan maken. De lagen die worden onderscheiden in het OSI model zijn de volgende:

- **Applicatie laag**: dit is het niveau van de applicatie waarmee de gebruiker werkt. Voorbeelden hiervan zijn FTP, Telnet, en E-mail;
- **Presentatie laag**: deze laag bepaalt hoe de data van de applicatie laag wordt gerepresenteerd. Voorbeelden zijn ASCII, JPG en MIDI;
- **Sessie laag**: deze laag zorgt voor het opzetten, managen en afsluiten van communicatie tussen applicaties, alsmede synchronisatie van communicatie. Een voorbeeldprotocol dat in deze laag zit is RPC;
- **Transport laag**: deze laag zorgt voor zaken zoals error correctie, het bijhouden welke pakketjes zijn verstuurd en ontvangen etc. Voorbeelden zijn TCP, UDP en SPX;
- **Netwerk laag**: deze laag zorgt voor adressering en de mogelijkheid tot routering, zodat de data op de plaats van bestemming aan kan komen. Voorbeeldprotocollen zijn IP, IPX en X.25;
- **Data link laag**: deze laag bestaat eigenlijk uit twee sublagen, te weten de Logical Link Control (**LLC**) en de Media Access Control (**MAC**). In de MAC laag wordt onder andere het hardware adres gespecificeerd van de netwerk interface. De data link laag zorgt voor het "low-level werk", zoals het omzetten frames in bits en vice versa, het uitvoeren van foutdetectie op de frames etc. Voorbeelden van protocollen in deze laag zijn HDLC, PPP en DSLC;
- **Fysieke laag**: zorgt voor het daadwerkelijk versturen van data en regelt zaken zoals voltage, de fysieke connectoren en data snelheden. Voorbeelden zijn standaarden zoals V.24 en EIA/TIA-449.

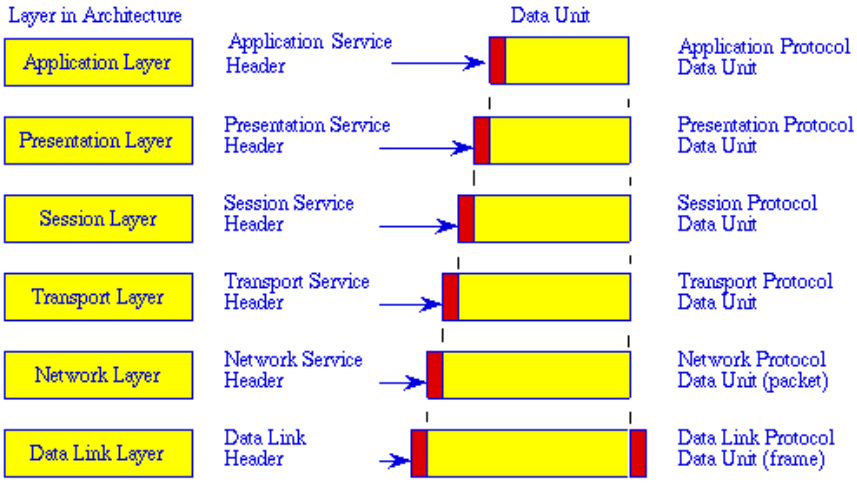
Aangezien weinig mensen de volgorde en de namen van deze lagen kunnen onthouden, is er een acroniem voor verzonnen, waarbij de eerste letter van het volgende woord overeenkomt met de bijbehorende laag (in het Engels, dus de laatste letter is niet correct) in het OSI model:

All **P**eople **S**eem **T**o **N**eed **D**ata **P**rocessing

Nu we weten waar het OSI model uit is opgebouwd, kunnen we gaan kijken naar de datagrammen zoals die over het netwerk vliegen. Stel je voor dat je een E-mail wilt versturen. De applicatie laag (het E-mail programma) stuurt de data van bovenaf door de protocol stack. Iedere laag voegt bepaalde controle informatie toe aan het ontvangen pakketje, en stuurt het nieuw gemaakte pakketje door naar de direct onderliggende laag. Deze laag ontvangt het pakketje van de bovenliggende laag, voegt er zijn eigen controle informatie aan toe, en stuurt het resultaat weer een laag naar beneden etc. Dit proces herhaalt zich net zolang totdat het pakketje uiteindelijk bij de fysieke laag uitkomt (er komen natuurlijk meer dingen bij kijken, maar dat is hier niet van belang). Als dit gebeurt, wordt het pakketje (in officiële termen spreekt men nu van een **frame**) vertaald in een rij enen en nullen, wordt het op het netwerk gezet en wordt verzonden.

Het proces dat hier is beschreven (toevoegen van "eigen" informatie in de vorm van een header en het doorsturen naar een volgende laag) wordt **encapsulation** genoemd. Schematisch ziet dit er als volgt uit[1]:

[1] Zoals wellicht duidelijk is zijn de headers niet in de correcte grootte afgebeeld. Headers op iedere laag verschillen van grootte, inhoud en structuur.



De rode verticale balk betreft de informatie aan die iedere laag toevoegt aan het van de boven gelegen laag ontvangen pakketje. De data link laag kan naast een **header** ook nog een **trailer** toevoegen (een zogenaamd **Frame Check Sequence**, een stuk informatie dat zorg draagt voor het detecteren van fouten als tijdens transmissie het frame beschadigt). Dit wordt echter als achterhaald beschouwd, waardoor de data link informatie beperkt blijft tot een header.

Als de datagram uiteindelijk aankomt bij het einddoel, dan worden precies de tegenovergestelde stappen doorlopen. Dit tegengestelde proces wordt **de-encapsulation** of **demultiplexing** genoemd. Het frame wordt van het netwerk gehaald en doorgegeven aan de data link layer. Die sloopt er de data link header vanaf, en stuurt het pakketje door naar het juiste netwerk protocol. Welk netwerk protocol dat is staat aangegeven in de data link header vermeld. De netwerk laag ontvangt het pakketje van de data link laag, haalt er zijn header vanaf, en stuurt het door naar het juiste transport layer protocol (welk protocol dat is staat wederom aangegeven; in dit geval in het protocol veld van de network layer header) etc. Deze stappen worden ondernomen voor alle frames die worden uitgewisseld tussen de broncomputer en de doelcomputer, net zolang totdat de verbinding tussen beide wordt onderbroken.

Enkele belangrijke dingen vloeien voort uit het OSI model:

- Protocollen die tot dezelfde laag behoren voeren hetzelfde type functie uit. Zo voeren het IP protocol en het IPX protocol dezelfde taken uit, ook al zien de headers die beide protocollen toevoegen aan het datagram er anders uit;
- Protocollen in verschillende lagen zijn onafhankelijk van elkaar, waardoor bijvoorbeeld IP over Ethernet mogelijk is, maar ook IP over Token Ring;
- Niet alle protocollen worden bij ieder te verzenden frame gebruikt. Zo wordt bij het versturen van een ping een datagram gemaakt die alleen de ICMP, IP en de data link header bevat. Een RPC datagram dat is gesniffed bevatte meer headers, te weten achtereenvolgens een MSRPC, SMB, NBT, TCP, IP en Ethernet header. De lagen zijn dus niet verplicht om een header toe te voegen als dat voor communicatie niet nodig is;
- Niet alle protocollen zijn makkelijk te plaatsen. Zo wordt het ICMP protocol (hetgeen onder andere wordt gebruikt voor het versturen van “ping” pakketten; -zie RFC 792-) beschouwd als een deel van het IP protocol, terwijl bij een gesniffed frame duidelijk zichtbaar is dat er een aparte ICMP header is toegevoegd “vóór” die van IP.

Een laatste punt dat nog kan worden gezegd over het OSI model is dat het aantal begrippen enorm is, maar dat vaak hetzelfde wordt bedoeld. Zo wordt de header die wordt toegevoegd door ieder protocol ook wel Protocol Control Information (**PCI**) genoemd. Het resulterende pakketje (ofwel, data van de bovengelegen lagen + PCI van de "eigen" laag) wordt een Protocol Data Unit (**PDU**) genoemd (zie ook bovenstaande figuur). Zo is er nog een scala van termen die allemaal terug zijn te voeren op bovenstaand verhaal. Laat je vooral niet in de war brengen door deze termen.

1.3 Frame adressering

We hebben nu in theorie gezien hoe een datagram wordt opgebouwd en verzonden. Dit is te vergelijken met het schrijven van een brief. We hebben nu de brief met postzegel, maar hoe zorgen we er nu voor dat de brief wordt bezorgd op het juiste adres? Makkelijk: we schrijven het adres, postcode en woonplaats van de ontvanger op de voorkant van de brief, schrijven eventueel ons eigen adres op de achterkant en doen hem op de post. Op een of andere manier zal dat dus waarschijnlijk ook gebeuren bij het versturen van frames. En inderdaad, in een van de headers definiëren we het adres van de computer waarmee we willen communiceren, alsmede ons eigen adres.

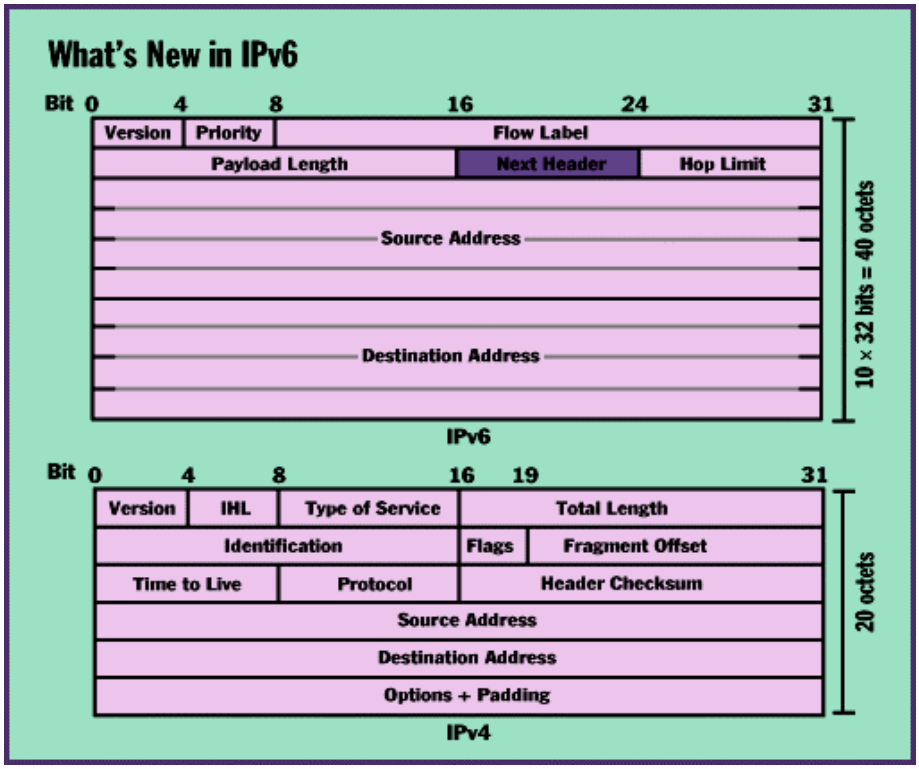
De manier waaarop bron- en doeladres worden weergegeven hangt af van het protocol dat in de netwerk laag wordt gebruikt. Zo is bij het IPX protocol het adres opgebouwd uit een **netwerk ID** (bestaande uit 8 hexadecimale getallen) en een **node ID** (bestaande uit 12 hexadecimale getallen). Het volgende schema geeft de IPX header weer in een Service Advertisement Protocol (**SAP**) frame[2]. De roodgedrukte adressen betreffen het bronadres en doeladres in de IPX notatie.

```
IPX: SAP Packet - 0.000039FAE138.4008 -> 0.FFFFFFFFFFFF.452 - 0 Hops
    IPX: Checksum = 65535 (0xFFFF)
    IPX: IDP Length = 96 (0x60)
    IPX: Transport control = 0 (0x0)
    IPX: Packet type = IPX
    IPX: Destination Address Summary 0.FFFFFFFFFFFF.452
        IPX: Destination IPX Address = 00000000.FFFFFFFFFFFF
        IPX: Destination Net Number = 0 (0x0)
        IPX: Destination Socket Number = SAP
    IPX: Source Address Summary 0.00 0039FAE138.4008
        IPX: Source IPX Address = 00000000.000039FAE138
        IPX: Source Net Number = 0 (0x0)
        IPX: Source Socket Number = 0x4008
    IPX: Data: Number of data bytes remaining = 66 (0x0042)
```

Het bronadres is gelijk aan `00000000.000039FAE138`, waarbij het deeel achter de punt de node ID is en automatisch op het MAC adres van de netwerkkaart wordt gezet. Een netwerk nummer van allemaal nullen geeft aan dat het bericht wordt verstuurd naar het netwerk waaraan de broncomputer is verbonden. Het doeladres is een broadcast (alle computers op het segment), en wordt aangegeven met `FFFFFFFFFFFF`.

[2] Het SAP protocol is een IPX-gerelateerd protocol, waarmee netwerk resources hun adres en hun diensten adverteren, en wordt standaard iedere 60 seconden verstuurd.

We zijn echter in deze tutorial echter helemaal niet geïntereseerd in IPX adressen. We richten ons hier op het IP protocol! Bij het IP protocol wordt een andere manier gehanteerd van adressering. Bij dit protocol wordt gebruik gemaakt van zogenaamde **IP adressen**, hetgeen ook unieke identificatienummers zijn, maar dan met een andere lengte en structuur. We zijn inmiddels aanbeland bij IP protocol versie 6 (IPv6, ook wel **IPng** –voor IP next generation- genoemd), hoewel deze versie alleen nog wordt gebruikt voor de zogenaamde **IPv6 backbone** (of kortweg **6bone**) De versie die momenteel het meest in gebruik is, is versie 4 (IPv4)[3]. De headers zoals die worden gemaakt door beide versies verschillen nogal van elkaar. Hieronder volgt een plaatje dat ik op mijn harde schijf heb gevonden. Het is ook ergens op het Internet te vinden.



