Computernetwerken III
Oplossingen van modelvragen

Tim Besard Dimitri Roose

14 juni 2010

Hoofdstuk 1

Reeks A

1.1 Configuratie van netwerkinterfaces (§1.2) en bridging (labnota's)

Bespreek alle opdrachten (ook hun opties en hun output) en configuratiebestanden (inclusief locatie en inhoud) die onder Linux voor de configuratie van de netwerkinterface kunnen gebruikt worden. Vergeet de opstartbestanden niet. Wat is IP-aliasing en op welke manieren kan dit ingesteld worden?

Voor de IP-configuratie van Linux netwerkinterfaces, zijn er twee grote commandos beschikbaar: `ifconfig` en `ip addr`.

ifconfig Dit is het oudste commando, dat gebruikt kan worden om netwerkinterfaces te configureren. De eerste parameter moet het device in kwestie zijn. Het command uitvoeren zonder extra parameters resulteert in informatie over dat device. Als de vermelding van het device ontbreekt, wordt informatie over alle actieve interfaces weergegeven (om ook de offline interfaces weer te geven is de "-a" parameter benodigd). Het commando kan ook gebruikt worden om instellingen van een interface te wijzigen, door de vermelding ervan te combineren met bepaalde parameters:

- `inet` $ADRES: instellen van het IP-adres, waarbij het "inet" sleutelwoord kan eventueel weggelaten worden
- `up` of `down`: starten of stoppen van het device
- `netmask` of `broadcast` $ADRES: instellen van een netmask of broadcast adres
- `[-]multicast`: dient voor het in- of uitschakelen van multicastfunctionaliteit
- `[-]promisc`: in of uitschakelen van promiscuous mode
- `pointtopoint` $ADRES: configuratie van een PPT-tunnel (wordt meestal alternatieve software voor gebruikt)

ip addr Dit is een recenter alternatief, dat wanneer aangeroepen zonder extra parameters een overzicht geeft van alle aanwezige netwerkdevices en hun configuratie. Het maakt het ook eenvoudiger toe is om verschillende IP-adressen toe te wijzen aan een bepaald device, vb. `ip addr add 10.0.0.2/24 dev eth1` (dit heet IP-aliasing). worden met behulp van het `iwconfig` commando.

Bespreek het equivalent onder Windows Server, zowel via de Command Prompt, als via de grafische interface.

Grafische interface Configuratie van netwerkinterfaces wordt bij een Windows Server hoofdzakelijk via de grafische interface gedaan, die toegankelijk is door in het Configuratiescherm op *Network Connections* te klikken. Hier kunnen verbindingen hernoemd worden, door er rechts op te klikken en *Rename* te kiezen. Verdere configuratie wordt uitgevoerd door in het *Properties* menu van de verbinding te gaan, en daar het juiste tabblad te selecteren. Monitoring van een verbinding wordt uitgevoerd door in het rechterklikmenu *Status* te kiezen.

Command prompt Standaard is de command prompt bruikbaar om instellingen te bekijken (via het `ipconfig` commando, met parameter "/all" voor gedetailleerde informatie) en om de verbinding te monitoren (via het `netstat` commando, met parameters "-e" voor Ethernet-statistieken, en "-s" voor protocolstatistieken, eventueel gecombineerd met "-p $PROTOCOL" om 1 of meerdere specifieke protocollen te selecteren).

Om configuratie vanaf de command-line interface uit te voeren moet de *Routing and Remote Access* service geactiveerd zijn. Is dit het geval, kan het `netsh` commando gebruikt worden om het netwerk te configureren.

```
netsh interface ip show ipaddress
netsh interface ip set address eth0 static 192.168.0.1 255.255.255.0
```

Hoe bepalen een verzameling bridges, die diverse netwerksegmenten tot één enkel subnetwerk groeperen, welke van hun poorten moeten geblokkeerd worden?

Dit wordt gerealiseerd door middel van het *spanning tree* algoritme, wat na uitvoering alle lussen in het netwerk uitschakelt zodat een eenduidige doorstroom van pakketten kan bereikt worden. Het algoritme werkt als volgt:

1. Detectie van de root bridge: aan de hand van het MAC-adres wordt 1 van de bridges gepromoveerd tot root-bridge.

2. Toekennen kostprijzen en root-port aanduiden: elke verbinding zal nu een kostprijs toegekend worden, zodat elke bridge een cumulatieve kostprijs kan berekenen om de root-bridge te bereiken. De poort langs waar de root-bridge het goedkoopst bereikt kan worden (het minimum van alle cumulatieve kostprijzen dus), heet men de *root-port*. De uiteindelijke kostprijs van elke bridge is dan de kost om de root-bridge via de root-port te bereiken

3. Designated port selecteren: per subnetwerk zal vervolgens de poort aangeduid worden die zal gebruikt worden om de root-bridge te bereiken, gemarkeerd worden als *designated port*. Dit is logischerwijs een poort van de goedkoopste bridge op het subnetwerk (met de kost zijnde de totaalprijs om uit die bridge de root te bereiken).

4. Uitschakelen overige poorten: alle poorten die geen *designated port* of *root port* zijn worden uitgeschakeld.

Hoe kun je, via de uitvoer van één enkele opdracht, op een als bridge geconfigureerd Linux toestel op diverse manieren controleren dat de juiste poorten geblokkeerd zijn, en om welke reden de andere poorten mogen geactiveerd blijven?

De topologie van een bridge kan bekeken worden via het `brctl showstp` commando. Als voorbeeld, de uitvoer van het commando `brctl showstp brconn`:

```
brconn
 bridge id              8000.56b283b000c3
 designated root        8000.02b2c474422e
 root port                 3              path cost            100
 max age                  20.00           bridge max age       20.00
 hello time                2.00           bridge hello time    2.00
 forward delay            15.00           bridge forward delay 15.00
 ageing time             300.00
 hello timer               0.00           tcn timer            0.00
 topology change timer     0.00           gc timer             183.29
 flags


eth0 (1)
 port id                8001              state                forwarding
 designated root        8000.02b2c474422e path cost           100
 designated bridge      8000.56b283b000c3 message age timer    0.00
 designated port        8001              forward delay timer 0.00
 designated cost         100              hold timer           0.00
 flags

eth1 (2)
 port id                8002              state                blocking
 designated root        8000.02b2c474422e path cost           100
 designated bridge      8000.2ac74dd8ebbc message age timer    19.04
 designated port        8002              forward delay timer 0.00
 designated cost         100              hold timer           0.00
 flags
```

```
eth2 (3)
 port id                8003                state                forwarding
 designated root        8000.02b2c474422e   path cost            100
 designated bridge      8000.02b2c474422e   message age timer    19.05
 designated port        8003                forward delay timer  0.00
 designated cost           0                hold timer           0.00
 flags
```

1.2 Statische routing (§1.3 zonder subsecties)

Bespreek het doel van routing, de werking, en de belangrijkste componenten van statische routing. Behandel de terminologie en problematiek die het routing proces kenmerkt.

Doel Door routing geeft men aan eindgebruikers te indruk dat ze allen verbonden op één en hetzelfde netwerk, ook al zijn ze fysiek aangesloten op verschillende netwerken met uiteenlopende technologieën. Bij statische routing configureert men de routetabellen daarvoor volledig manueel.

Componenten Het netwerk bestaat uit verschillende *gateways* of *routers*: computers die tegelijk deel uitmaken van meerdere LAN en/of WAN netwerken en bereid zijn berichten van het ene netwerk naar het andere door te sturen. Aangezien dit meestal een zware taak is, wordt hiervoor meestal specifieke hardware gebruikt (hardware router). Soms wordt de taak echter toch door een reguliere PC uitgevoerd (gateway of software router).

Een router heeft een *routetabel* om te kunnen bepalen hoe berichten moeten doorgestuurd worden. Een routetabel bestaat uit verschillende entries, namelijk: de aanduiding van het subnetwerk (vergezeld van subnetmasker of prefixlengte), het adres van de router naar waar de berichten moeten gestuurd worden, een metriek die de kostprijs van het pad indiceert, en een time-to-live die de geldigheid van de entry aanduidt.