Het *Source Address* veld bevat het IP adres van de verzender van het frame; het *Destination Address* veld bevat het IP adres van de ontvanger van het frame. Zoals je ziet is de lengte van een IP adres bij IPv6 vele malen groter dan bij IPv4 (te weten 128-bits versus 32-bits adressen). Dat moet ook wel, want door de enorme groei van het Internet dreigde de routeringtabellen onhandelbaar groot te worden, en dreigde tevens een tekort te ontstaan aan IP adressen in het algemeen en klasse B netwerken in het bijzonder (over klasse B netwerken later meer). Dat tekort is met de nieuwe nummering verdwenen. Hieronder volgt een voorbeeld van een IP header. De adressen zijn roodgedrukt[4].

[3] Zie voor een gedetailleerd overzicht van IPv4 en IPv4 de desbetreffende RFCs, respectievelijk RFC 791 en RFC 2460. Om zelf te zoeken naar RFCs, ga naar http://www.rfc-editor.org/rfcsearch.html. Voor meer informatie over IPv6, ga naar http://www.6bone.net/.

[4] Wellicht ten overvloede, maar de headers bevatten alleen maar enen en nullen, die allemaal een aparte betekenis hebben. De Network Monitor van Microsoft interpreteert deze enen en nullen en geeft het textuele equivalent ervan weer. Zo staat in het frame niet de string "Normal Service", maar een 0, hetgeen door alle systemen wordt geïnterpreteerd als "Normal Service".

```
IP: ID = 0x4ED7; Proto = UDP; Len: 328
    IP: Version = 4 (0x4)
    IP: Header Length = 20 (0x14)
    IP: Precedence = Routine
    IP: Type of Service = Normal Service
    IP: Total Length = 328 (0x148)
    IP: Identification = 20183 (0x4ED7)
    IP: Flags Summary = 0 (0x0)
        IP: .......0 = Last fragment in datagram
        IP: ......0. = May fragment datagram if necessary
    IP: Fragment Offset = 0 (0x0) bytes
    IP: Time to Live = 128 (0x80)
    IP: Protocol = UDP - User Datagram
    IP: Checksum = 0x2A25
    IP: Source Address = 192.168.0.1
    IP: Destination Address = 255.255.255.255
    IP: Data: Number of data bytes remaining = 308 (0x0134)
```

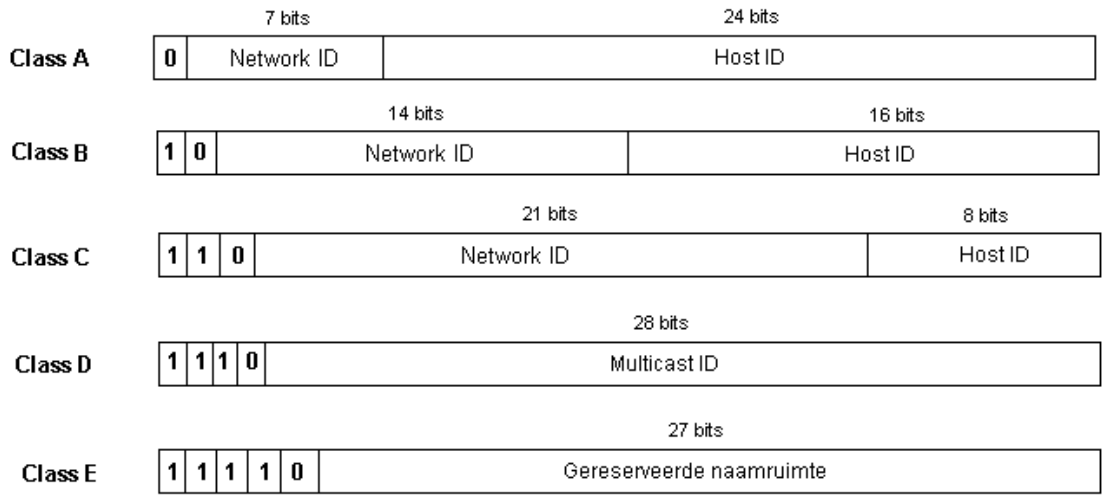
In de komende hoofdstukken zal aan de hand van de IPv4 specificatie een uitleg worden gegeven over structuur en opbouw van IP adressen, en hoe deze adressen ons helpen te communiceren met andere computers op het Internet.

2 TCP/IP adressering

Zoals we in hoofdstuk 1 hebben gezien, heeft iedere computer op een netwerk een uniek identificatienummer, waarmee het zich onderscheidt van de andere computers op het netwerk. Dit unieke identificatienummer is, als met het IP protocol wordt geocmmuniceerd, het IP adres van de computer, die naast een unieke identificatie ervoor zorg draagt dat de computer gevonden kan worden op het netwerk. Dit hoofdstuk zal een inleiding geven tot de classful en classless IP adressering, twee adresseringstypen die worden gebruikt bij het op een gestructureerde manier toewijzen van identificatienummers aan computers.

2.1 Classful IP adressering

In tegenstelling tot bijvoorbeeld het NetBIOS protocol, is de IP adressenstructuur hiërarchisch opgebouwd. Dit wil zeggen dat we niet een grote platte structuur hebben waarbij de uit te geven identificatienummers doorloopt van bijvoorbeeld 1 tot 10.000.000, maar dat er een bepaalde gelaagde structuur in te ontdekken is. Anders gezegd: IP adressen worden bij **classful IP adressering** ingedeeld in zogenaamde **klassen**. Onderstaand figuur geeft een overzicht van de klassen zoals die lang geleden zijn gedefinieerd.



De klassen A, B en C worden gebruikt voor het uitgeven van IP adressen aan devices die een interface hebben met het netwerk (er zijn enkele IP adressen die er buiten vallen; hierover later meer). Klasse D IP adressen worden gebruikt voor multicasting (zoals video conferencing), terwijl klasse E is gereserveerd voor toekomstig gebruik. Zoals kan worden afgelezen bestaat een klasse A, B of C IP adres uit twee delen, te weten een **Network ID** (het eerste gedeelte van het IP adres; dit deel wordt ook wel de **network prefix** genoemd) en een **Host ID** (het tweede gedeelte van het IP adres). Laten we eens wat verder ingaan op bovenstaand figuur.

Het network ID ligt bij classful IP adressering in omvang vast; de lengte van het network ID ligt vast op 7, 14 en 21 bits voor respectievelijk klasse A, klasse B en klasse C netwerken. Dit resulteert in een

structuur waarbij er veel meer C netwerken dan B netwerken zijn, en waarbij er veel meer B netwerken dan A netwerken zijn. Het volgende schema geeft een overzicht van aantallen netwerken, alsmede het maximale aantal computers op het netwerk.

Klasse	Network ID	Aantal netwerken	Host ID	Aantal hosts
A	7 bits	$2^7 - 2$	24 bits	$2^{24} - 2$
B	14 bits	$2^{14} - 2$	16 bits	$2^{16} - 2$
C	21 bits	$2^{21} - 2$	8 bits	$2^8 - 2$

Hoe komen we nu aan het aantal netwerken en het aantal hosts per netwerk? Laten we als voorbeeld klasse C nemen. Klasse C heeft een network ID van 21 bits, die allemaal in waarde kunnen variëren tussen 0 en 1. Aangezien iedere bitpositie 2 mogelijkheden heeft, volgt met de vermenigvuldigingsregel dat het totale aantal mogelijke C netwerken gelijk is aan 2 x 2 x ... x 2 (21 keer) = 2^{21} = 16.777.216. Van dat aantal moeten we er nog twee af halen, doordat een **all 1s network ID** en **all 0s network ID** niet is toegestaan. Voor de berekening van het aantal hosts kan min of meer dezelfde berekening worden gemaakt. Er zijn voor een klasse C netwerk in totaal 8 host ID bits die allemaal twee mogelijke waarden kunnen aannemen. Dit levert een totaal aantal hosts op van 2 x 2 x ... x 2 (8 keer) = 2^8 = 256. Ook hier moeten we er weer 2 van af halen, doordat het **all 1s host ID** en **all 0s host ID** niet is toegestaan (het all 0s host ID wordt gebruikt om het netwerk zelf aan te duiden; het all 1s host ID wordt gebruikt als broadcast adres op het netwerk segment). Op dezelfde manier kan het aantal netwerken en hosts voor klasse A en B worden berekend.

We hebben nu een hele rij met enen en nullen als uniek identificatienummer voor een computer. Dat is natuurlijk niet te onthouden. Om dat op te lossen noteren we een IP adres meestal als numerieke waarden in groepjes van 8 bits. Wat houdt dat precies in? Welnu, je hakt de string enen en nullen op in 4 groepen van 8 bits, en vertaalt dit binaire getal naar het decimale getallenstelsel.

Voorbeeld:
IP adres (binair): `10000010 11101010 00010011 00000101`
IP adres (decimaal): `130.234.19.5`

De decimale notatie van IP adressen wordt de **dotted decimal notation** genoemd en is voor mensen veel makkelijker te onthouden dan de binaire representatie. Hoe gaat deze conversie van binair naar decimaal in zijn werk? Het idee erachter is hetzelfde als ons decimale stelsel. In het decimale stelsel bepaalt de positie van een cijfer in een getal zijn waarde. Bijvoorbeeld:

$493 = 10^0 \times 3 + 10^1 \times 9 + 10^2 \times 4$

Zo geldt voor het binaire getal 11101010 hetzelfde, alleen werken we nu met grondtal 2 in plaats van grondtal 10:

$11101010 = 2^0 \times 0 + 2^1 \times 1 + 2^2 \times 0 + 2^3 \times 1 + 2^4 \times 0 + 2^5 \times 1 + 2^6 \times 1 + 2^7 \times 1 = 234$

Om te converteren van decimaal naar binair kan natuurlijk ook gebruik worden gemaakt van `calc.exe`.

Je kunt je nu afvragen of aan het IP adres kan worden afgelezen in welke klasse een IP adres thuishoort. Welnu, dat kan! Laten we als voorbeeld eens klasse B bekijken. Hierboven is gemeld dat klasse B netwerken een 14-bits network ID hebben. De waarde van de eerste byte (de eerste acht bits) kan dus de volgende waarden aannemen (de rode bits mogen niet veranderen):

```
10000000    =>    128
10000001    =>    129
10000010    =>    130
10000011    =>    131
...
10111100    =>    188
10111101    =>    189
10111110    =>    190
10111111    =>    191
```

Hierbij moet opgemerkt worden dat bij de eerste entry (10000000) en de laatste entry (10111111) de tweede byte niet gelijk mag zijn aan 00000000 respectievelijk 11111111, anders hebben we een all 0s en all 1s netwerk, en dat mag niet. Zo te zien kan een IP adres uit een klasse B netwerk dus herkend worden aan het eerste byte, die in de range 128 – 191 (inclusief) ligt. Dezelfde berekening kan worden uitgevoerd voor alle klassen, hetgeen in de volgende tabel is samengevat:

Klasse	**Network ID**	**Geldige range 1ste byte**
A	*a*.0.0.0	1 – 126
B	*a.b*.0.0	128 – 191
C	*a.b.c*.0	192 – 223
D	Niet van toepassing	224 – 239
E	Niet van toepassing	240 – 255

De reden waarom klasse A niet het 127.*x.x.x* IP adres bevat komt doordat dit network ID nummer is gereserveerd voor het testen van de lokale TCP/IP implementatie. Het 127.*x.x.x* adres wordt het **local loopback** adres genoemd. Alhoewel de tweede, derde en viede byte mogen variëren, gebruiken de meeste implementaties IP adres 127.0.0.1 voor de loopback functie.

Als voorproefje op wat komen gaat wordt hier alvast het begrip **subnet mask** uitgelegd (ook wel **address mask** genoemd). Zoals we hebben gezien is een IP adres onderverdeeld in een network ID en een host ID. De grens tussen deze twee ligt bij classful IP adressering vast. Bij classless IP adressering kan deze echter verschillen per host, zelfs als deze hosts in dezelfde klasse zijn ingedeeld. Om de grens aan te geven tussen network ID en host ID wordt een string van 32 bits gebruikt die bestaat uit een serie aansluitende enen, gevolgd door een serie aansluitende nullen. Het aantal enen (en daaruitvolgend het aantal nullen) bepaalt de grens tussen network ID en host ID. Er is een 1-op-1 mapping tussen het aantal enen in de subnet mask en het aantal bits in het IP adres dat dienst doet als network ID. Een voorbeeld zal dit duidelijker maken.

Voorbeeld.

We hebben het classful IP adres van 213.4.6.189. Zoals kan worden afgelezen in bovenstaande tabel hebben we te maken met een klasse C netwerk. Dit netwerk heeft een network ID van 24 bits. We geven dit met het volgende subnet mask weer:

```
11111111  11111111  11111111  00000000
```

Wat houdt die string in? Dit houdt in dat het network ID van deze host de eerste 24 bits van het IP adres beslaat, hetgeen overeenkomt met een network ID van 213.4.6.0 (let op de notatie (0 op het eind)). Voor de notatie van het network ID geldt:

> *Alleen de bits in het IP adres die behoren tot het network ID worden weergegeven bij de dotted decimal representatie van de network ID.*

In dit geval is dat dus 213.4.6.0, maar als we een IP adres hebben van 134.34.6.89 /16, dan is het network ID van deze host gelijk aan 134.34.0.0. Dit kan een beetje verwarrend zijn. Immers, had ik hierboven niet gezegd dat het network ID van een klasse C netwerk slechts 21 bits besloeg? Dat is inderdaad juist, maar bij de notatie van het subnet mask worden ook de in het IP adres vaste bits meegenomen, waardoor we toch spreken over een /24 netwerk als we praten over een classful C netwerk.

Samengevat: te zien aan het subnet mask heeft het IP adres 24 bits die dienst doen als network ID, en dat we *in het geval van classful IP adressing* te maken hebben met een klasse C netwerk. (Waarom dit cursief is gedrukt wordt duidelijk in de volgende paragraaf.) Het aantal enen in de subnet mask wordt vaak achter het IP adres gezet, zodat we meteen kunnen zien wat de combinatie {IP adres; subnet mask} is. Dit wordt als volgt gedaan:

<IP adres> /<aantal enen in subnet mask>

Bijvoorbeeld:

- 23.32.65.234 /8
- 167.216.250.12 /16
- 203.23.45.189 /24

Deze specifieke notatie wordt ook wel de **network prefix notation** genoemd. Wat opvalt bij de subnet mask is dat de enen zich aan de linkerkant van de string bevinden (in duur taalgebruik: de **high-order bits** zijn 1) terwijl de bits aan de rechterkant gelijk zijn aan nul (de **low-order bits** zijn gelijk aan 0). Een ander punt dat naar voren komt is het feit dat de enen een aaneengesloten rij vormen. De enen en nullen worden dus niet afwisselend in de subnet mask gezet.