Werking Als een router een pakket krijgt, wordt er gekeken of de bestemmeling rechtstreek te bereiken is op het netwerk waarop de router resideert. Is dit het geval, dan spreekt men van *direct delivery*.

Wanneer een router een pakket niet rechtstreeks kan bezorgen, moet een route uit de routetabel geselecteerd worden om het pakket te verwerken (*indirect delivery*). Hierbij wordt de route genomen waarmee het meest specifieke subnetwerk te bereiken is met de laagste metriek. Vindt men zo geen route, wordt een *ICMP Destination Unreachable* bericht teruggestuurd (het *Internet Control Message Protocol* wordt gebruikt om de afzender op de hoogte stellen van netwerkproblemen).

Het is de gezamelijke taak van alle routers op het internetwerk om ervoor te zorgen dat berichten doorgestuurd worden opdat ze uiteindelijk een router bereiken die zich op

het netwerk van de eindbestemming bevindt.

Terminologie

- Routing: het doorsturen van berichten van de ene router naar de andere.
- Hops: de keten van routers die het bericht doorloopt om bij zijn bestemming te geraken.
- Hopafstand: het minimale aantal hops tussen twee toestellen nodig om een bericht te bezorgen.
- Diameter: de maximale hopafstand van een netwerk.
- Routingtabel: een lijst gegevens die voor elke router aanduiden hoe die berichten moet doorsturen (rechtstreeks afleveren, forwarden, welke interface, ...).
- Metriek: het getal dat voor elk aanduidt wat de kost is om een bericht erlangs te sturen.
- Lifetime: een aanduiding hoe lang de route-entry nog geldig is.
- TOS routing: het doorsturen van berichten waarbij het gekozen pad afhankelijk is van het *Type of Service* veld.

Problematiek Het configureren van routetabellen, dat voor elke router moet herhaald worden, is een tijdrovend en foutgevoelig proces. Zo is het eenvoudig om *routing loops* te creëeren, waarbij bepaalde pakketten een pad volgen dat verwijst naar één van de intermediaire routers van het pad. Ook *black holes* zijn veelvoorkomende fouten, waarbij datagrammen verloren gaan bij een specifieke router (hetzij door een foute configuratie, of onbeschibaarheid van de router in kwestie).

Vergelijk statische en dynamische routing, zonder in detail in te gaan op specifieke routingprotocollen.

Statische routing Hierbij staat de netwerkadministrator in voor het configureren van routers. Dit is een tijdrovend en foutgevoelig proces, dat onhoudbaar is bij complexere netwerktopologieën. Aangezien het `lifetime` veld ook meestal op oneindig wordt ingesteld, is het netwerk weinig foutbestendig. Het kent echter geen overhead, en de configuratie is relatief vanzelfsprekend.

Dynamische routing Hierbij zal het netwerk zichzelf configureren, en bovendien de configuratie aanpassen bij wijzigingen van de topologie. Hoewel dit een sterk voordeel is, kan dit leiden tot *route flapping*: het excessief uitwisselen van gegevens bij het uitvallen en snel terugkomen van een router. Ook is er bepaalde overhead aanwezig vanwege de routingprotocollen.

Geef de verschillende alternatieven om de routingtabel van niet-routers te configureren. Indien er hiertoe op Linux of Windows bijzondere componenten moeten geïnstalleerd of geconfigureerd worden, bespreek hoe dit moet gebeuren.

Routetabellen bij werkposten worden meestal statisch ingevuld met een default gateway. Op subnetwerken waar er meerdere routers aanwezig zijn is dit echter mogelijk suboptimaal. Daarom gaat men soms ook deze routetabellen dynamisch opstellen, wat men *host routing* noemt.

Dynamisch

Dynamische configuratie kan gerealiseerd worden op verschillende manieren:

- Router-discovery met ICMP: een werkpost kan, bij gebrek aan een default-gateway of specificatie via DHCP, routers op het subnetwerk detecteren door een *ICMP Router Sollicitation* bericht te sturen naar 224.0.0.2, waarop routers vervolgens antwoorden met een *ICMP Router Advertisement* bericht. Deze vorm van discovery is op Windows toestellen standaard ingeschakelt, maar Windows Server moet expliciet geconfigureerd worden om ook te reageren op dergelijke aanvragen.

- Routing-table optimalisatie: om de performantie te verhogen op subnetwerken waar meerdere routers aanwezig zijn, kan een router die berichten krijgt van een client die eigenlijk de router waarnaar het pakket doorgesluisd zal worden reeds rechtstreeks kan bereiken, een *ICMP Redirect* bericht sturen. Deze functionaliteit staat echter meestal uitgeschakeld, omdat ze toelaat een kwaadaardige gebruiker de configuratie in de war te sturen.

- Subnetwerk broadcasts: aangezien routers dynamische routing meestal verwezelijken via boardcast berichten op gezette tijdstippen, kunnen werkposten aan de hand van die uitwisselingen hun eigen routetabellen configureren of optimaliseren. Hiervoor luisteren ze naar relevante broadcasting berichten, zonder daarbij zelf aan interactie te doen. In geval van RIP berichten heet dit bijvoorbeeld *silent rip*. Op Linux toestellen kan men dergelijk gedrag bekomen via het commando `routed -d`, terwijl dit op Windows gerealiseerd wordt door de *RIP Listener* te installeren (dewelke zich bevindt in de *Network Services* categorie van de *Add/Remove Windows Components* wizard). Hierbij worden steeds enkel RIPv1 berichten geinterpreteerd.

Statisch

In geval van statische configuratie hangt het procédé sterker af van het besturingssysteem.

Linux Bij het configureren van een router moet eerst het forwarden ingeschakeld worden, wat op verschillende manieren kan:

- `echo 1 > /proc/sys/net/ipv4/ip_forward`
- `sysctl -w net.ipv4.ip_forward=1`
- `net.ipv4.ip_forward = 1` in `/etc/sysctl.conf`

Routes kunnen vervolgens bekeken worden met behulp van het `route` commando, dat de volgende parameters kent:

- `-n`: geef alle adressen in numerieke vorm, en probeer ze niet te resolven
- `-een`: geeft alle velden van de routetabel weer
- `add`: een route toevoegen, wat op zijn beurt nieuwe parameters vereist:
  - `-net` *adres/prefixlengte*: een subnetwerkspecificatie (`default` is een verkorte manier om `-net 0.0.0.0/0` te schrijven)
  - `dev`: netwerkdevice naar waar de pakketten moeten gestuurd worden
  - `gw` *adres*: beter alternatief voor `dev`, adres van de router naar waar de pakketten moeten gesturud worden
  - `reject`: een keyword dat expliciet uitdrukt dat de pakketten geweigerd moeten worden
- `del`: een route verwijderen, waarbij de parameters gelijk moeten zijn aan die gebruikt om de route in kwestie aan te maken (waarbij soms de `gw` specificatie mag weggelaten worden)

Net zoals bij `ifconfig`, is ook hier een recenter equivalent beschikbaar binnen het `ip` commando: `ip route`. Geen parameters aan dit commando meegeven zorgt voor een weergave van de routetabel, en andere parameters zijn ook mogelijk:

- `ip route add 0/0 via 172.16.33.17`: voegt een route toe
- `ip route change 0/0 via 192.168.16.89`: verandert een route
- `ip route flush root 0/0`: ledigt de routetabel
- `ip route get 193.190.126.71`: haalt de route-entry op die gebruikt zou worden om dit adres te bereiken

Windows Hier moet eveneens indien benodigd de router functionaliteit eerst expliciet ingeschakeld worden, wat op twee manieren kan gebeuren:

- Via de *Routing and Remote Access* service
- Via een register-wijziging: de waarde 1 geven aan de sleutel `IPEnableRouter` in `HKLM\System\CurrentControlSet\Services\TCPIP\Parameters`.

Dit kan vervolgens geverifiëerd worden via het `ipconfig /all` commando, meer bepaald op de lijn met het *IP Routing Enabled* veld.

Vervolgens is er een `route` commando aanwezig om de routingtabel weer te geven. Hiertoe dient `route print` of `netstat -r`. Toevoegen van een route wordt gedaan met het `route add` commando, met de volgende parameters:

- eerste parameter: de subnetspecificatie
- `mask` *adres*: het netmask adres (zonder dit wordt een netmask van 255.255.255.255 gebruikt, wat dus duidt op een *host route*)
- *router adres*: router naar waar de datagrammen moeten gestuurd worden
- `-p`: hiermee wordt de toegevoegde route persistent opgeslaan

De routetabel flushen kan met de `-f` parameter.

Bovenvermelde taken kunnen ook steeds uitgevoerd worden binnen de `netsh route` omgeving, of door gebruik de maken van de grafische configuratietool van de *Routing and Remote Access* service, *Routing and RAR Access* genaamd.

1.3 Dynamische routing (§1.5)

Bespreek het doel en de voordelen van dynamische routing. Behandel de terminologie en de problematiek die dynamische routing kenmerkt.

Bij dynamische routing gaat men de routingtabellen niet langer manueel invullen, maar laten genereren door een routingprotocol. Dit kent verschillende voordelen:

- Minder werk: het statisch configureren van routers is, zeker voor grotere netwerken, een enorm werk. Routing protocollen automatiseren dit.
- Minder foutgevoelig: statische configuratie elimineert de menselijke factor, waardoor zeker bij grotere netwerken de kans op routing-loops of black-holes gereduceerd wordt.
- Foutbestendiger: falen van routers wordt vlugger vastgesteld, waarna de routers zichzelf herconfigureren om terug naar een stabiele toestand te convergeren.
- Weinig onderhoud: eenmaal de initiële configuratie afgerond is, vereist een routingprotocol weinig onderhoud. Ook bij wijzigingen van de netwerktopologie is het niet langer nodig om alle routers te herconfigureren.

Terminologie

- Convergentie: het evolueren naar een stabiele toestand waarbij voor elke verbinding een optimaal pad beschikbaar is.

- Route flapping: het omslaan van de routingtabellen wegens het uitvallen van een verbinding. Hierbij bevindt het netwerk zich tijdelijk in een onstabiele toestand, waarbij routing-loops en black-holes mogelijk zijn.
- Convergentieperiode: het tijdsbestek nodig om het netwerk tot een stabiele toestand te laten convergeren.

Beschrijf het routing proces op Internet schaal (inclusief de relatie tot ISPs).

Omdat een routingprotocol loslaten op het gehele internet niet realistisch is, deelt men het internet op in verschillende *administratieve domeinen* (autonomous system, AS). Een dergelijk AS wordt meestal beheerd door een enkele organisatie, meestal een *Internet Service Provider* (ISP).

Om informatie over ASs heen uit te wisselen, worden steeds 1 of meer *AS Border Routers* (ASBR) aangeduid. Deze routers wisselen ook informatie uit met ASBRs van andere ASs. Deze informatie is echter zo weinig mogelijk geaggregeerd, wat bekomen wordt door zoveel mogelijk adresbereiken te aggregeren (wat enkel mogelijk is indien de hoogste-orde bits van de subnetten overeenkomen en niet voorkomen in subnetten van andere AS).

ISPs ontmoeten zo andere ISPs in *regional peering points* of *regional exchanges*, maar om de hoeveelheid dataoverdracht te reduceren kennen grote tier-1 ISPs ook *private peering points* om onderling informatie uit te wisselen.

Maak een classificatie van routingprotocollen, volgens twee criteria. Omschrijf de terminologie die je hierbij invoert. Geef van elke klasse de meest courante vertegenwoordigers. Het is niet de bedoeling in te gaan op een gedetailleerde vergelijking tussen de verschillende klassen en hun specifieke vertegenwoordigers.

Volgens domein

Men kan routing protocollen indelen volgens het domein waarin ze actief zijn.

Interior Gataway Protocol (IGP) Dergelijke routing protocollen zijn actief binnen eenzelfde AS. Hierbij wordt meestal de beste route berekend voor alle mogelijke eindbestemmingen, om die dan te propageren naar alle routers binnen het AS. Voorbeelden van dergelijke protocollen zijn: RIP, EIGRP, OSPF en IS-IS

Exterior Gateway Protocol (EGP) Deze routing protocollen staan in voor het uitwisselen van routing-info over de grenzen van een AS heen. Deze protocollen zijn dan ook enkel actief op *AS Border Routers* (ASBR). De informatie die daarbij doorgestuurd wordt zoveel mogelijk geaggregeerd, en een EGP kan meestal ook rekening houden met verschillende andere factoren (zoals de kost, of politieke factoren).

Volgens technologie

Routingprotocollen kunnen ook ingedeeld worden volgen technologie, zoals welke informatie uitgewisseld wordt en hoe die over het netwerk verdeeld wordt. Daarom zullen ze ook verschillen wat betreft convergentieperiode en netwerkbelasting.

Distance vector protocollen Bij dergelijke protocollen zal een router aan zijn buren een *distance vector* doorsturen, waarin staat welk netwerk aan welke kost bereikt kan worden via welke volgende hop, informatie die lijkt op de inhoud van een routingtabel. Aan de hand van die informatie kan elke router zo zijn eigen routingtabel bijwerken indien een netwerk bereikbaar wordt aan een lagere kost. Deze protocollen hebben als nadeel dat netwerkbelasting relatief groot is: hoewel de pakketten enkel uitgewisseld wordt met de directe buren, gebeurt dit periodiek en wordt steeds alle informatie doorgestuurd. Ook blijft dit periodiek zenden steeds actief, zelfs als het netwerk reeds geconvergeerd is.

Voorbeelden van dergelijke routing protocollen zijn: BGPv4, RIP en EIGRP.

Link state protocollen Hierbij zal een router alle andere routers in het netwerk enkel duidelijk maken welke buren bereikbaar zijn. Aan de hand van die connectiviteitsinformatie kan elke router dan een map opstellen van het netwerk, en daarmee optimale berekenen om de routingtabel op te vullen. Deze methode is een pak rekenintensiever, aangezien elke router nu individueel een map van het netwerk moet kunnen opstellen. Het protocol is echter beter bestand tegen wijzigingen van de topologie (geen count-to-infinity), en convergeert ook beduidend sneller. Ook is ligt de netwerkbelasting lager, aangezien het verzenden pas gebeurt indien noodzakelijk en daarbij enkel de wijzigingen doorstuurt.

Binnen deze categorie vindt men onder andere OSPF en IS-IS.

1.4 Routing Information Protocol (RIP) (§1.6, behalve §1.6.3 en §1.6.4)

Geef een gedetailleerde beschrijving van de werking van RIP. Bespreek de mogelijkheden, beperkingen, en problemen. Bespreek in het bijzonder de gehanteerde metriek, en hoe RIP berichten verpakt worden (cfr. het OSI 7-lagen model).

Werking RIP is een interior, distance vector routing protocol. Routers adverteren al hun *route-vectoren* via UDP-pakketten op een broadcast adres, wat andere routers op het subnetwerk toelaat hun routingtabel bij te werken. Die berichten worden periodiek, maar asynchroon (los van andere routers) verstuurd. Elke router slaat enkel het pad met de laagste metriek op (met de metriek zijnde de hop-afstand), en zal enkel een metriek verhogen als de eerste intermediaire hop dit meldt. RIP broadcast standaard om de 30 seconden, óók als het netwerk reeds geconvergeerd is tot een stabiele toestand.

RIP kent echter een aantal nadelen:

- Maximale netwerkdiameter: 15 hops (als alle verbindingen een metriek van 1 krijgen).
- Grote netwerkbelasting, die bovendien permanent aanwezig is.
- Trage convergentie.
- Count-to-Infinity probleem.
- Routing loops: steeds mogelijk tijdens de convergentieperiode.

Wat wordt bedoeld met reductie van de convergentieperiode (inclusief oorzaken)? Bespreek de verschillende technieken om dit te verwezenlijken.