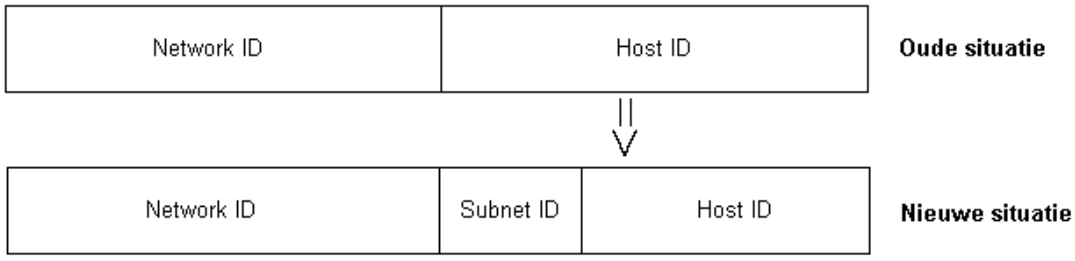
Nu zul je denken: je hebt me net verteld om aan de hand van het eerste byte de klasse van een IP adres te achterhalen, dus het subnet mask is niet echt nodig om te zien wat het network ID behorende bij een IP adres is. Daar heb je gelijk in; bij classful IP adressen hebben we dat strikt gesproken niet nodig. De noodzaak komt pas naar voren als we gaan subnetten. Dat is het onderwerp van de volgende paragraaf.

2.2 Subnetting

We hebben nu gezien hoe een IP adres is opgebouwd, te weten als combinatie van een host ID en een network ID. Als we nu goed kijken naar de omvang van een klasse B netwerk, dan is het aantal mogelijke hosts op zo'n netwerk tamelijk groot (65.534), om over een klasse A netwerk nog maar te zwijgen! Stel je nu eens voor dat een onderneming die zich specialiseert in grafische applicaties (streaming media research etc.) een klasse B adres toegewezen heeft gekregen. Het verkeer dat over dat interne netwerk zal gaan is door al die multimedia activiteiten enorm. Als alle hosts aan het communiceren slaan, dan zal in no-time het hele netwerk verzadigd raken (in het Engels: **saturation**). Het zou dus handig zijn om dit enorme netwerk op te splitsen in een verzameling kleinere deelnetwerkjes, die op hun beurt aan elkaar worden geknoopt door middel van speciale netwerk devices die verkeer tussen deze deelnetwerkjes kunnen reguleren. Als dit slim wordt aangepakt, dan zal het verkeer voornamelijk beperkt blijven tussen computers die op hetzelfde deelnetwerk zitten, waardoor de systeembeheerder bandbreedte-intensieve computers die veel met elkaar communiceren weet te neutraliseren (het verkeer van die computers hoeft immers niet voorbij die netwerk device, die dat deelnetwerkje scheidt van de andere deelnetwerkjes). Is dat in praktijk uit te werken? Jazeker! De techniek die hiervoor wordt gebruikt heet subnetting.
Subnetting is het opdelen van een network ID in meerdere kleinere network IDs. Aan de hand van een voorbeeld zal worden uitgelegd hoe dit in zijn werk gaat[5].

2.2.1 Subnet IDs en host IDs

Stel je voor dat we te maken hebben met een klasse B netwerk. Dit netwerk heeft een network ID van 14 bits en een host ID van 16 bits. Wat we doen is het volgende: we delen het host ID op in twee delen, te weten het **subnet ID** en een nieuwe host ID. Grafisch ziet dit er als volgt uit:



Het network ID en het subnet ID vormen samen het **extended network prefix**, en vormen een "nieuw" network ID voor een host. De subnet mask van een host die zich in dit gesubnette netwerk bevindt is nu gelijk aan het aantal bits dat het extended network prefix beslaat. Laten we dit eens uitwerken.

Voorbeeld.

Laten we uit gaan van het klasse B netwerk 180.34.0.0 /16. In binaire vorm wordt dit gerepresenteerd als

```
10110100 00100010 00000000 00000000
```

[5] Voor meer informatie over subnetting, bekijk RFC 950 en RFC 1878.

De blauwe bits vormen het klasse B network ID (even de twee eerste bits buiten beschouwing latend), terwijl de zwarte bits de host ID vormen. Wat we nu gaan doen is het volgende: we splitsen de host ID in twee delen, waarvan de subnet ID 5 bits beslaan en de nieuwe host ID 11 bits (16 – 5):

```
10110100 00100010 00000000 00000000
```

De okergele bits vormen het subnet ID, terwijl de rode bits het nieuwe host ID vormen. De blauwe plus de okergele bits vormen samen de extended network prefix. De subnet mask is nu veranderd van /16 naar /21 (16 oorspronkelijke bits + 5 nieuwe subnet bits). Met deze vijf extra bits gaan we nieuwe kleinere netwerken maken. Hoeveel? Aangezien ieder bit wederom 2 mogelijke waarden kan aannemen, hebben we in totaal 2 x 2 x 2 x 2 x 2 = 2^5 = 32 kleinere netwerken, of 32 **subnets**.

Het aantal bits dat het subnet ID vormt bepaalt hoeveel subnets gemaakt kunnen worden.

Dit nieuwe netwerk, dat bestaat uit meerdere kleinere segmenten die met elkaar worden verbonden door routers, wordt een **internetwork** genoemd. Hieronder staan de subnets uitgewerkt voor ons 180.34.0.0 netwerk met 21 extended network prefix bits. De extended network prefix is onderstreept, het subnet ID is vetgedrukt.

Basis netwerk: 180.34.0.0 /16 (Let op: **16**!!)

```
Subnet #0       10110100 00100010 00000000 00000000      180.34.0.0 /21
Subnet #1       10110100 00100010 00001000 00000000      180.34.8.0 /21
Subnet #2       10110100 00100010 00010000 00000000      180.34.16.0 /21
Subnet #3       10110100 00100010 00011000 00000000      180.34.24.0 /21
....
Subnet #29      10110100 00100010 11101000 00000000      180.34.232.0 /21
Subnet #30      10110100 00100010 11110000 00000000      180.34.240.0 /21
Subnet #31      10110100 00100010 11111000 00000000      180.34.248.0 /21
```

Wat is het gevolg van deze opdeling in subnets? Laten we de drie belangrijkste gevolgen eens opsommen:

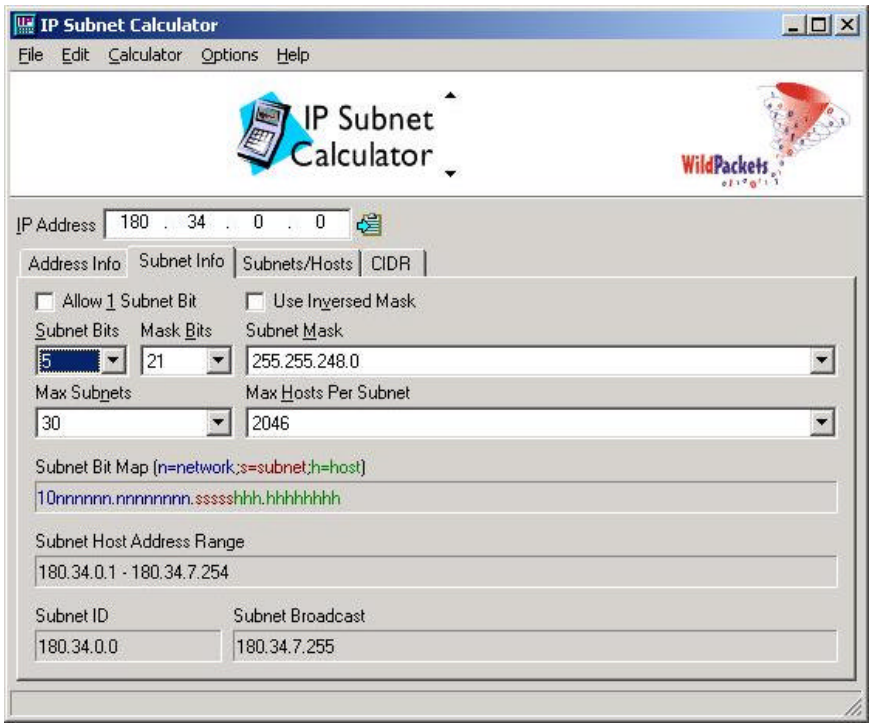
1. We hebben nu 32 kleinere netwerk segmenten ($2^{\text{aantal subnet ID bits}}$) in plaats van 1 groot segment. Dit komt, mits goed gepland, de schaalbaarheid en netwerk efficiency ten goede. Het aantal te gebruiken subnets ligt overigens op 30; subnet #0 en subnet #31 vallen beide af. Deze twee subnets worden ook wel het **all 0s subnet ID** en het **all 1s subnet ID** genoemd. Ik kom hier dadelijk nog op terug;
2. Het maximaal aantal subnets dat kan worden gemaakt is altijd een veelvoud van twee. Dit komt, doordat het getal 2 als grondtal wordt gebruikt bij berekening van het aantal subnets;
3. Het aantal mogelijke hosts op ons netwerk is afgenomen. Laten we eens gaan tellen. We hadden eerst een netwerk met $2^{16} – 2$ = 65.534 mogelijke hosts. Nu hebben we 30 bruikbare kleinere netwerksegmenten (subnet #0 en subnet #31 vallen af), die ieder ($2^{\text{aantal host ID bits}} – 2$) = $2^{11} – 2$ = 2046 hosts kunnen bevatten, hetgeen in totaal 30 x 2046 = 61.380 hosts zijn. Ofwel: we hebben 65.534 – 61.380 = 3.154 hosts minder tot onze beschikking. Dit resultaat is een standaard regel bij subnetting:

   Hoe meer subnets worden gedefinieerd, hoe minder host adressen er in totaal beschikbaar zijn.

Het totaal aantal subnets dat is berekend dient steeds met twee te worden verminderd. Bijvoorbeeld, bij een subnet ID van 4 bits is het totaal aantal subnets gelijk aan $2^4 - 2 = 14$, terwijl bij een subnet ID van 6 bits het totaal aantal subnets gelijk is aan $2^6 - 2 = 62$. Het all 0s subnet ID en het all 1s subnet ID worden niet meegenomen in het IP adresserings schema. Waarom is dat zo? Deze definitie stamt af van de subnetspecificatie in RFC 950, die deze subnets verbood. Waarom? Omdat routers in verwarring konden raken! Hoe komt dat? Laten we eens naar bovenstaand voorbeeld kijken.

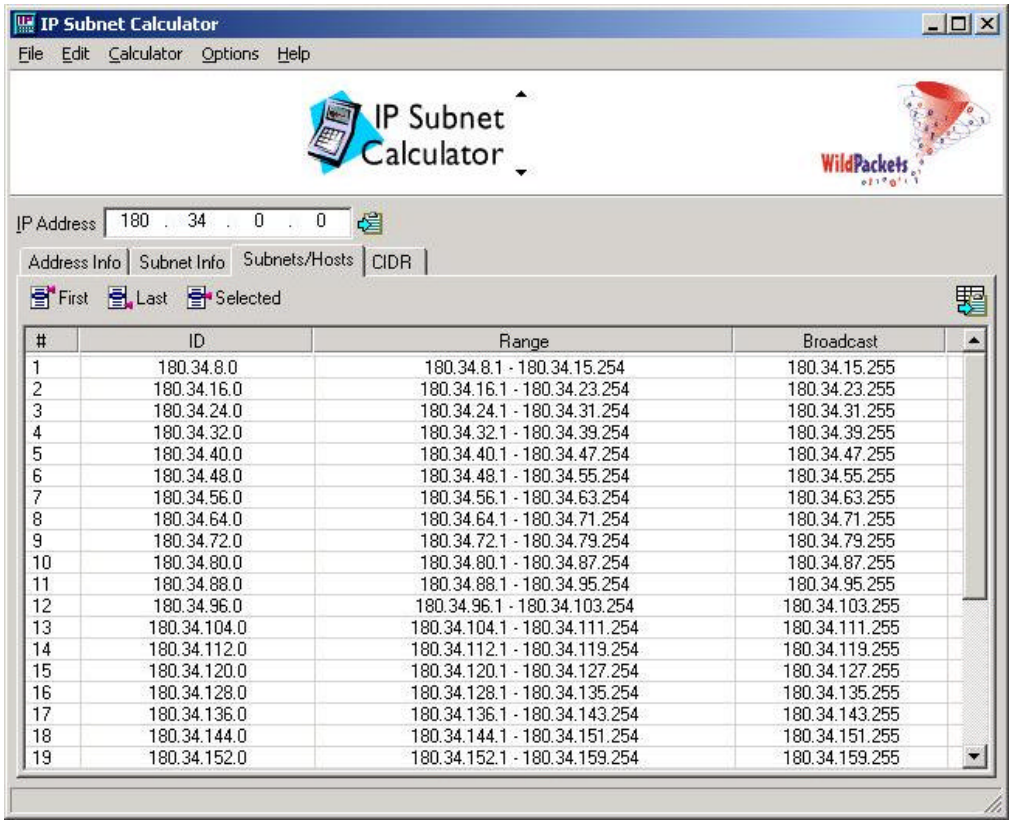
Het basis netwerk heeft een IP adres van **180.34.0.0 /16**, terwijl het all 0s subnet ID een IP adres heeft van **180.34.0.0 /21**. Als ik nu de subnet mask niet definieer, dan heb ik twee dezelfde IP adressen, waarbij niet herleid kan worden of ik de een of de ander bedoel! Bedoel ik nu subnet #0, of bedoel ik nu het basis netwerk van waaruit de subnets zijn gedefinieerd? Hetzelfde geldt voor een broadcast bericht. Ik stuur twee broadcast berichten; eentje naar netwerk 180.34.0.0 /16 (het basis netwerk) en eentje naar netwerk 180.34.248.0 /21 (subnet # 31). Als ik in beide gevallen alle host ID bits op 1 zet (de definitie van broadcast), dan komt in beide gevallen hetzelfde IP adres tevoorschijn: **180.34.255.255**. Ofwel: zonder subnet mask is er geen onderscheid te maken. De vraag is of deze stellingen met de huidige software nog wel geldig zijn. Zo geeft Cisco de mogelijkheid om het all 0s subnet ID gewoon te gebruiken. Het zal van de betreffende software afhangen wat wordt ondersteund.