RIP convergeert traag: het is niet in staat op wijzigingen van de topologie snel op te vangen. Hier zijn verschillende redenen voor:

- Paden met hogere metrieken worden genegeerd, enkel diegene met de laagste metriek wordt opgeslaan (hoewel de standaard voorschrijft dat alle paden zouden moeten opgeslaan worden, respecteren de meeste implementaties dit niet).
- Hoge lifetime parameter: standaard is deze 3 minuten, om potentieel packet-loss inherent aan UDP op te vangen.
- Periodieke advertering: wijzigingen van de topologie triggeren geen adverteringen, waardoor het lang duurt voor andere routers weet hebben van de wijziging.

Een klassiek probleem dat deze convergentieperiode nog kan verergeren, is het *count-to-infinity* probleem. Dit komt voor als een router, die merkt dat een verbinding onbeschikbaar geworden is en de metriek dus op oneindig zet, een advertering van een buur ontvangt vooraleer die wijziging kunnen doorgestuurd te hebben. Hierdoor wordt de juist ingestelde metriek vervangen door de vorige metriek + 1, met de adverterende router als volgende hop. Dit proces herhaalt zich bij elke advertentie, waardoor de metriek gestaag toeneemt. Om dit probleem ietwat te reduceren, wordt de maximale metriek vastgelegd op 16, waardoor dit probleem nooit meer dan enkele minuten in beslag zal nemen.

Om de convergentieperiode te reduceren, worden verschillende technieke toegepast:

- Split horizon: hierbij zullen routers niet langer adverteren langs de verbinding waar ze een bepaalde route vernomen hebben. Dit verlaagt de netwerkbelasting en elimineert het count-to-infinity probleem, tenzij er meerdere paden naar die buur bestaan.

- Poison reverse: deze aggresieve variant van split horizon adverteert ontvangen routes wel langs de verbinding van de zender, maar met de metriek ingesteld op de maximale metriek mogelijk. Dit reduceert de kans op count-to-infinity nog meer, maar ditmaal zonder de bijkomende verlaging van netwerkbelasting.

- Triggered updates: hierbij zullen routers wijzigingen doorsturen wanneer die gedetecteerd worden. Ook wanneer de lifetime van een route verloopt zal die geadverteerd worden met een maximale metriek. Als deze techniek consequent toegepast wordt bij alle routers van het netwerk, verkleint de convergentieperiode drastisch. Een nadeel is echter dat het de netwerkbelasting merkaar verhoogt, desondanks de kleine delay die triggered advertenties voorafgaan om een cascade aan berichten te vermijden. In geval van een lifetime die verloopt, worden triggered updates lichtjes anders toegepast: de route wordt bijna onmiddellijk gebroadcast met een maximale metriek. Hiermee wordt de kans op het count-to-infinity probleem opnieuw sterk verkleind.

- General RIP request: opstartende routers kunnen een dergelijke request sturen, waarop ondersteunende routers zullen reageren met een advertisement van hun gehele routing tabel. Hierdoor zal het netwerk snelle convergeren bij introductie van een nieuwe router.

Bespreek de verschillende verbeteringen van RIPv2 ten opzichte van RIPv1.

- Multicasting: RIP-berichten worden niet langer gebroadcast (wat enkel *silent RIP* als voordeel had), maar gemulticast op adres 224.0.0.9.

- Subnetmasker binnen route-vectoren: RIPv1 ging er van uit dat de prefixlengte af te leiden was uit het IP adres, maar sinds de invoering van CIDR mag dit niet meer verondersteld worden.

- Next-hop veld: een RIPv2 bericht stuurt nu ook het adres van de eerste intermediaire hop mee (0.0.0.0 indien rechtstreeks te bereiken). Hierdoor kan een router bij ontvangst van een route-vector eerst kijken of de hop in kwestie niet rechtstreeks te bereiken is, en zo een *dubbele hop* vermijden. Dit is te vergelijken met de *ICMP Redirect* techniek.

- Eenvoudige authentificatie: om te vermijden dat het netwerk kwaadwillig in de war gestuurt wordt, kunnen authentificatiegegevens binnen het RIP bericht vereist worden.

1.5 Open Shortest Path First protocol (OSPF) (§1.8, enkel subsectie §1.8.1 inclusief)

Geef een gedetailleerde beschrijving hoe OSPF werkt, inclusief de diverse mechanismen van berichtenuitwisseling en de bijzondere functie die OSPF routers kunnen hebben, zonder daarbij in te gaan op de uitwerking van het algoritme van Dijkstra en het concept van OSPF area's.

OSPF is een interior, link state routing protocol en stuurt dus informatie over de bereikbare buren (*link state advertisements*) naar alle routers binnen het netwerk. Die berichten worden ook enkel gestuurd bij het opstarten of na het detecteren van een wijziging binnen de topologie, en moeten bevestigd worden door te ontvanger.

Autonomous System Border Routers (ASBRs) die toegang hebben tot routers die niet tot het AS behoren, adverteren die routes in *external route LSAs*, die meestal geaggregeerd worden tot één enkele *default route LSA*.

Alle routers compileren vervolgens apart een *link state database* (LSDB) als weergave van alle toegankelijke subnetwerken, waarin elke LSA een *lolly number* bevat om onderscheiden te kunnen worden van nieuwere gegevens. De LSDB bevat 1 LSA per router, en zal daarom ter identificatie elke router een *router ID* toekennen. [1]

Met behulp van het algoritme van Dijksta kan vervolgens het pad met de laagste metriek berekend worden voor elk subnetwerk, waarbij zo niet alleen de eerstvolgende hop maar het volledige pad bekomen wordt. Met deze paden wordt vervolgens een *shortest path first* (SPF) tree opgebouwd, waaruit tenslotte de routingtabel kan afgeleid worden.

Link state database synchronisatie LSBDs worden gesynchroniseerd via flooding van LSAs: een router zal ook de LSA die hij van een rechtstreekse buur op een interface ontvangen heeft, versturen langs de andere interfaces. Om convergentie verder te versnellen vormen routers die van elkaar bewust zijn en LSDBs gesynchroniseerd hebben een *adjacency*. Hierdoor moet een router bij veel minder routers verifiëren of de LSBD wel degelijk gesynchroniseerd is. Om adjacencies te vormen, verstuurt een router *Hello-pakketten*, de het ID van de router bevat alsook de IDs van alle naburige routers waarvan de router reeds een Hello-pakket ontvangen heeft. Hierbij worden alle routers gedetecteerd binnen een Hello-pakket van een andere router beschouwd als rechtstreeks bereikbaar, waardoor het detectieproces opnieuw versneld wordt.

Nu de routers binnen de adjacency gedetecteerd zijn, kan begonnen worden aan het *database exchange process*. Hierbij wisselt elk koppel routers binnen de adjacency *Database Description* pakketten uit, die de LSAs en hun lolly number binnen de LSDB beschrijven. Zo kan een router detecteren of zijn LSBD verouderde informatie bevat, en die eventueel bijwerken met behulp van een *Link State Request* pakket, die beantwoord worden met een *Link State Update* pakket. Hierna bevat elke router in de adjacency

[1] Deze ID, uniek in het hele AS, identificeerd een router (en niet 1 van zijn interfaces!).

up-to-date informatie, en kan de LSBD als gecompileerd beschouwd worden. Na dit proces wordt normaliter geen informatie meer uitgewisseld tussen de routers.

Periodiek worden echter toch nog Hello-pakketten uitgewisseld om duidelijk te maken dat de router nog steeds actief is. In geval van een wijziging van de topologie, zal een router Link State Update pakketten sturen, die ervoor zorgen dat de LSDBs opnieuw gesynchroniseerd worden. Deze pakketten worden, conform het flooding principe, doorgestuurd naar alle andere buren. Hierbij worden echter, in tegenstelling tot distance-vector protocollen, enkel de wijzigingen doorgestuurd.

Bij het oprichten van adjacencies wordt echter een optimalisatie doorgevoerd, aangezien er in de huidige opzet veel te veel adjacencies gevormd worden. Om dit te vermijden, wordt binnen elk subnetwerk een *Designated Router* (DR, een eigenschap van een routerinterface) verkozen, en moet elke router enkel een adjacency vormen met de DR. Aangezien de DR aangrenzend is aan elke router van het subnetwerk, zal een wijziging in het netwerk enkel moeten doorgegeven worden aan de DR (die luistert op multicast adres 224.0.0.6) die op zijn beurt zal zorgen dat de wijziging gepropageerd wordt naar alle andere routers binnen het subnetwerk. Hierdoor neemt de netwerkbelasting slechts lineair toe met het aantal routers, en niet langer kwadratisch. Om te vermijden dat het netwerk in de war gestuurd wordt bij uitvallen van de DR, wordt steeds een *Backup Designated Router* (BDR) aangeduidt die enkel luistert op bovenstaand multicast-adres (en zo alle relevante informatie bevat), om zo snel de taak van de DR te kunnen overnemen.

De DR en BDR worden verkozen tijdens het uitwisselen van Hello-pakketen op basis van de *router prioriteit*.

Beschrijf, o.a. aan de hand van een figuur, wat er precies gebeurt indien er een nieuwe router in een door OSPF gestuurd internetwerk wordt opgenomen.

Na plaatsing van een nieuwe router zal die Hello-pakketten beginnen sturen naar de rechtstreeks bereikbare routers. Die routers zullen zo besluiten een adjacency te vormen, waarna Database Description en eventuele Link State Request/Update pakketten zullen uitgewisseld worden. Zo wordt de adjacency, bestaande uit de nieuw geintroduceerde router en zijn rechtstreekse buren, volledig gesynchroniseerd.

Na berekening van een nieuwe SPF-tree zullen de routers binnen de nieuwe adjacency Link State Update pakketten sturen naar alle andere routers waarmee die een adjacency vormen, waardoor ook hun LSDB opnieuw gesynchroniseerd wordt. Hierbij zal gebruikt gemaakt worden van flooding om snel tot een geconvergeerde toestand te komen. Hierbij wordt enkel een LSA van de nieuwe router verspreid, wat wederom de convergentietijd verkleint.

Hoe worden OSPF berichten verpakt (cfr. het OSI 7-lagen model)?

OSPF pakketten worden rechtstreeks ingekapseld in IP-datagrammen. Ook wordt er, in tegenstelling tot RIP, gewacht op ontvangstbevestiging. Blijft die uit, dan wordt het pakket opnieuw verzonden. Meermaals uitblijven van een antwoord wordt geinterpreteerd

als het falen van een verbinding.

Wat is TOS routing, en in hoeverre ondersteunt OSPF dit?

OSPF ondersteunt *Type of Service* (TOS), waarbij de LSA voor elke TOS een aparte metriek kan bevatten. Een router die geen TOS ondersteund, zal enkel een metriek hebben voor een TOS die gelijk is aan 0 (een dergelijke waarde moet altijd vermeld zijn). Bij het genereren van de routingtabel zal vervolgens een SPF tree gegenereerd worden voor elke TOS waarde. Hierbij moet opgemerkt worden dat bij berekening van een SPF tree met een TOS waarde verschillend van 0, er geen gebruik mag gemaakt worden van verbindingen die geen metriek hebben voor die TOS waarde! Zo kan het zijn dat gebruik makende van bepaalde metrieken, delen van het netwerk onbereikbaar zijn!

1.6 RIP, OSPF en het Enhanced Interior Gateway Routing Protocol (EIGRP) (§1.7 en §1.8.4)

Van welke categorieën routingprotocollen zijn RIP en OSPF typische vertegenwoordigers?

RIP is een *distance vector protocol*, aangezien het steeds de volledige routing tabel doorstuurt naar zijn rechtstreekse buren.

OSPF is een *link state protocol*, omdat het eenmalig naar alle routers in het hele subnetwerk LSAs doorstuurt die elke router toelaten een LSDB op te stellen die het hele netwerk voorstelt.

Beide zijn *interior gateway protocollen*, daar ze gebruikt worden binnen een enkel AS.

Maak een gedetailleerde vergelijking tussen de mogelijkheden en beperkingen van deze twee categorieën.

Distance-vector protocollen

Voordelen

- Eenvoudig mechanisme: 1 type bericht, 1 tabel, eenvoudig adverteren en updaten van routingtabellen.
- Eenvoudig te configureren: meestal gewoon service starten.
- Geen complexe verwerking van advertisements: nagenoeg zonder berekening opnemen in routingtabel.
- Goede resultaten (enkel bij eenvoudige topologieën en stabiele verbindingen).

Nadelen

- Grote routingtabellen: verschillende mogelijke paden moeten in routingtabel opgenomen worden, onbruikbaar voor grote netwerken.
- Maximale diameter van 15 hops.
- Produceren veel netwerkverkeer: zelf na convergentie blijven routes periodiek geadverteerd worden.
- Lange convergentieperioden: bij gebrek aan synchronisatie of bevestiging, en dankzij problemen zoals *count-to-infinity* en mogelijke *routing loops* en *black holes*.

Link-state protocollen

Voordelen

- Kleine routingtabellen: slechts 1 enkele route voor elke eindbestemming.
- Lagere netwerkbelasting: wegens bijhouden van complete kaart van het netwerk hoeft men immers enkel wijzigingen door te sturen. Ook na het convergeren verbruiken link-state protocollen veel minder, daar ze enkel heel compacte Hellopakketten periodiek uitwisselen. Toch veroorzaakt een *route flap* relatief veel verkeer, namelijk tweemaal het *flooden* van een LSA naar elke router binnen het netwerk.
- Toepasbaar op grote routingdomeinen: OSPF kent geen bovengrens, zeker al niet omdat het netwerk in verschillende *OSPF area's* kan opgedeeld worden.
- Kortere convergentieperioden: nagenoeg onmiddellijk. Ook zijn *routing loops* en *count-to-infinity* problemen steeds onmogelijk aangezien elke router over dezelfde LSDB beschikt.
- Elke router bezit overzicht van het hele subnetwerk.

Nadelen

- Complex: 5 soorten berichten en 3 verschillende procedures (Hello, Database Exchange, Flooding).
- Configureren van area's vereist planning en configuratie.
- Geheugen- en processorintensief: berekenen van routingtabel uit de link state database is niet eenvoudig. Zeker wanneer de router een specifieke OSPF rol vervult (DR, BDR, ABR) wordt nog meer inspanning vereist.

Tot welke categorie behoort EIGRP? Bespreek de meerwaarde van dit protocol in vergelijking tot de andere vertegenwoordiger in deze klasse, en (in detail) hoe dit gerealiseerd wordt.

EIGRP is een distance vector protocol dat periodiek de routetabel multicast naar al zijn buren. Het is een proprietaire verbetering van RIPv1, dat tevens *split horizon* en *triggered updates* toepast om de kans op lussen te verkleinen. Omdat het inherent verbonden is aan Cisco wordt veeleer gekozen voor open alternatieven zoals OSPF. Technisch kent het protocol echter enkele verbeteringen ten opzichte van RIPv1, zijn tegenhanger binnen de distance vector protocollen.

Meerdere metrieken EIGRP kent 4 metrieken:

- Statische metrieken: afgeleid uit het type subnetwerk, of ingesteld door de
  - Delay: de som van latenties tussen de router en de bestemming, uitgedrukt in eenheden van 10 μs.
  - Bandwidth: de bandbreedte van de zwakste verbinding, uitgedrukt in eenheden van 10 μs benodigd om 10 Kb te transporteren.
- Dynamische metrieken: verkreden door het verkeer in de verbindingen statistisch te analyseren.
  - Reliability: percentage pakketverlies.
  - Load: bezettingsgraad van de drukste verinding.

De samengestelde metriek wordt berekend op basis van deze 4 waarden, waarbij het gewicht van elke parameter ingesteld wordt door de netwerkbeheerder.

Diffusing Update Algorithm (DUAL) Deze techniek vermijdt routing loops.

Hoofdstuk 2

Reeks B

2.1 Dynamic Host Configuration Protocol (DHCP) (§2.1 en §2.1.1)

Geef een overzicht van de belangrijkste DHCP concepten en terminologie.