Om de IP adressen van de netwerken allemaal handmatig uit te moeten rekenen is ook geen pretje. De (gratis) software om het allemaal automatisch te doen is WildPackets' IP Subnet Calculator. Hierin ziet de voorgaande subnet berekening er als volgt uit:



In de "IP Address" text area geef je het IP adres van het netwerk op (in ons geval 180.34.0.0), terwijl op de "Subnet Info" tab alle parameters worden opgegeven die van belang zijn. Bij ons is dat de subnet mask, die in twee vormen opgegeven kan worden: via de dotted-decimal notatie ("Subnet Mask"), of via

het aantal subnet bits ("Mask Bits"). Alle andere informatie, zoals het aantal subnets en het aantal hosts per subnet kunnen meteen worden afgelezen. De onderste drie getallen betreffen subnet #0, en bieden weinig informatie. De blauwe `10` in 10nnnnnn.nnnnnnnn.sssssshhh.hhhhhhhh (in het Subnet Bit Map deel van het plaatje) komt overeen met de bits die vastliggen voor een klasse B netwerk (de eerste byte is 180, hetgeen een klasse B netwerk inhoudt, hetgeen weer betekent dat de eerste twee bits gelijk zijn aan `10`). Het aantal bits van de extended network prefix (aangegeven in het blauw en rood) is gelijk aan 21, waardoor er nog 11 host ID bits overblijven (aangegeven in het groen). Voor een overzicht van alle subnetten en de range van mogelijke host IDs in ieder subnet moet het Subnets/Hosts tabblad worden aangeklikt:



#	ID	Range	Broadcast
1	180.34.8.0	180.34.8.1 - 180.34.15.254	180.34.15.255
2	180.34.16.0	180.34.16.1 - 180.34.23.254	180.34.23.255
3	180.34.24.0	180.34.24.1 - 180.34.31.254	180.34.31.255
4	180.34.32.0	180.34.32.1 - 180.34.39.254	180.34.39.255
5	180.34.40.0	180.34.40.1 - 180.34.47.254	180.34.47.255
6	180.34.48.0	180.34.48.1 - 180.34.55.254	180.34.55.255
7	180.34.56.0	180.34.56.1 - 180.34.63.254	180.34.63.255
8	180.34.64.0	180.34.64.1 - 180.34.71.254	180.34.71.255
9	180.34.72.0	180.34.72.1 - 180.34.79.254	180.34.79.255
10	180.34.80.0	180.34.80.1 - 180.34.87.254	180.34.87.255
11	180.34.88.0	180.34.88.1 - 180.34.95.254	180.34.95.255
12	180.34.96.0	180.34.96.1 - 180.34.103.254	180.34.103.255
13	180.34.104.0	180.34.104.1 - 180.34.111.254	180.34.111.255
14	180.34.112.0	180.34.112.1 - 180.34.119.254	180.34.119.255
15	180.34.120.0	180.34.120.1 - 180.34.127.254	180.34.127.255
16	180.34.128.0	180.34.128.1 - 180.34.135.254	180.34.135.255
17	180.34.136.0	180.34.136.1 - 180.34.143.254	180.34.143.255
18	180.34.144.0	180.34.144.1 - 180.34.151.254	180.34.151.255
19	180.34.152.0	180.34.152.1 - 180.34.159.254	180.34.159.255

Het eerste subnet wordt door de IP Subnet Calculator buiten beschouwing gelaten, alsmede het laatste subnet (niet zichtbaar op deze screenshot).

De definitie van subnet mask moet nu iets aangepast worden. We zien namelijk dat de bits die dienst doen als subnet ID ook tot het network ID van een IP adres behoren. Kortom, voor een subnet mask geldt:

- Alle bits die behoren tot het network ID hebben de waarde 1;
- Alle bits die behoren tot het subnet ID hebben de waarde 1;
- Alle bits die behoren tot het host ID hebben de waarde 0.

Het aantal bits dat wordt "afgesnoept" van het oorspronkelijke host ID moet goed worden gepland. Immers, dit aantal bepaalt hoeveel subnets we kunnen maken, en hoeveel hosts er op ieder subnet mogelijk zijn. Hierbij is de huidige situatie van belang, maar nog meer de toekomstige situatie, alsmede

de onderhoudbaarheid. Zo zal in bovenstaand voorbeeld een extended subnet mask van /27 niet echt goed te onderhouden zijn, doordat we een totaal aantal subnetten zouden hebben van $2^{11} = 2048$ met slechts een klein aantal hosts op ieder subnet. Da's niet echt overzichtelijk, afgezien van de kosten voor installatie van routers om al die netwerk segmenten aan elkaar te koppelen. Meestal wordt bij een planning van het IP adressering schema gekeken hoe het netwerk er over 5 jaar (of nog langer!) uitziet. Misschien zijn 10 segmenten nu voldoende, maar groeit de organisatie uit tot het viervoudige. Problemen met verkeerd geplande IP adresseringsschema's zijn enorm arbeidsintensief om ongedaan te maken, dus een goed IP adresseringsplan is van groot belang.

2.2.2 Directe en indirecte communicatie

In het laatste voorbeeld van paragraaf 2.1 is makkelijk te zien welke bits behoren tot het network ID, en hoe dit network ID wordt gerepresenteerd, doordat het aantal enen in de subnet mask op een byte grens ligt (dit is inherent aan de classful adressering die van toepassing is). Wat lastiger wordt het als de grens tussen network ID en host ID niet op een byte grens ligt. Hoe weten we dan wat het network ID van een host is? We hebben nu kennis nodig van logische operatoren, te weten de **logical AND**. Wat houdt de logical AND in? Kijk naar het volgende tabelletje.

A	**B**	**A AND B**
1	1	1
1	0	0
0	1	0
0	0	0

Wat houdt dit nu precies in? Het tabelletje zegt het volgende: als we een logical AND operatie uitvoeren met als operanden A en B, en één van de twee operanden is, of beide zijn gelijk aan 0, dan is het resultaat van de (bitwise) logical AND berekening ook 0. Laten we dit eens met een voorbeeld toelichten.

Voorbeeld.

We hebben het volgende IP adres en subnet mask:
IP adres: 130.234.19.5
Subnet mask: 255.248.0.0

Het subnet mask is in binaire notatie gelijk aan `11111111 11111000 00000000 00000000`, dus bovenstaande informatie kan ook worden weergegeven met 130.234.19.5 /13. Zowel het IP adres als de subnet mask worden omgezet naar de binaire notatie, waarna de bitwise logical AND wordt berekend van de bits in het IP adres en de bijbehorende bits in de subnet mask. Het betreft hier steeds een bitwise logical AND van de bits die op dezelfde positie staan (bijvoorbeeld, allebei op positie 1, of allebei op positie 17):

```
IP adres:        10000010  11101010  00010011  00000101
Subnet mask:     11111111  11111000  00000000  00000000
Logical AND      10000010  11101000  00000000  00000000
```

De uitkomst van deze berekening is het network ID van de computer. In dit geval is dat (in dotted decimal notation) 130.232.0.0. Nu vraag jij je af: en wat kan ik hier in hemelsnaam mee?? Het antwoord is belangrijk:

> *Als de network IDs van twee computers gelijk zijn, dan bevinden de twee computers zich op hetzelfde netwerk segment. Zo niet, dan bevinden de twee hosts zich op verschillende segmenten.*

Kortom: als we een IP adres hebben, alsmede de bijbehorende subnet mask, dan kunnen we het netwerk segment van de computer bepalen, en als gevolg daarvan bepalen of twee computers direct met elkaar kunnen communiceren (anders gezegd: een **direct route** hebben), of dat ze alleen kunnen communiceren met tussenkomst van een router (een **indirect route**). Nog een voorbeeld.

Voorbeeld
Bevinden de volgende hosts zich op hetzelfde netwerk segment?

- IP adres 1: 231.198.23.133 /27
- IP adres 2: 231.198.23.192 /27

We moeten de logical AND uitrekenen van de binaire representatie van zowel het IP adres als de subnet mask. Hieronder staan ze uitgewerkt:

IP adres 1:	11100111	11000110	00010111	10000101
Subnet mask:	11111111	11111111	11111111	11100101
Logical AND:	11100111	11000110	00010111	10000000
IP adres 2:	11100111	11000110	00010111	11000000
Subnet mask:	11111111	11111111	11111111	11100000
Logical AND:	11100111	11000110	00010111	11000000

Het network ID van host 1 is gelijk aan 231.198.23.128 (converteer de uitkomst van de bitwise logical AND naar het decimale stelsel). Het network ID van host 2 is gelijk aan 231.198.23.192. Kortom: de hosts bevinden zich niet op hetzelfde netwerk segment. Een belangrijk punt dat hier naar voren komt is het volgende:

> *De host ID van een computer op een netwerk segment moet uniek zijn voor dat netwerk segment.*

Als dat niet het geval zou zijn, dan zouden er IP conflicten optreden, doordat twee computers hetzelfde IP adres hebben, hetgeen de communicatie van een computer met andere computers compleet ontregelt.

We kunnen nu dus, aan de hand van het eigen subnet en IP adres, alsmede het subnet en IP adres van de doelcomputer, bepalen of twee computers zich op hetzelfde netwerk bevinden. Is dit zo verschrikkelijk

interessant om te weten? Voor de gebruiker wellicht niet, maar voor communicatie tussen computers is dit enorm belangrijk. We komen hier in paragraaf 3.1 nog op terug. In die paragraaf zal een basale uitleg worden gegeven van het routeringsproces, en het hierbij belangrijke ARP protocol.

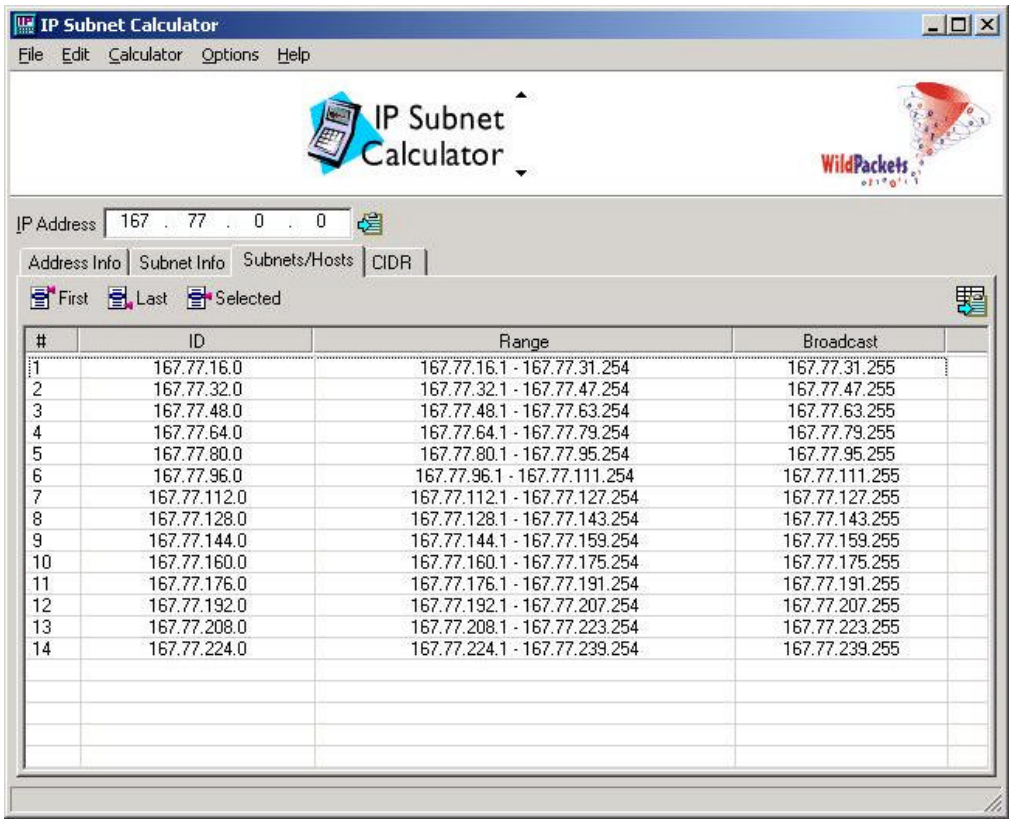
2.2.3 Variable Length Subnet Mask (VLSM)

We hebben gezien dat bij subnetting de IP ruimte wordt verdeeld in een *x* aantal gelijke parten, die we subnets noemen. Een organisatie kan besluiten dat iedere afdeling zijn eigen subnet krijgt toegewezen. Nu kan die afdeling beslissen dat het subnet dat ze krijgen opgedeeld moet worden in nog kleinere netwerkjes. Is dat mogelijk? Als de routing protocollen die zijn geimplementeerd op de routers het toelaten wel! Dit opnieuw opdelen gaat als volgt in zijn werk. Stel je voor dat we de volgende IP range tot onze beschikking hebben: 167.77.0.0 /16. We gaan dit netwerk opdelen in 16 subnets. Voor 16 subnets hebben we 4 subnet bits nodig (immers, $2^4 = 16$), zodat we de volgende situatie krijgen.

Basis netwerk: 167.77.0.0 /16

```
Subnet #0   10100111  01001101  0000 0000  00000000    167.77.0.0 /20
Subnet #1   10100111  01001101  0001 0000  00000000    167.77.16.0 /20
Subnet #2   10100111  01001101  0010 0000  00000000    167.77.32.0 /20
...
Subnet #13  10100111  01001101  1101 0000  00000000    167.77.208.0 /20
Subnet #14  10100111  01001101  1110 0000  00000000    167.77.224.0 /20
Subnet #15  10100111  01001101  1111 0000  00000000    167.77.240.0 /20
```

WildPackets' output ziet er als volgt uit:



#	ID	Range	Broadcast
1	167.77.16.0	167.77.16.1 - 167.77.31.254	167.77.31.255
2	167.77.32.0	167.77.32.1 - 167.77.47.254	167.77.47.255
3	167.77.48.0	167.77.48.1 - 167.77.63.254	167.77.63.255
4	167.77.64.0	167.77.64.1 - 167.77.79.254	167.77.79.255
5	167.77.80.0	167.77.80.1 - 167.77.95.254	167.77.95.255
6	167.77.96.0	167.77.96.1 - 167.77.111.254	167.77.111.255
7	167.77.112.0	167.77.112.1 - 167.77.127.254	167.77.127.255
8	167.77.128.0	167.77.128.1 - 167.77.143.254	167.77.143.255
9	167.77.144.0	167.77.144.1 - 167.77.159.254	167.77.159.255
10	167.77.160.0	167.77.160.1 - 167.77.175.254	167.77.175.255
11	167.77.176.0	167.77.176.1 - 167.77.191.254	167.77.191.255
12	167.77.192.0	167.77.192.1 - 167.77.207.254	167.77.207.255
13	167.77.208.0	167.77.208.1 - 167.77.223.254	167.77.223.255
14	167.77.224.0	167.77.224.1 - 167.77.239.254	167.77.239.255

Als de juridische afdeling subnet # 13 toegewezen krijgt, dan hebben ze netwerk segment 167.77.208.0 /20 tot hun beschikking. Wat we echter hebben gedaan in de vorige stap, kunnen we nu gewoon herhalen: we delen de host ID (de laatste 12 bits ; 32 bits minus de 20 subnet bits) gewoon nog een keer op in een subnet ID en een nieuw host ID! Stel je voor dat de juridische afdeling heeft besloten om hun segment op te delen in 5 subnetten. Hiervoor zijn 3 subnet bits nodig, hetgeen 8 subnets oplevert (zoals gezegd kunnen subnets alleen aangemaakt worden in veelvouden van 2). De situatie die zich nu voordoet is hieronder uitgewerkt. Het nieuwe subnet ID is vetgedrukt; het nieuwe network ID is onderstreept.

Basis netwerk: 167.77.208.0 /20 (Let op: nu **/20**; de laatste vier bits van het network ID zijn de subnet bits van de eerste subnetting procedure):

```
Subnet #0   10100111  01001101  1101000 0  00000000    167.77.208.0 /23
Subnet #1   10100111  01001101  1101001 0  00000000    167.77.210.0 /23
Subnet #2   10100111  01001101  1101010 0  00000000    167.77.212.0 /23
Subnet #3   10100111  01001101  1101011 0  00000000    167.77.214.0 /23
Subnet #4   10100111  01001101  1101100 0  00000000    167.77.216.0 /23
Subnet #5   10100111  01001101  1101101 0  00000000    167.77.218.0 /23
Subnet #6   10100111  01001101  1101110 0  00000000    167.77.220.0 /23
Subnet #7   10100111  01001101  1101111 0  00000000    167.77.222.0 /23
```

Helaas kan de IP Subnet Calculator dit niet weergeven, doordat de classful subnet mask in de som

Classful subnet mask + subnet ID bits = subnet mask

bij de Calculator vaststaat voor een IP adres (correct voor classful adressering). Echter, in bovenstaand voorbeeld is dit getal gelijk aan 20 en niet 16 (de classful subnet mask van een adres dat begint met 167). Voor dit voorbeeld dus geen plaatje.

Hier geldt hetzelfde verhaal als bij ieder andere subnetting procedure: het all 0s subnet (subnet #0) en het all 1s subnet (subnet #7) komen te vervallen, waardoor het totale aantal subnets dat beschikbaar is voor de juridische faculteit gelijk is aan 6.

Laten we nu eens teruggaan naar bladzijde 12, waar ik het volgende had neergezet:

[...]
Samengevat: te zien aan het subnet mask heeft het IP adres 24 bits die dienst doen als network ID, en dat we *in het geval van classful IP adressing* te maken hebben met een klasse C netwerk.
[...]

Bij classful adressering konden we aan de hand van het subnet mask de klasse van het netwerk afleiden. Een subnet mask van 8 stond gelijk aan een klasse A netwerk, een subnet mask van 16 stond gelijk aan een klasse B netwerk, en een subnet mask van 24 stond gelijk aan een klasse C netwerk. Dat verhaal gaat nu niet meer op! Het kan nu goed zijn dat een /24 netwerk betrekking heeft op een gesubnet klasse B netwerk (een B netwerk met 8 subnet bits).

Wellicht is nu ook duidelijk waarom men spreekt over Variable Length Subnet Mask. Je ziet dat het netwerk opgedeeld kan worden in niet even grote deelsegmenten, maar dat verschillende segmenten met verschillende groottes mogelijk zijn, juist doordat we kunnen kiezen voor subnet masks die niet allemaal aan elkaar gelijk zijn (in bovenstaand voorbeeld, 20 en 23 bits). Vandaar de term Variable Length. Voor mer informatie over VLSM, bekijk RFC 1009 en RFC 1812.

3 Werking van de TCP/ IP protocol suite

3.1 Routering

Het idee achter het Internet is gebaseerd op het zogenaamde "netwerk van netwerken" concept. Dit houdt in dat er enorm veel netwerken aan elkaar zijn geknoopt door middel van allerlei netwerk devices (routers), en dat computers op deze netwerken met elkaar kunnen communiceren. De vraag die gesteld kan worden is:

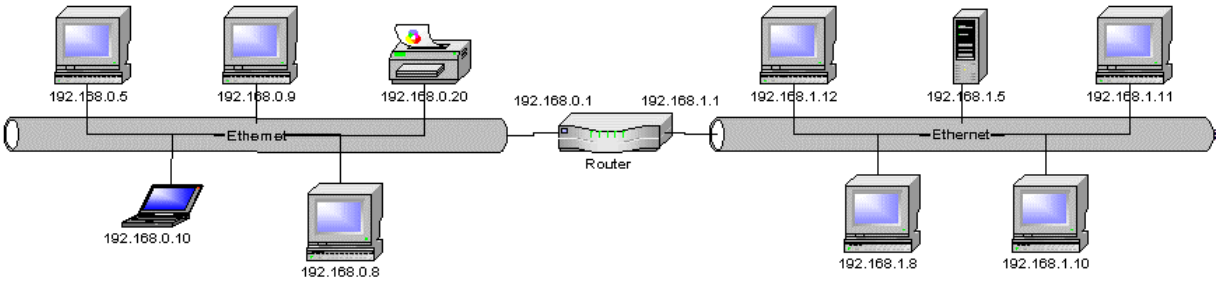
Hoe kunnen computers, die niet op hetzelfde netwerk zitten, met elkaar communiceren?

Ofwel, hoe bereikt een computer in het zuiden van Nederland een computer in het noord-westen van Nederland, of een computer in Amerika? Om daar een idee van te krijgen moeten we wat weten van **routering**. Grof gezegd is routering het correct afleveren van datagrammen tussen computers die niet op hetzelfde netwerk segment zitten. Je zult al in de gaten hebben dat hier het concept van IP adres een centrale rol zal gaan spelen. De volgende paragrafen zullen een zeer globaal overzicht geven van dit proces. Als je echt bent geïnteresseerd in de diepgaande router technieken (wie niet), dan adviseer ik je ergens wat Cisco cursusmateriaal te bemachtigen.

3.1.1 De routing tabel

We hebben in paragraaf 2.2 gezien dat subnetting resulteerde in het ontstaan van kleine deelsegmenten. Om ervoor te zorgen dat computers op deze segmenten met elkaar kunnen praten, worden deze deelsegmenten aan elkaar geknoopt door netwerk devices. Nu zijn er verschillende netwerk devices die kunnen zorgen voor koppeling van netwerksegmenten. Hiervan zijn **hubs**, **switches** en **routers** wel de meest bekendsten. Het gaat hier te ver om de precieze verschillen tussen deze netwerk devices te beschrijven. Het grootste verschil is dat een router een zogenaamd **layer 3 device** is (een router werkt op basis van informatie die in laag 3 -de netwerk laag- van het OSI model is te vinden, in ons geval het IP adres), terwijl een switch een **layer 2 device** is (een switch werkt op basis van MAC adressen) en een hub een **layer 1 device** (een hub is niets anders dan een veredeld verlengstuk van de kabel, waarbij als nevenaffect netwerkfunctionaliteit voor meer dan twee machines mogelijk wordt). Dit onderscheid is niet helemaal zuiver, doordat er layer 3 switches op de markt zijn. Globaal kan worden gezegd dat hoe hoger de laag is waarop wordt gewerkt, hoe geavanceerder de mogelijkheden zijn van het netwerk device. Een router heeft dus normaal gesproken meer (configuratie)mogelijkheden dan een switch, die op zijn beurt weer meer mogelijkheden heeft dan een hub.

Laten we als basis voor de theorie het volgende plaatje als uitgangspunt nemen, waarbij een router wordt gebruikt om twee subnets te koppelen.



Subnets 192.168.0.0 /24 en 192.168.1.0 /24 worden verbonden via een router. Doordat een router een verbinding heeft met meer dan een netwerksegment, heeft een router altijd meer dan 1 netwerkkaart nodig. Men zegt ook wel dat de router **multihomed** is. Dit kan dual-homed (2 netwerkkaarten) zijn, maar ook tri-homed (3 netwerkkaarten). De volgende optie is geloof ik quadruple-homed, maar dat soort routers zul je niet snel tegenkomen.

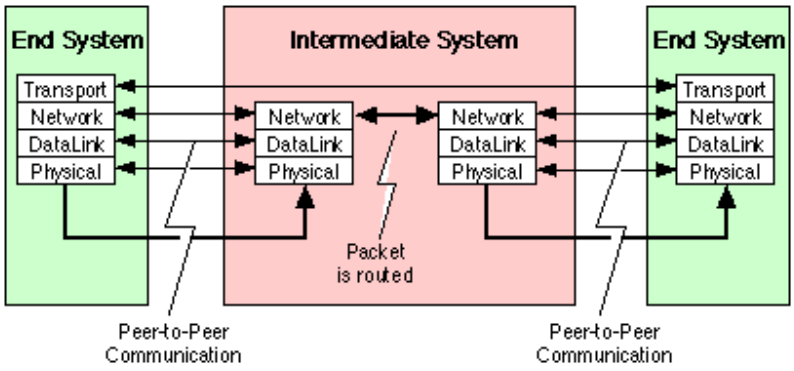
Als de host 192.168.0.5 een bericht wil versturen naar een ander segment (zij het naar subnet 192.168.1.0, zij het naar de andere kant van de wereld), dan zal de data die wordt uitgewisseld een of meerdere routers moeten passeren. Hoe gaat dit precies in zijn werk? De computer die data wil versturen zal eerst kijken waar het frame naartoe moet. Er zijn hier twee keuzes: of het frame moet naar een computer op hetzelfde segment, of naar een computer op een ander segment. Hoe bepaalt een computer dat? Dat hebben we in paragraaf 2.2.2 gezien! Door het eigen {IP adres; subnet mask} paar te vergelijken met het {IP adres; subnet mask} paar van de doelcomputer kan worden bekeken of de data op het eigen segment blijft of niet. Deze voor routing enorm belangrijke informatie wordt weergegeven in een routing tabel. Deze informatie wordt op een Windows 2000 machine weergegeven met het `route print` commando (of via het `netstat -r` commando), en ziet er als volgt uit:

```
G:\>route print
===========================================================================
Interface List
0x1 ........................... MS TCP Loopback interface
0x1000003 ...00 01 02 08 a9 a8 ...... 3Com EtherLink PCI
===========================================================================
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0      192.168.0.1     192.168.0.5       1
        127.0.0.0        255.0.0.0        127.0.0.1       127.0.0.1       1
      192.168.0.0    255.255.255.0      192.168.0.5     192.168.0.5       1
      192.168.0.5  255.255.255.255        127.0.0.1       127.0.0.1       1
    192.168.0.255  255.255.255.255      192.168.0.5     192.168.0 .5       1
        224.0.0.0        224.0.0.0      192.168.0.5     192.168.0.5       1
  255.255.255.255  255.255.255.255      192.168.0.5     192.168.0.5       1
Default Gateway:       192.168.0.1
===========================================================================
Persistent Routes:
  None
```

De eerste twee kolommen stellen de computer in staat te bepalen of het frame naar een ander segment moet of niet. Anders gezegd: de tweede kolom geeft aan welke bits van de bits van de eerste kolom

precies moeten matchen om die routing tabel regel van toepassing te laten zijn. Een logical AND van kolom 1 en 2, vergelijkbaar met wat we hebben gedaan in paragraaf 2.2.2, levert na vergelijking met de logical AND van het eigen IP adres en subnet mask het gewenste antwoord. De derde kolom vermeld het IP adres van de volgende hop. Dit kan het eigen IP adres zijn (wanneer de doelcomputer op het eigen segment ligt), of het IP adres van de router (wanneer de doelcomputer op een ander segment ligt). De interface kolom geeft aan over welke netwerkkaart het frame verzonden dient te worden.

Als de computer heeft bepaald dat het frame naar een ander segment moet, dan wordt het frame verstuurd naar de **default gateway**. De default gateway in de routing tabel is herkenbaar aan de **0.0.0.0** in zowel de Network Destination als Netmask kolom, en ontvangt de frames die naar een ander segment moeten worden verstuurd. Hoe weet de router nu naar welk segment het frame verstuurd moet worden? Welnu, de router heeft ook een (uitgebreidere) routing tabel, en daar staat informatie in die de router gebruikt voor het doorsturen van het ontvangen frame. Laten we dit aan de hand van een plaatje toelichten.