DHCP is een techniek om de TCP/IP configuratie van clients zoveel mogelijk te automatiseren en centraliseren. Het werkt volgens het client-server model: een zal client bij het opstarten een DHCP-aanvraag broadcasten, waarop een DHCP-server zal antwoorden met de gepaste configuratieinformatie. Deze vraag wordt, bij gebrek aan netwerkconfiguratie, enkel gebroadcast op het lokale subnetwerk.

Een DHCP server bevat zowel statische adressen (voor specifiek geregistreerde computers) als dynamische adressen (voor niet-geregistreerde computers), die komen uit een bepaalde bereik, de *DHCP-scope*. Dit onderscheid kan nuttig zijn indien bepaalde computers (zoals servers) een vast adres moeten toegewezen worden.

Elk uitgedeeld adres kent een bepaalde geldigheidsduur, de *lease*. Verloopt die, dan zal de client de server moeten contacteren om de lease te verlengen of een nieuw adres aan te vragen. De server onthoudt hierbij welke adressen reeds bezet zijn, en welke niet, zodat het mogelijk is om meer computers (weliswaar niet tegelijk) te gebruiken dan er IPs beschikbaar zijn op een netwerk.

Waarin verschilt DHCP van het Bootstrap Protocol (BOOTP)? Beschrijf de structuur van DHCP berichten. Je hoeft de verschillende types berichten niet te vermelden.

BOOTP is een minder recente methode die gebruikt werd om de TCP/IP configuratie van clients te centraliseren, maar is minder flexibel dan DHCP. Zo vereist BOOTP dat men een lijst van MAC-adressen van alle clients beschikt, en kent het geen mogelijkheid om tijdelijk IP-adressen ter beschikking te stellen (DHCP kan dit wel met behulp van de DHCP scope). DHCP ondersteunt voorzieningen om maximale compatibiliteit met

BOOTP te garanderen.

Een DHCP bericht is steeds samengesteld uit twee delen. Eerst is er de *DHCP header*, een gedeelte met constante grootte van exact 300 bytes, dat de essentiële delen van de configuratie bevat:

- Transactie ID.
- IP en MAC van de client.
- IP en naam ven de server.

Deze formattering is tot nu toe identiek aan het BOOTP protocol.

Hierna komen de *DHCP opties*, TLV geëncodeerd in een deel van het pakket met variabele grootte. De opties worden voorafgegaan door een *magic cookie*, bestaande uit 4 bytes 99.130.83.99, dat het bericht onderscheid van een BOOTP bericht (dat geen opties kent). Elke optie bestaat uit drie velden:

- Identificatie van de optie (1 byte).
- Grootte van het volgende veld (1 byte).
- Effectieve waarde van de optie.

Het einde van het opties-gebied wordt aangeduid door *optie 255*: een optie die enkel een identificatie-veld bevat. Alles na deze optie wordt genegeerd. Tenslotte bestaat er ook nog een *optie 0*, opnieuw enkel met identificatie-veld, dat gebruikt wordt om opties af te lijnen op woordgrenzen.

Samengevat kunnen we zeggen dat DHCP de BOOTP technologie uitbreidt door er een dynamische scope aan toe te voegen, en aan het pakket variabele opties toe te voegen.

Waarom voert DHCP opties in?

Opties worden gebruikt om naast de standaard TCP/IP configuratie, ook andere informatie door te sturen, zoals een in te stellen hostnaam of een routingtabel. Deze opties worden niet noodzakelijk door alle client-implemenaties ondersteund, en kunnen dus soms gewoon genegeerd worden.

Ook zorgen opties ervoor dat er verschillende DHCP berichten kunnen onderscheiden worden. Wegens plaatsgebrek kan dit immers niet in de BOOTP header geplaatst worden, en zal men deze differentiatie wegschrijven in optie 53.

Geef een overzicht van de courant gebruikte DHCP opties.

Tag	Beschrijving
1	Subnetmasket van het clientsubnet.
3	Lijst met IP-adressen van de default gateways (in volgorde te gebruiken).
6	Lijst met IP-adressen van DNS nameservers (in volgorde te gebruiken).
12	Hostnaam (maximumlengte van 63 tekens).
15	DNS suffix, te gebruiken bij het herleiden van niet gekwalificeerde DNS namen.
19	Bepaalt of de client de functie van router moet vervullen.
23	Standaard TTL-waarde voor uitgaande datagrammen.
28	Te gebruiken adres voor broadcasts (normaal 255.255.255.255).
31	Geeft aan of de client routers moet aanvragen via ICMP router discovery.
33	Lijst met IP-paren te gebruiken als routing informatie (subnet adres en router adres).
35	Time-out waarde voor vermeldingen in de ARP cache.
69	Lijst met IP-adressen van SMTP servers (geen volgorde opgelegd).
70	Lijst met IP-adressen van POP3 servers (geen volgorde opgelegd).

Beschrijf wat er gebeurt op DHCP niveau bij het heropstarten van een Windows Server toestel, nadat men een shutdown heeft uitgevoerd.

Windows DHCP-clients bewaren de DHCP-configuratie lokaal op de harde schijf (in de register *hive*). Hierdoor zal bij het heropstarten geprobeerd worden om de oude lease te verlengen. Als de server die de oude lease uitgedeeld heeft niet beschikbaar is, dan probeert de client te detecteren of hij nog steeds in hetzelfde subnet aanwezig is (door een ping uit te voeren naar zijn default gateway). Indien dit zo is, zal de oude lease stilzwijgend verder gebruikt worden. Is dit niet het geval, zal de client een nieuwe lease aanvragen bij de lokaal aanwezige DHCP server.

Beschrijf wat er gebeurt indien een Windows Server DHCP-cliënt geen DHCP-server kan bereiken.

De client zal dan blijven een DHCP-Discovery sturen, maar de wachtperiode tussen opeenvolgende berichten stijgt exponentieel (machten van 2). Als na 4 pogingen (16 seconden wachttijd) nog steeds geen server geantwoord heeft, zal het proces stilgelegd worden en periodiek om de 5 minuten herhaald worden.

In die tussenperiode waarbij geen configuratie gevonden is, zal de client gebruik maken van een tijdelijke IP-configuratie: Automatic Private IP Addressing (APIPA). Hierbij worden klasse-B adressen gebruikt uit het 169.254/16 bereik, die niet gebruikt worden op het internet. Deze toewijzingen zijn transparant voor de gebruiker, waardoor dit heel nuttig is voor clients in kleine netwerken waar geen DHCP server aanwezig is.

2.2 Leaseprocessen en relay-agents (§2.1.2 en §2.1.3)

Geef een overzicht van de verschillende types DHCP berichten.

Het type van de verschillende DHCP-berichten wordt wegens plaatsgebrek verpakt als waarde van optie 53 binnen het variabele gedeelte van een DHCP-pakket. De verschillende types zijn:

- DHCP-Discovery: dit bericht, dat steeds gebroadcast wordt, wordt gebruikt om servers te ontdekken. Het bevat ook de opties die men wil aanvragen: optie 51 wordt gebruikt om een specifieke lease-tijd aan te vragen, terwijl optie 55 een geordende lijst van gewenste opties bevat. Het adres van de client wordt binnen dit bericht als 0.0.0.0 ingesteld.
- DHCP-Offer: dit bericht wordt als reactie op een DHCP-discovery gestuurd, en biedt een client een IP-adreslease aan alsook de waarde van de aangevraagde opties (eventuele vergeten opties kunnen later met een DHCP-Inform opgevraagd worden). Optie 54 wordt ingesteld met de serveridentificatie (IP-adres).
- DHCP-Request: dit bericht wordt door clients gebruikt om een verkregen DHCP-offer te finaliseren. Optie 54 vermeldt hierbij de server identificatie.
- DHCP-Acknowledgment: hiermee bevestigt de server een DHCP-request van een client, en stuurt nogmaals de vereiste DHCP-opties mee. Dit bericht wordt eveneens gebroadcast, zodat andere DHCP-servers een tijdelijk gereserveerd adres weer kunnen vrijgeven.
- DHCP-Decline: dit bericht kan door de client naar de server gestuurd worden indien de ontvangen TCP/IP configuratieinformatie foutief blijkt te zijn. Het initialisatieproces wordt dan herstart.
- DHCP-Release: dit bericht zal een client sturen indien hij de ontvangen lease niet meer nodig heeft.
- DHCP-Inform: dit bericht kan gebruikt worden om aan de server aanvullende informatie te vragen (zoals extra opties).