De **end systems** zijn de broncomputer en de doelcomputer. Alle tussenliggende netwerk devices zijn **intermediate systems**. De broncomputer stuurt het bericht naar de default gateway (= intermediate system). Deze kijkt naar het adres van de doelcomputer. Dit adres is te vinden in de header van de netwerk laag, en is in ons geval het IP adres van de doelcomputer, maar kan dus ook het IPX adres van een host zijn als met IPX/SPX wordt gecommuniceerd. De router herhaalt nu de stappen die de client ook heeft doorlopen. Het kijkt eerst of de doelcomputer direct kan worden benaderd. Zo ja, dan wordt het frame afgeleverd bij de eindbestemming aan het aangrenzende segment (niet het segment waar de broncomputer op zit, omdat er bij directe communicatie geen router aan te pas komt. Hierover later meer). Zo niet, dan wordt het frame doorgestuurd. Hoe weet de router nu naar welke volgende router het frame doorgestuurd moet worden? Hoe de router aan deze informatie komt gaat deze tutorial te boven. Neem echter maar aan dat routers met elkaar communiceren via allerlei protocollen (misschien heb je wel eens van **OSPF** of **RIP** gehoord), en dat een router op basis van deze ontvangen informatie kennis opdoet van de status van het netwerk en op basis hiervan zijn router tabellen aanpast. Op basis van deze aanpassingen kunnen "betere" keuzes worden gemaakt aangaande de volgende hop waar het frame naartoe moet worden gestuurd. De volgende router herhaalt weer precies dezelfde stappen, net zolang totdat het frame afgeleverd kan worden bij de eindbestemming.

Nu kun je bij jezelf denken: een router werkt op basis van laag 3 (op basis van IP adressen), hoe werkt een switch dan? Heeft die ook zoiets als een routing tabel om te beslissen waar een frame naartoe moet worden gestuurd? Het antwoord daarop is: jazeker! Er zijn echter wat kleine verschillen:

1. Men spreekt bij switches niet over een routing tabel, maar over een **MAC tabel**;

2. Aangezien een switch niet op basis van laag 3 werkt, maar op basis van laag 2, staan er in de MAC tabel van een switch geen IP adressen, maar, je raadt het waarschijnlijk al, MAC adressen.

Een switch kijkt namelijk niet naar het IP adres van een frame. Een switch is daarom ook niet geschikt voor het doorsturen van frames naar segmenten die een andere IP range hebben. Daar is een router voor nodig. Het doorsturen van een frame door een switch gebeurt op basis van het MAC adres van de doelcomputer. Stel je voor: een unicast frame komt op een bepaalde poort binnen bij de switch. De switch kijkt of het bronadres van het frame met bijbehorende poort in zijn MAC tabel staat. Als deze informatie nog niet in de MAC tabel staat, dan wordt deze informatie toegevoegd. Daarna kijkt de switch of het doeladres van het frame in zijn MAC tabel staat, en welke poort daar bij hoort. Als het MAC adres daadwerkelijk voorkomt in de MAC tabel, dan wordt het frame alleen naar de desbetreffende poort gestuurd. Als het MAC adres niet in de MAC tabel staat, dan wordt het frame doorgestuurd naar alle poorten, behalve de poort waarop het frame is ontvangen. Dit proces staat bekend onder de term **flooding**. Op het moment dat de doelcomputer een frame stuurt (bijvoorbeeld een antwoord op het ontvangen frame), dan wordt zijn MAC adres met bijbehorende poort in de MAC tabel opgenomen, zodat de volgende keer een frame naar deze computer niet naar alle poorten wordt verstuurd. Er is dan geen sprake meer van flooding, maar van **forwarding**, ofwel het naar 1 enkele poort doorsturen van het frame.

Tot zover een globaal overzicht van switching.

3.1.2 Default gateway configuratie

Als een computer een bericht wil versturen naar een ander segment met een andere IP range, dan moet er een default gateway zijn gedefinieerd. Een computer op verschillende manieren aan het IP adres van de default gateway komen. Een mogelijkheid is het handmatig invoeren van dit IP adres aan de properties van de netwerkkaart. Een andere mogelijkheid is het toewijzen van deze informatie aan de client via een Dynamic Host Configuration Protocol (**DHCP**) server[6]. Een derde mogelijkheid is die via het zogenaamde ICMP Router Discovery Protocol (**IRDP**, beschreven in RFC 1256). Bij dit protocol stuurt de router om de zoveel seconden een **Router Advertisement**, waarin het IP adres van de router staat vermeld. Hieronder volgt een gesniffed frame, dat standaard naar multicast adres 224.0.0.1 wordt verstuurd:

```
ICMP: Router Advertisement
    ICMP: Packet Type = Router Advertisement
    ICMP: Router Advertisement Code = 0 (0x0)
    ICMP: Checksum = 0x2E4C
    ICMP: Number of addresses = 1 (0x1)
    ICMP: Address entry size = 2 (0x2)
    ICMP: Lifetime = 1800 (0x708)
    ICMP: Router Address = 192.168.0.1
    ICMP: Preference Level = 0 (0x0)
```

Zoals je kunt zien zullen de clients, die zijn geconfigureerd om naar dit soort berichten te luisteren, worden geconfigureerd met een default gateway IP adres van 192.168.0.1 (het IP adres van de router op mijn eigen Ethernet netwerk). Voor de Windows 2000 liefhebbers: het toevoegen van de REG_DWORD

[6] Een DHCP server is een server die allerlei netwerkgerelateerde settings door kan voeren op een client, zoals IP adres, het IP adres van een DNS server, trailer encapsulation, time-out intervallen etc.

value **PerformRouterDiscovery** met de waarde 1 aan de registry key HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\<interface> zorgt ervoor dat de client gaat luisteren naar dit soort IRDP berichten.

3.1.3 Hardware adres versus IP adres

Zoals we nu hopelijk weten, wordt data tussen computers verzonden in de vorm van frames. Nu kunnen hier verschillende netwerktechnologieën voor worden gebruikt. Denk onder andere aan Ethernet, TokenRing en ATM (Asynchronous Transfer Mode)[7].

Als er communicatie plaatsvindt tussen computers op een Ethernet netwerk, dan gebeurt dat niet alleen via het IP adres, maar ook via het hardware adres van de netwerkkaart van de computer. Het afleveren van datagrammen in een Ethernet netwerk gebeurt op basis van een uniek identificatienummer dat staat vermeld op de netwerkkaart. Dit adres is vaak in het ROM van de netwerkkaart gebrand en wordt het **hardware adres** of **fysieke adres** van de netwerkkaart genoemd. Dit hardware adres is bij Ethernetkaarten 48 bits lang en is in Windows 2000 op te vragen via het `ipconfig /all` commando:

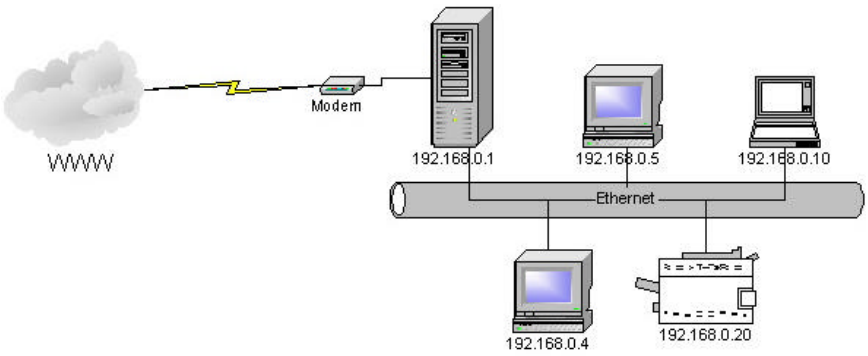
```
Ethernet adapter Connection1:

        Connection-specific DNS Suffix  . :
        Description . . . . . . . . . . . : 3Com EtherLink XL 10/100 PCI
        Physical Address. . . . . . . . . : 00-01-02-08-A9-A8
        DHCP Enabled. . . . . . . . . . . : No
        IP Address. . . . . . . . . . . . : 192.168.0.5
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.0.1
        DNS Servers . . . . . . . . . . . : 192.168.0.10
```

Nu zul je misschien denken: "Wat heb ik met een hardware adres te maken? Ik stuur toch een datagram naar een bepaald IP adres?". Klopt, en voor zover het de eindgebruiker betreft is dit hardware adres ook helemaal niet interessant. Echter, voor de correcte aflevering van datagrammen is dit adres onontbeerlijk. Op basis van het hardware adres in het Ethernet datagram weet de netwerkkaart namelijk of een bepaald datagram voor hem is bestemd of niet.

Nu hebben we een probleem. Als we vanaf computer *X* een bericht willen sturen naar computer *Y*, dan hebben we zowel het IP adres van computer *Y* nodig (dat in de IP header komt te staan), alsmede het hardware adres van zijn netwerkkaart (dan in de Ethernet header komt te staan). Het IP adres kunnen we achterhalen met het Domain Name System (DNS; zie paragraaf 3.2). Echter, hoe komen we nu aan het hardware adres van de netwerkkaart? Gelukkig hebben bepaalde mensen hier al een oplossing voor gevonden in de vorm van het **Address Resolution Protocol** (ARP). Dit protocol is beschreven in RFC 826. Aan de hand van de eerder beschreven termen direct route en indirect route (zie paragraaf 2.2.2) zal worden uitgelegd hoe ARP werkt, en hoe het een sleutelrol vervuld bij het routeringsproces. Het proces wordt uitgevoerd in een netwerk zoals dat hieronder in de diagram is weergegeven.

[7] Onderstaande paragrafen zullen zich richten op Ethernet, omdat ik toevallig een Ethernet netwerk thuis heb liggen.



3.1.4 Direct routes

De computer met IP adres 192.168.0.5 pingt de computer met IP adres 192.168.0.1 via het commando `ping 192.168.0.1`. Wat gebeurt er nu achter de schermen? Zoals beschreven in hoofdstuk 2 bepaalt de broncomputer (de computer met IP adres 192.168.0.5) eerst of het IP adres van de doelcomputer in de routing tabel staat. Als dat niet het geval is, dan controleert de broncomputer of de doelcomputer zich op hetzelfde netwerksegment bevindt als hijzelf. Dit tweede geval blijkt inderdaad het geval te zijn: de network IDs van beide hosts zijn gelijk aan 192.168.0.0 /24. Nu heeft de broncomputer het hardware adres van de computer nodig. Laten we er even vanuit gaan dat de broncomputer dit adres nog niet heeft. De broncomputer stuurt nu een **ARP Request** de deur uit. Dit is niets meer dan een vraag van computer 192.168.0.5 in de vorm van: "Kan de computer met IP adres 192.168.0.1 a.u.b. antwoorden met zijn hardware adres?". Dit pakketje wordt naar iedere computer op het netwerksegment van 192.168.0.5 verstuurd (een **broadcast** dus). Het **ARP Request** bericht ziet er als volgt uit (enkele onbelangrijke elementen zijn verwijderd):

```
ETHERNET: ETYPE = 0x0806 : Protocol = ARP:  Address Resolution Protocol
    ETHERNET: Destination address : FFFFFFFFFFFF
        ETHERNET: .......1 = Group address
        ETHERNET: ......1. = Locally administered address
    ETHERNET: Source address : 00010208A9A8
        ETHERNET: .......0 = No routing information present
        ETHERNET: ......0. = Universally administered address
    ...
ARP_RARP: ARP: Request, Target IP: 192.168.0.1
    ARP_RARP: Hardware Type = Ethernet (10Mb)
    ...
    ARP_RARP: Opcode = Request
    ARP_RARP: Sender's Hardware Address = 00010208A9A8
    ARP_RARP: Sender's Protocol Address = 192.168.0.5
    ARP_RARP: Target's Hardware Address = 000000000000
    ARP_RARP: Target's Protocol Address = 192.168.0.1
    ...
```

De FFFFFFFFFFFF als destination adres geeft aan dat het een broadcast bericht betreft. Het hardware adres van 192.168.0.5 (vermeld in het Sender's Hardware Address veld) is gelijk aan 00010208A9A8; het hardware adres van 192.168.0.1 (vermeld in het Target's Hardware Address veld) het is nog niet bekend, en is daarom vermeld als 000000000000. Bij ontvangst van dit bericht door 192.168.0.1 wordt er een antwoord op het netwerk geplaatst, te weten een **ARP Reply**. Dit antwoord ziet er als volgt uit (wederom zijn er enkele velden weggelaten):

```
ETHERNET: ETYPE = 0x0806 : Protocol = ARP:  Address Resolution Protocol
    ETHERNET: Destination address : 00010208A9A8
        ETHERNET: .......0 = Individual address
        ETHERNET: ......0. = Universally administered address
    ETHERNET: Source address : 000039FAE138
        ETHERNET: .......0 = No routing information present
        ETHERNET: ......0. = Universally administered address
    ...
ARP_RARP: ARP: Reply, Target IP: 192.168.0.5 Target Hdwr Addr: 00010208A9A8
    ARP_RARP: Hardware Type = Ethernet (10Mb)
    ...
    ARP_RARP: Opcode = Reply
    ARP_RARP: Sender's Hardware Address = 000039FAE138
    ARP_RARP: Sender's Protocol Addre ss = 192.168.0.1
    ARP_RARP: Target's Hardware Address = 00010208A9A8
    ARP_RARP: Target's Protocol Address = 192.168.0.5
```

Zoals je ziet is dit bericht geen broadcast bericht (het Destination Address veld is immers ongelijk aan FFFFFFFFFFFF), maar een **unicast** bericht naar de computer met hardware adres 00010208A9A8 (ofwel, de computer met IP adres 192.168.0.5). Het hardware adres van computer 192.168.0.1 staat vermeld in het **Sender's Hardware Address** veld, en is 000039FAE138. Als het hardware adres van de doelcomputer eenmaal is bemachtigd, kan communicatie beginnen; in dit geval het versturen van ICMP berichten. Hieronder is het eerste datagram (deels) vermeld:

```
ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
    ETHERNET: Destination address : 000039FAE138
        ETHERNET: .......0 = Individual address
        ETHERNET: ......0. = Universally administered address
    ETHERNET: Source address : 00010208A9A8
        ETHERNET: .......0 = No routing information present
        ETHERNET: ......0. = Universally administered address
ICMP: Echo: From 192.168.00.05 To 192.168.00.01
    ICMP: Packet Type = Echo
    ICMP: Echo Code = 0 (0x0)
    ICMP: Checksum = 0x455C
    ICMP: Identifier = 512 (0x200)
    ICMP: Sequence Number = 1536 (0x6 00)
    ICMP: Data: Number of data bytes remaining = 32 (0x0020)
```

Het benodigde hardware adres staat vermeld in de Ethernet header van het ping pakketje. Al deze dingen zien er bij de Windows versie van tcpdump als volgt uit:

```
G:\windump>windump -n
windump: listening on\Device\Packet_{FAB9430F-E628-4616-895A-1D5C23FE7EF3}
15:07:33.247879 arp who-has 192.168.0.1 tell 192.168.0.5
15:07:33.248271 arp reply 192.168.0.1 is-at 0:0:39:fa:e1:38
15:07:33.248291 192.168.0.5 > 192.168.0.1: icmp: echo request
15:07:33.248646 192.168.0.1 > 192.168.0.5: icmp: echo reply
15:07:34.245040 192.168.0.5 > 192.168.0.1: icmp: echo request
15:07:34.245502 192.168.0.1 > 192.168.0.5: icmp: echo reply
...
```

Als er nu weer met 192.168.0.1 gecommuniceerd moet worden, dan zou het makkelijk zijn als niet opnieuw met ARP requests en ARP replies gewerkt hoefde te worden. Het zou dus makkelijk zijn om de mapping van IP adres naar hardware adres tijdelijk op te slaan op zowel de doelcomputer (192.168.0.1) als de broncomputer (192.168.0.5). Dit gebeurt ook. De mappings worden opgeslagen in een zogenaamde **ARP cache**. Dit is een verzameling mappings in de vorm {IP adres; hardware adres}. Als een bericht moet worden verstuurd door een computer, dan kijkt deze eerst in deze ARP cache. Als hier het hardware adres van de doelcomputer al in staat vermeld, dan wordt dit hardware adres eruit gehaald, en kan de uitwisseling van ARP berichten achterwege blijven. Als het hardware adres van de doelcomputer niet in de ARP cache staat vermeld, dan zal een broadcast ARP request worden verstuurd. De aanwezige hardware adressen blijven standaard enkele minuten in de ARP cache, waarna ze worden verwijderd. Deze tijdsduur is handmatig in de registry aan te passen (voor de geïnteresseerden: de value ArpCacheLife in key "HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters"). De inhoud van de cache is vluchtig: herstarten van een computer zorgt voor een geheel lege ARP cache.

De ARP cache kan met het `arp -a` commando zichtbaar worden gemaakt:

```
G:\>arp -a

Interface: 192.168.0.5 on Interface 0x1000003
  Internet Address      Physical Address      Type
  192.168.0.1           00-00-39-fa-e1-38     dynamic
```

3.1.5 Indirect routes

We hebben net gezien hoe communicatie plaatsvindt als de doelcomputer zich op hetzelfde netwerksegment bevindt als de broncomputer. Men spreekt dan, zoals we hebben gezien, over een **direct route**. Wat gebeurt er nu als de doelcomputer zich op een ander netwerksegment bevindt (een **indirect route**)? We krijgen nu te maken met de default gateway. In dit voorbeeld is de default gateway gelijk aan 192.168.0.1. Op deze machine is routing software geïnstalleerd en geconfigureerd, zodat deze computer kan fungeren als router. Dit stuk software zorgt kortom voor een correcte routering van de inkomende datagrammen.

Laten we eens een voorbeeld bekijken waarbij een ping wordt verstuurd van de computer met IP adres 192.168.0.5 naar www.dsinet.org.

Computer 192.168.0.5 stuurt een serie pings. Aangezien er geen direct route bestaat tussen de broncomputer en de webserver van dsinet, moeten de datagrammen worden verstuurd naar de default gateway. Ook hierbij geldt: we hebben het hardware adres nodig van de volgende hop. In dit geval is dat het hardware adres van de netwerkkaart van de default gateway! Hoe nu verder? Nou, de broncomputer kijkt eerst of dit hardware adres in zijn ARP cache staat. Zo niet, dan wordt er een ARP Request de deur uit gedaan met de vraag: "Kan de computer met IP adres 192.168.0.1 reageren met zijn hardware adres?". Na ontvangst van het correcte hardware adres wordt het frame naar de default gateway verstuurd. Deze gateway dient er voor te zorgen dat het verstuurde frame richting doelcomputer wordt verstuurd. Dit kan op een aangrenzend netwerksegment zijn, maar dit kan ook 9 netwerksegmenten verder liggen. De router (default gateway) herhaalt nu de stap die de client heeft uitgevoerd: het bekijkt of er een direct route mogelijk is tussen hemzelf en de broncomputer (de webserver). Zo ja, dan kan de datagram worden afgeleverd bij de eindbestemming. Zo niet, dan wordt de datagram naar een volgende router verstuurd, die het hele proces weer herhaalt. Uiteindelijk zal het frame op zijn eindbestemming aankomen. Het ping proces heeft in mijn geval de volgende stappen gegenereerd:

```
G:\windump>windump –n
windump: listening on\Device\Packet_{FAB9430F-E628-4616-895A-1D5C23FE7EF3}
15:10:09.663155 arp who-has 192.168.0.1 tell 192.168.0.6
15:10:09.663463 arp reply 192.168.0.1 is-at 0:0:39:fa:e1:38
15:10:09.663485 192.168.0.6.1203 > 192.168.0.1.53:  99+ A? www.dsinet.org. (32
15:10:09.908001 192.168.0.1.53 > 192.168.0.6.1203:  99 2/3/2 (205)
15:10:09.911231 192.168.0.6 > 213.201.155.186:  icmp: echo request
15:10:11.101217 192.168.0.6 > 213.201.155.186:  icmp: echo request
15:10:12.102650 192.168.0.6 > 213.201.155.186:  icmp: echo request
15:10:13.104041 192.168.0.6 > 213.201.155.186:  icmp: echo request
```

Dit is een combinatie van dingen die we hebben besproken:

- Er wordt een ARP Request verzonden en een ARP Reply ontvangen om achter het fysieke adres van de router te komen;
- Er wordt een DNS query verstuurd om achter het IP adres van www.dsinet.org te komen (dit gaat over poort 53). Hierover in de volgende paragraaf meer;
- De ICMP berichten worden de deur uit gestuurd. Helaas krijgen we geen antwoord van de server. Hoogstwaarschijnlijk worden bepaalde ICMP datagrammen gefilterd. Waar dat gebeurt is uit dit overzicht niet af te leiden.

Er is nu 1 stelling die je dient te onthouden als je over direct routes en indirect routes praat:

> *Als we te maken hebben met direct routes, dan behoren het hardware adres in de Ethernet header en het IP adres in de IP header van dezelfde frame toe aan **dezelfde** machine. Als er sprake is van een indirect route, dan behoort het hardware adres toe aan de eerstvolgende hop (de intermediate system), terwijl het IP adres toebehoort aan de doelcomputer.*

3.1.6 Private IP adressen

Als een computer niet aan het Internet is verbonden, dan maakt het niet uit welke range van IP adressen wordt gehanteerd op een netwerk. Echter, wanneer een bedrijf wel het Internet op gaat, dan is het voor identificatie belangrijk een uniek nummer toebedeeld te krijgen van een ISP. Om er nu voor te zorgen dat niet het hele IP nummerplan herschreven en opnieuw geïmplementeerd moet worden, is het aan te bevelen een IP adres te kiezen uit de **private IP address range**, zoals gedefinieerd in RFC 1918. In deze RFC staan drie IP ranges vermeld die niet worden gerouteerd door de routers op het Internet. Door een van deze drie ranges te kiezen voor interne nummering is het mogelijk dat interne hosts wel het Internet op kunnen (bijvoorbeeld via gebruik van een proxy server, of via een Network Address Translation (NAT) constructie), terwijl computers op het Internet niet of slechts heel moeilijk het interne netwerk kunnen bereiken. De drie IP ranges die door de IANA zijn gespecificeerd zijn:

- 10.0.0.0 /8;
- 172.16.0.0 /12;
- 192.168.0.0 /16.

Vraag

Welke IP adressen behoren allemaal tot de range 172.16.0.0 /12?

Uitwerking

De binaire representatie van 172.16.0.0 is gelijk aan

```
10101100  00010000  00000000  00000000
```

Hiervan liggen de rode bits vast (de subnet mask is immers gelijk aan 12 bits). Het derde en vierde byte kunnen variëren tussen 0 en 255. Echter, welke waarde kan de tweede byte aannemen? Hieronder zijn de mogelijke waarden uitgewerkt. De rode bits liggen wederom vast.

```
00010000     =>    16
00010001     =>    17
00010010     =>    18
00010011     =>    19
00010100     =>    20
...
00011100     =>    28
00011101     =>    29
00011110     =>    30
00011111     =>    31
```

Zoals is te zien kan de waarde van het tweede byte niet boven de 31 uitkomen. Oftewel, de range loopt van 172.16.0.0 /12 tot en met 172.31.255.255 /12.

3.2 Domain Name System (DNS)

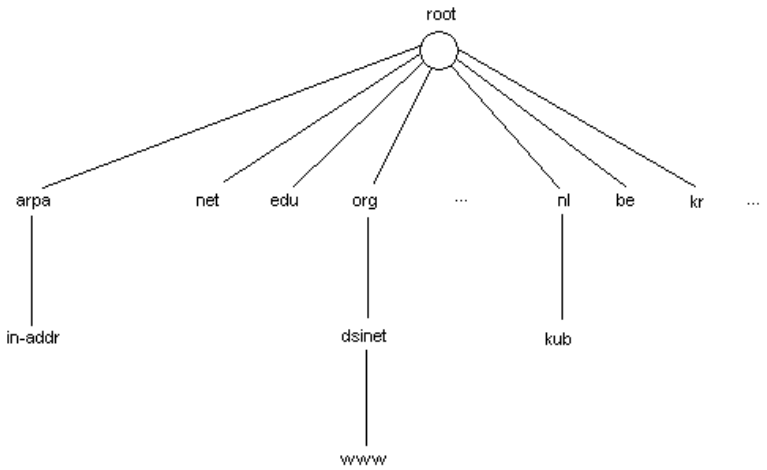
Stel je voor dat je de website van DSINet wilt bezoeken. Je typt de URL http://www.dsinet.org/ in, en wonder boven wonder verschijnt de pagina van DSINET voor je neus. Hoe kan dat? Computers kunnen, in tegenstelling tot mensen, niet met de string die jij hebt ingetypt omgaan. Computers werken op basis van IP adressen, niet op basis van makkelijk te onthouden namen. Er heeft dus schijnbaar een vertaalslag plaatsgevonden van de string www.dsinet.org naar IP adres 213.201.155.186 (het IP adres van de website van DSINET). Hoe ging dat nu precies in zijn werk? Een overzicht.

3.2.1 DNS lookup

Vroeger, toen we allemaal nog jong en onbezonnen waren, bestond er ng geen DNS structuur zoals we die nu kennen. In plaats daarvan was er op een werkstation een bestand te vinden, waarin alle IP adressen stonden (b.v. 213.201.155.186) met bijbehorende naam (b.v. www.dsinet.org). Dit zogenaamde **HOSTS bestand** is op een Windows machine nog steeds te gebruiken, en is te vinden in de %systemroot%\system32\drivers\etc folder. De twee voorbeelden die erin staat zien er als volgt uit:

```
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com               # x client host
```

Ze worden voorafgegaan door een #, hetgeen betekent dat ze als commentaar zijn opgenomen (ze doen dus niets), maar het idee is hopelijk duidelijk. Een lijst zoals dit is natuurlijk met de huidige groei van het Internet niet meer bij te houden, dus moest er een andere oplossing komen. Die kwam er in de vorm van een gedistribueerde opzet. Dit houdt in dat meerdere computers verantwoordelijk zijn voor het conversie proces. Om dit te begrijpen moet de structuur in kaart worden gebracht zoals die nu wordt gebruikt. Deze structuur staat vermeld in volgend plaatje.



Helemaal bovenaan de DNS hiërarchie is de **root** (ook wel de **unnamed root** genoemd, doordat het geen naam heeft). Daaronder bevinden zich de top-level domeinen (**TLD**s), die worden onderverdeeld in:

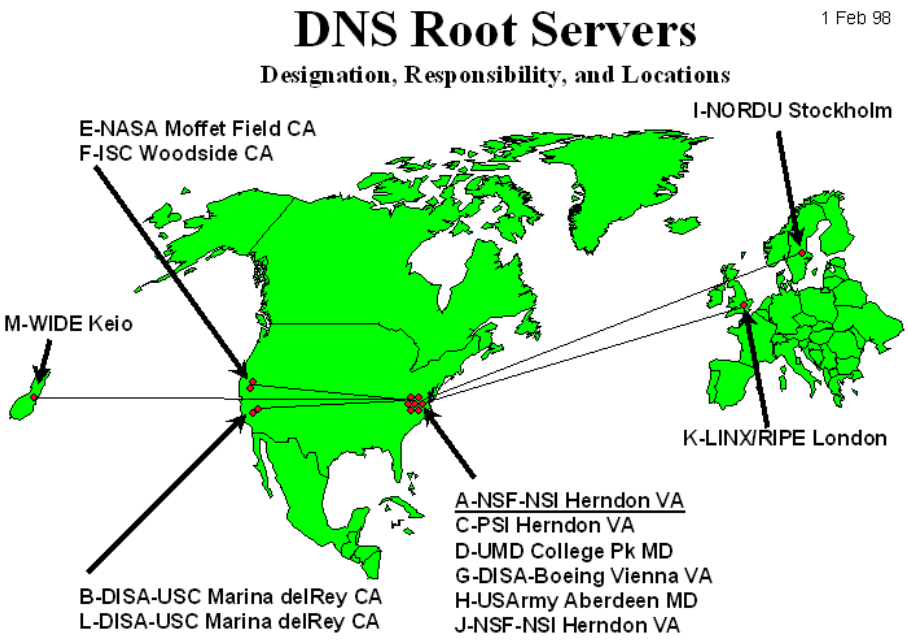
- de **generieke domeinen** (ook wel **organisatie domeinen** genoemd), zoals com, edu en org;
- de **landdomeinen** (of **geografische domeinen**), zoals it, cz en nl;
- het **arpa** domein, dat wordt gebruikt voor naam-naar-IP adres conversie;
- zeven **nieuwe domeintypen**, te weten .biz, .info, .name, .museum, .coop, .aero en .pro[8];
- het **IP6.INT domein**, voor IP versie 6 reverse lookups.