Bespreek de opeenvolgende stappen van beide DHCP leaseprocessen.

DHCP kent twee leaseprocessen: de *initialisatie* voor wanneer een client voor het eerst gestart wordt, en de *vernieuwing* voor wanneer een client een reeds ontvangen lease wil vernieuwen.

Initialisatie

1. Client broadcast een DHCP-Discovery bericht (met bronadres 0.0.0.0 en doeladres 255.255.255.255, opties 51 en 55 ingesteld).

2. Servers antwoorden met DHCP-Offer (server ID in optie 54).
3. Client selecteert een lease, en vraagt die formeel aan met een DHCP-Request (met de gekozen server zijn ID in optie 54).
4. Server bevestigt de lease door een DHCP-Acknowledgment te broadcasten, of weigert ze met een negatieve DHCP-Acknowledgement. Dit wordt gebroadcast zodat andere servers tijdelijk gereserveerde leases opnieuw kunnen vrijgeven.
5. Client vraagt eventueel extra informatie met een DHCP-Inform.
6. Client neemt de lease in gebruik, of weigert ze met een DHCP-Decline.
7. Client geeft de lease vrij met een DHCP-Release.

Vernieuwing Dit proces wordt ondernomen wanneer een lease een tijdlang in gebruik geweest is, meestal nadat de helft van de lease-tijd verlopen is (tenzij anders ingesteld via optie 58).

1. Client stuurt een DHCP-Request naar de server die de lease gestuurd heeft.
2. Server antwoordt met een DHCP-Acknowledgement (met opnieuw de opties ingesteld zodat de clientconfiguratie correct kan bijgewerkt worden).

Als de client geen DHCP-Acknowledgement krijgt, wordt dit procédé herhaald met identieke wachttijden als bij het initialisatieproces (machten van 2, na 4 pogingen 5 minuten wachten).

Indien de client geen DHCP-Acknowledgement gekregen heeft en de lease dreigt te vervallen, meestal nadat $7/8^e$ van de lease-tijd verlopen is (tenzij anders ingesteld via optie 59), wordt de *rebinding status* geactiveerd:

1. Client probeert de lease te vernieuwen bij een willekeurige server door DHCP-Request berichten te broadcasten.
2. Server antwoordt met een DHCP-Acknowledgement, de client kan nu weer gerust zijn lease verder gebruiken.

Indien geen DHCP-Acknowledgement ontvangen wordt, zal de client dit bericht tot driemaal toe opnieuw verzenden (opnieuw met de gekende wachttijden).

Als de client hierna nog steeds geen DHCP-Acknowledgement ontvangen heeft (of als 1 van de andere servers gereageerd heeft met een negatief DHCP-Acknowledgement bericht), moet de client direct stoppen met het gebruiken van de lease. Hierna wordt de initialisatiefase aangevat om een nieuwe een lease te bekomen.

Bespreek doel en werking van DHCP relay-agents. Welke velden in DHCP berichten helpen deze functie realiseren?

Doel Aangezien DHCP gebruik maakt van broadcasting, functioneert het maar op een enkel subnetwerk. Om DHCP te gebruiken over verschillende subnetwerken, zijn er dus verschillende servers nodig. Als alternatief kan men relay-agents gebruikt, die DHCP berichten doorsturen naar andere subnetwerken.

Opmerking: aangezien DHCP berichten identiek geformatteerd zijn zoals BOOTP berichten volstaat een BOOTP relay-agent, een functionaliteit die vaak terug te vinden is op routers.

Werking Relay-agents kunnen berichten op twee manieren doorgeven: ofwel sturen ze ontvangen DHCP/BOOTP berichten *gericht* door naar de actieve server, of zullen ze het bericht opnieuw broadcasten op alle externe subnetten waarop ze aangesloten zijn.

Om deze functionaliteit te behelpen, worden enkele speciale opties binnen DHCP-berichten geïntroduceerd:

- Hop veld: dit is een teller die met 1 verhoogd wordt telkens een relay-agent het bericht opnieuw broadcast.
- Gateway veld: hierin wordt het IP van de doorsturende router opgeslaan. Wanneer een server een DHCP-Discovery bericht ontvangt, kan die kijken naar het gateway-veld om eventueel adressen uit een andere scope te gebruiken.

Het leaseproces ziet er bijgevolg iets anders uit:

- Client stuurt een DHCP-Discovery bericht met het gateway-veld op 0.0.0.0.
- Relay-agent stuurt het DHCP-Discovery bericht door, en vult het gateway-veld op met het IP-adres van de router (mogelijk zijn eigen IP-adres) indien het veld nog op 0.0.0.0 stond.
- Server antwoordt met een DHCP-Offer, waarbij het adres eventueel uit een andere scope gehaald is afhankelijk van het gateway-veld. Dit bericht wordt rechtstreeks naar de relay-agent gestuurd.
- Relay-agent geeft de adreslease door aan de client, door het te broadcasten in het relevante subnetwerk.

Indien je over een aantal DHCP servers beschikt, hoe kun je deze best configureren om een internetwerk, bestaand uit een aantal netwerken, te ondersteunen?

Om fouttolerantie te verhogen worden de DHCP-servers best in verschillende subnetten geplaatst. Bij gebrek aan communicatiemethode tussen de DHCP-servers mogen de scopes echter niet overlappen! Om toch fouttolerantie te bekomen wordt in elk subnetwerk een relay-agent geplaatst die DHCP-berichten kan doorsturen naar een extern subnet

indien de lokale server offline blijkt te zijn. Hiervoor moet elke server wel een adresbereik hebben voor elk subnetwerk. Een goede maatstaaf hierbij is dat 80% van de scope beheerd wordt door een DHCP server lokaal aan het subnetwerk, en 20% uitgespendeerd is aan een externe DHCP server zodat clients nog steeds hun interfaces kunnen configureren als de lokale DHCP server offline is.

Hoe kunnen Linux en Windows Server toestellen als DHCP relay-agent worden geconfigureerd?

Linux Op dergelijke toestellen kan men het commando `dhcrelay` gebruiken, met als argumenten de IP-adressen van 1 of meerdere servers aan dewelke DHCP-berichten moeten doorgesturud worden.

Windows Hierbij is de relaying-functionaliteit verpakt in het *DHCP Relay Agent* routing-protocol, dat analoog aan RIP en OSPF moet toegevoegd worden met behulp van de *Routing and Remote Addressing Management Console* via het commando `rrasmgmt.msc`. Hierbij moet steeds minstens 1 interface toegevoegd worden via de controlestructuur, en moet het *General* tabblad correct ingevulg worden (maximum waarde van het hop-veld, vertragingsinterval). De locaties van externe servers kunnen vervolgens ingevuld worden door rechts te klikken op de *DHCP Relay Agent* controlestructuur, te kiezen voor het *Properties* tabblad, en ze vervolgens eveneens in het *General* tabblad in te vullen.

2.3 Configuratie van Domain Name System (DNS) servers onder Linux

De figuur in bijlage stelt een intranet bestaand uit een aantal Linux computers voor, met corresponderend IP-adres, van de vorm 192.168.16.z . Het getal z lees je af links van de naam van de computer. De getallen rechts van de naam van de computer moet je negeren. De computers staan gegroepeerd in een tabel met als header de naam van het domein waarin ze zich bevinden. De rechthoeken die domeinen groeperen stellen dan weer een zone voor. Stippellijnen duiden op een domein/sub domein relatie. De pijlen laten toe om de primaire name server van elke zone te achterhalen. Je hoeft geen reverse DNS te configureren.

Bespreek in detail het begrip secundaire nameserver, inclusief voordelen, beperkingen en problemen (§3.1 en §3.3.3).

Secundaire nameservers houden exact dezelfde informatie bij als de primaire nameservers, en worden gebruikt enerzijds om de taak van de primaire nameserver te verlichten, en anderzijds om te zorgen voor fouttolerantie in het geval dat de primaire nameserver zou uitvallen.