Hoe gaat de conversie van host naam naar IP adres nu in zijn werk als ik bijvoorbeeld www.google.com intik? Laten we eerst kijken wat er gebeurt als we nog nooit op www.google.com zijn geweest. Mijn computer vraagt nu aan een bepaalde computer: "kun jij mij het IP adres geven dat is toegewezen aan de machine www.google.com?". De server waaraan dat wordt gevraagd wordt een **DNS server** genoemd. Dit is een server die bevoegd is om conversies uit te voeren voor een bepaald deel van de DNS hiërarchie, en die mij is toegewezen door mijn service provider. De bevoegdheid van deze DNS server is beperkt tot de IP adressen waar het verantwoordelijk voor is. Daar hoort www.google.com zeker niet bij. De DNS server kan mij het antwoord dus niet direct geven, en roept daarom de hulp in van andere DNS servers. Welke zijn dat? De server is geconfigureerd met de IP adressen die verantwoordelijk zijn voor de top van de DNS hiërarchie, en daar gaat hij eerst naartoe. Welke DNS servers zijn dat? Een blik in de **cache.dns** file op de DNS server, alsmede een beetje zoeken op Internet geeft de volgende informatie:

Naam DNS server	**IP adres**
a.root-servers.net	198.41.0.4
b.root-servers.net	128.9.0.107
c.root-servers.net	192.33.4.12
d.root-servers.net	128.8.10.90
e.root-servers.net	192.203.230.10
f.root-servers.net	192.5.5.241
g.root-servers.net	192.112.36.4
h.root-servers.net	128.63.2.53
i.root-servers.net	192.36.148.17
j.root-servers.net	198.41.0.10
k.root-servers.net	193.0.14.129
l.root-servers.net	192.32.64.12
m.root-servers.net	2002.12.27.33

Nog DoS IP adressen nodig?

[8] Voor meer informatie over deze nieuwe top-level domeinen, ga naar http://www.icann.org/tlds/.



Een van deze DNS servers wordt de vraag voorgelegd: "Weet jij misschien het IP adres van www.google.com?". De betreffende server heeft echter geen bevoegdheid voor het google.com domein, en zal dus reageren met: "Ik weet het niet, maar ik heb wel het IP adres voor je van een server die dichter bij het antwoord ligt, te weten de DNS server die verantwoordelijk is voor het com domein. Alsjeblieft, probeer het daar maar.". Mijn DNS server gaat naar de DNS server die verantwoordelijk is voor het com domein (wiens IP adres hij net heeft ontvangen), en vraagt aan hem: "Weet jij het IP adres van www.google.com?". Ook deze DNS server is niet bevoegd voor dat deel van de hiërarchie en zal dus het antwoord niet weten, maar geeft me wel het IP adres van een DNS server die op zijn beurt weer dichter bij het antwoord ligt. In dit geval is dat een verwijzing naar de DNS server die bevoegd is voor het uitvoeren van naamsconversies voor het google.com domein. Deze DNS server (met als IP adres 216.239.34.10) stuurt mijn DNS server het IP adres van de server www.google.com, die op zijn beurt het IP adres doorstuurt naar mijn computer, waardoor ik verbinding kan gaan maken met www.google.com. Als we de datagrammen nu eens van het netwerk af gaan halen, dan zien we het volgende.

```
DNS: 0x2D:Std Qry for www.google.com. Of type Host Addr on class INET addr.
    DNS: Query Identifier = 45 (0x2D)
    DNS: DNS Flags = Query, OpCode  - Std Qry, RD Bits Set, RCode  - No error
        DNS: 0............... = Request
        DNS: .0000........... = Standard Query
        DNS: .....0.......... = Server not authority for domain
        DNS: ......0......... = Message complete
        DNS: .......1........ = Recursive query desired
        DNS: ........0....... = No recursive queries
        DNS: .........000.... = Reserved
        DNS: ............0000 = No error
    DNS: Question Entry Count = 1 (0x1)
    DNS: Answer Entry Count = 0 (0x0)
    DNS: Name Server Count = 0 (0x0)
```

```
    DNS: Additional Records Count = 0 (0x0)
    DNS: Question Section: www.google.com. of type Host Addr o n class INET addr
        DNS: Question Name: www.google.com.
        DNS: Question Type = Host Address
        DNS: Question Class = Internet address class
```

Hier zie je een deel van het verstuurde frame. Je ziet duidelijk dat er een aanvraag de deur uit gaat voor www.google.com. Het antwoord dat ik terug krijg van mijn DNS server ziet er als volgt uit:

```
DNS: Additional Records Section: NS2.google.com. of type Host Addr on class INET
addr.(4 records present)
        DNS: Resource Record: NS2.google.com. of type Host Addr on class INET addr.
            DNS: Resource Name: NS2.google.com.
            DNS: Resource Type = Host Address
            DNS: Resource Class = Internet address class
            DNS: Time To Live = 84616 (0x14A88)
            DNS: Resource Data Length = 4 (0x4)
            DNS: IP address = 216.239.34.10
        DNS: Resource Record: NS1.google.com. of type Host Addr on class INET addr.
            DNS: Resource Name: NS1.google.com.
            DNS: Resource Type = Host Address
            DNS: Resource Class = Internet address class
            DNS: Time To Live = 1647 (0x66F)
            DNS: Resource Data Length = 4 (0x4)
            DNS: IP address = 216.239.32.10
        DNS: Resource Record: NS3.google.com. of type Host Addr on class INET addr.
            DNS: Resource Name: NS3.google.com.
            DNS: Resource Type = Host Address
            DNS: Resource Class = Internet address class
            DNS: Time To Live = 83860 (0x14794 )
            DNS: Resource Data Length = 4 (0x4)
            DNS: IP address = 216.239.36.10
        DNS: Resource Record: NS4.google.com. of type Host Addr on class INET addr.
            DNS: Resource Name: NS4.google.com.
            DNS: Resource Type = Host Address
            DNS: Resource Class = Internet address class
            DNS: Time To Live = 79828 (0x137D4)
            DNS: Resource Data Length = 4 (0x4)
            DNS: IP address = 216.239.38.10
```

Dit zijn allemaal zogenaamde **authoritative name servers**, hetgeen wil zeggen dat ze bevoegd zijn om queries op te lossen voor dit bepaalde deel van de DNS hiërarchie. Je ziet dat er meerdere DNS server geldig zijn voor het geven van het IP adres. Vaak zijn er in een organisatie meerdere DNS servers aanwezig die allemaal een conversie mogen uitvoeren voor een bepaald deel van de DNS hiërarchie, om ervoor te zorgen dat de continuïteit gewaarborgd blijft. Stel je eens voor dat iemand per ongeluk de stekker uit de DNS server schopt. Einde DNS verkeer! Men spreekt in het geval van een enkele DNS server van een **single point of failure** (SPOF), hetgeen niet acceptabel is.

In dezelfde frame wordt het IP adres vermeld van www.google.com:

```
DNS: Answer section: www.google.com. of type Host Addr on class INET addr.
    DNS: Resource Name: www.google.com.
    DNS: Resource Type = Host Address
    DNS: Resource Class = Internet address class
    DNS: Time To Live = 275 (0x113)
    DNS: Resource Data Length = 4 (0x4)
    DNS: IP address = 216.239.37.100
```

Kortom: de volgende stap is het maken van een verbinding met IP adres 216.239.37.100 via een three-way handshake, waarna het programma (in dit geval mijn web browser) de pagina toont die ik wilde hebben. Met dank aan het DNS systeem.

3.2.2 Caching

Er zijn enkele punten die aangevuld moeten worden, waaronder het concept **caching**. Caching houdt in dat resultaten van vorige DNS queries lokaal worden opgeslagen in de loke **DNS cache**. Dat werkt tijdbesparend. Als ik nu weer een verbinding wil maken met www.google.com, dan hoef ik niet eerst te wachten op het antwoord van de DNS server die bevoegd is voor het google.com domein, maar kan ik meteen uit mijn cache het juiste IP adres halen van de www.google.com server, waarna direct een verbinding wordt gemaakt. Deze cache wordt in Windows 2000 getoond via het `ipconfig /displaydns` commando en bevat o.a. de volgende entry:

```
ns3.google.com.
----------------------------------------
  Record Name . . . . . : NS3.google.com
  Record Type . . . . . : 1
  Time To Live  . . . . : 84702
  Data Length . . . . . : 4
  Section . . . . . . . : Answer
  A (Host) Record . . . : 216.239.36.10
```

En da's precies hetzelfde IP adres als staat gedefinieerd in bovenstaande frame.

De entries die in de DNS cache staan vermeld blijven daar niet eeuwig staan. Na een poosje worden ze automatisch verwijderd. De tijdsperiode die een entry in de DNS cache blijft zitten wordt geregeld via de MaxCacheEntryTtlLimit registry entry, en is aan te passen naar believen.

3.2.3 Resource Records

We hebben net gezien dat de DNS servers verantwoordelijk zijn voor conversie van host naam naar IP adres. De DNS server doet echter meer dan alleen dat. Zo voorziet het op een Windows 2000 machine clients in verwijzingen naar allerlei belangrijke machines, en zorgt het voor verwijzingen naar mail servers, zodat mail correct kan worden afgehandeld. Hoe wordt deze informatie opgeslagen in de DNS server? Dat gaat met **resource records**. Enkele voorbeelden van resource records staan hieronder (aan de resource records die hier staan vermeld zou ik eigenlijk niet moeten kunnen komen. Zie de volgende paragraaf over zone transfers hoe ik de informatie toch wist te bemachtigen).

```
ipv6-router             AAAA        3ffe:8114:2000:6a0:210:7bff:fe30:d250
news                    CNAME       news.energis.nl
overflow                MX          100  overflow.concepts .nl
concepts.nl.            NS          dns.conceptsfa.nl
administratie           A           213.197.30.25
```

De resource records worden aangeduid met een bepaalde lettercombinatie, afhankelijk van wat ze precies doen. Zo is het A resource record verantwoordelijk voor de conversie van host naam naar IP adres (een zogenaamde **forward lookup**, in tegenstelling tot een **reverse lookup**, die de conversie van IP adres naar host name bewerkstelligt en die m.b.v. het PTR resource record mogelijk wordt gemaakt), terwijl het MX resource record message routing naar een mail exchanger host voor zijn rekening neemt. Het CNAME resource record staat de systeembeheerder toe om naar een host te laten verwijzen die een andere naam heeft.

Voorbeeld

De web servers die op het internet staan zullen niet allemaal de naam "www" hebben, maar een implementatie-afhankelijke naam (bijvoorbeeld QKL-WS6). Om deze informatie niet vrij te geven, wordt er een CNAME record aangemaakt. Om toch de naam "www" te kunnen gebruiken voor een host, wordt het volgende CNAME record in DNS gezet:

```
www.webservices.com           CNAME      QKL-WS6.webservices.com
```

Zodoende geeft het bedrijf niets weg over interne naamgevingsconventies, en wordt de bezoeker niet lastig gevallen met onmogelijke host namen die hij niet kan onthouden. Zoek voor een totaal overzicht van verschillende resource records op het Internet. Er zijn er nog veel meer.

3.2.4 Zone transfer

Om te begrijpen wat een zone transfer is, moet eerst het begrip zone worden uitgelegd. Een **zone** is een deel van DNS hiërarchie waarvoor een DNS server bevoegd is om queries op te lossen. Dit kan overeenkomen met een heel domein, bijvoorbeeld het dsinet domein, maar dat hoeft helemaal niet zo te zijn. Een zone kan ook meerdere domeinen beslaan. Informatie over een zone is opgeslagen in **zone files** (= **DNS database files**), die op hun beurt resource records bevatten.

Zoals vermeld is het verstandig om meer dan 1 DNS server te hebben, omdat anders een single point of failure ontstaat. Om een SPOF te voorkomen worden meestal twee of meer DNS servers ingezet. Als er nu veranderingen worden aangebracht op de ene DNS server, dan moeten die veranderingen naar de andere DNS server worden gekopieerd, zodat alle DNS servers in het netwerk up-to-date blijven met de gedefinieerde resource records. Dit kopiëren van resource records wordt een **zone transfer** genoemd. Nu getuigt het van een juiste beveiligingshouding als slechts bepaalde computers mogen deelnemen aan dit zone transfer proces. Wat niet de bedoeling is, is dat iedereen alle zones van een computer kan kopiëren. Sommige ISPs denken daar anders over, waardoor ik de nodige resource records naar mijn computer kan kopiëren. Nu vallen de extra hack mogelijkheden die ik met deze extra informatie heb wel mee, maar het is slordig, en geeft onnodig veel informatie aan buitenstaanders. Stel je voor dat ik resource records onder mijn neus krijg met de namen "test", "firewall", "financien" etc. Dat geeft leuke mogelijke invalshoeken voor een aanval, doordat testmachines meestal niet extreem goed zijn beveiligd en computers met financiële informatie gewilde objecten zijn om te kraken.

Een zone transfer uitvoeren is makkelijk. Hieronder staan de stappen die zijn doorlopen op een Windows 2000 machine.

```
G:\>nslookup
DNS request timed out.
    timeout was 2 seconds.
*** Can't find server name for address 192.168.1.100: Timed out
Default Server:  dns.conceptsfa.nl
Address:  213.197.28.3

> server 213.197.28.3
DNS request timed out.
    timeout was 2 seconds.
Default Server:  [213.197.28.3]
Address:  213.197.28.3

> ls -d concepts.nl > concepts.txt
[[213.197.28.3]]
###############################################################
Received 3207 records.
>_
```

Ik had de mededeling moeten krijgen dat ik niet mocht kopiëren. Niet dus.