Een secundaire nameserver is echter niet even flexibel in gebruik: zo kunnen wijzigingen van zone-gegevens enkel toegepast worden bij de primaire nameserver. Om de wijzigingen eveneens te ontvangen zal de secundaire nameserver dus geregeld contact moeten opnemen met de primaire nameserver, om een kopie van de nieuwe data te ontvangen.

Om gegevensoverdracht toe te laten tussen de primaire en de secundaire nameserver, zal de primaire nameserver als volgt moeten geconfigureerd worden:

```
allow-transfer {
  192.168.0.2;
};
```

Bij het configureren van de secundaire zal men zijn type op "slave" moeten zetten, en de juiste directives voorzien zodat de service weet waar hij zijn gegevens moet halen. Het bestand dat als zone-file gespecifieerd wordt zal dan gebruikt worden om de ontvangen gegevens in weg te schrijven, waardoor de secundaire nameserver snel als primaire nameserver kan geherconfigureerd worden moest dit nodig zijn.

```
zone "example.com" {
  type slave;
  masters {
    192.168.0.1;
  };
  file "db.example.com";
}
```

2.4 Configuratie van DNS servers onder Linux

De figuur in bijlage stelt een intranet bestaand uit een aantal Linux ...(cfr.vraag B3)... achterhalen. Geen enkele zone heeft een secundaire nameserver. Je hoeft geen reverse DNS te configureren.

Stel het configuratiebestand en alle zonebestanden op van volgende DNS servers. Gebruik relatieve DNS namen waar mogelijk. Gebruik noch forwarders, noch de $ORIGIN opdracht!

Bespreek in detail het formaat van een zonebestand en zijn records. Je mag dit doen op basis van één van de oplossingen in a), maar je moet ook alternatieve records en formaten beschrijven, die je niet noodzakelijk hebt gebruikt (§3.3.1 en §3.3.4).

Een zonebestand bevat de naamgegevens voor een enkele zone. Daarbij correspondeert doorgaans elke lijn met een afzonderlijk DNS-record. Meerdere lijnen kunnen daarbij gegroepeerd worden tot 1 record door gebruik te maken van ronse haken.

Elk DNS-record bestaat uit 5 vaste velden: `naam ttl klasse recordtype waarde`.

- Naam: de naam van een machine of domein (in geval van een spatie of tab wordt de naam van het vorige record gebruikt). Namen kunnen op verschillende manieren genoteerd worden:
  - Niet-gekwalificeerde alfanumerieke naam: een dergelijke naam is relatief ten opzichte van het domein waarvoor dit een zonebestand is.
  - Absolute domeinnaam: deze eindigt op een punt.
  - Speciale naam "@": deze naam verwijst naar de huidige oorsprong, en wordt doorgaans enkel gebruikt bij het SOA record bovenaan het bestand. De oorsprong kan eventueel gewijzigd worden door gebruik te maken van de $ORIGIN opdracht.
- Time-to-live: dit veld (dat eventueel mag weggelaten worden) geeft aan hoelang het DNS-record geldig is, en hoelang het dus door een andere computer mag gedached worden. Dit wordt weinig gebruikt aangezien het SOA record een default TTL waarde kent.
- Klasse veld: dit veld definieert het protocol waarvoor het record geld, wat heden ten dage steed "IN" zal zijn (wat staat voor het internet). [1]
- Recordtype: dit veld bepaalt wat voor type het record is, met mogelijke waarden: A, AAAA, PTR, MX, SOA, NS en CNAME.
  - A: een regulier IPv4 adres. Indien een bepaalde DNS-naam gebonden is aan verschillende IP-adressen zal cyclisch gebruik gemaakt worden van de opgegeven informatie. [2]
  - AAAA: een IPv6 adres.
  - PTR: een pointer record, dat verwijst naar een andere hostnaam. Dit wordt gebruikt bij *reverse DNS resolving*.
  - MX: een mailserver record, dat wijst naar de mailserver die gecontacteerd moet worden bij het afleveren van mail aan deze zone. Bij meerdere MX records moet ook voorzien worden in een prioriteitsveld, waarbij vervolgens steeds de record met de laagste prioriteit effectief gebruikt zal worden.
  - SOA: een Start of Authority record, dat de zone definieert.
  - NS: een nameserver record, dat definieert welke nameserver verantwoordelijk is voor deze zone.
  - CNAME: een canonical name record, wat gebruikt wordt om een alias op te geven voor een bepaalde DNS-naam. Daarbij mag de gekozen alias enkel links van een CNAME-definitie staan, en niet aan de linkerkant van een andere definitie! Het is dan ook aangeraden om CNAME-records zoveel mogelijk te vermijden, en A-records te gebruiken.

[1] Andere mogelijke waarden zijn "HS" en "CH", beide historische protocollen ontwikkeld aan het MIT.

[2] Dit is eigenlijk een primitieve vorm van load-balancing.

- Waarde: het laatste veld tenslotte bepaalt de effectieve waarde van het record, in geval van een A-type record zal dit het IP-adres zijn dat overeen komt met de opgegeven naam.

Het eerste record in een zonebestand is het *Start of Authority* (SOA) record: dit geeft het begin aan van de records van een bepaalde zone. Er kan dus ook maar 1 SOA record zijn als we per zone 1 zonebestand gebruiken.

```
@IN SOA primaire_dns email (
  volgnummer
  refresh interval
  retry interval
  expire interval
  default ttl
);
```

Het SOA veld kent verschillende parameters die de zone configureren:

- Primaire DNS: een absolute naam die verwijst naar de primaire DNS server verantwoordelijk voor deze zone.
- Email: het contactadres van de persoon verantwoordelijk voor de server (waarbij "@" vervangen is door een regulier punt).
- Volgnummer: een nummer dat door secundaire DNS-servers gebruikt wordt om na te gaan of de informatie wel nog recent is. Bij elke wijziging moet dit nummer dus verhoogd worden. Hierbij kan de conventie gerespecteerd worden, die stelt dat de eerste 8 cijfers gevormd worden door het jaartal, de maand en de dag, terwijl de laatste 2 cijfers incrementeel de wijziging indiceren.
- 4 tijdsduren:
  - Refresh interval: geeft aan hoe vaak de secundaire nameserver de database van deze zone moet veranderen.
  - Retry interval: indiceert hoelang de secundaire nameserver moet wachten om de primaire server opnieuw te contacteren in geval van een mislukte refresh.
  - Expire interval: hoelang de gegevens mogen bewaard worden indien de refresh mislukt is.
  - Default TTL: de standaard TTL waarde voor records.

Vervolgens moet, vooraleer andere records uit te schrijven, duidelijk gemaakt worden welke nameservers verantwoordelijk zijn voor de zone (zelf al is de primaire server vermeld in het SOA record de enige naamserver voor deze zone).

```
@  NS  primaire_dns
@  NS  secundaire_dns
```

In tegenstelling tot het SOA record, mogen deze DNS vermeldingen relatief zijn. Bij de root server tenslotte moet de naam hier gebruikt ook nog expliciet in een A-record geregistreerd worden, aangezien die nergens anders opgezocht kan worden.

Stel het configuratiebestand en alle zonebestanden op van volgende DNS servers, waarbij je er rekening moet mee houden dat elk van deze servers ook secundaire nameserver is voor alle zones van de andere server. Gebruik relatieve DNS namen waar mogelijk. Gebruik noch forwarders, noch de $ORIGIN opdracht!

2.5 Configuratie van DNS servers onder Linux

De figuur in bijlage stelt een intranet bestaand uit een aantal Linux ...(cfr.vraag B3)... achterhalen. Geen enkele zone heeft een secundaire nameserver. Je hoeft geen reverse DNS te configureren.

Bespreek in detail het formaat van een configuratiebestand. Je mag dit doen op basis van één van de oplossingen in a), doch je moet ook alternatieve opdrachten en sleutelwoorden beschrijven, die je niet noodzakelijk hebt gebruikt.

Een configuratiebestand is onderverdeeld in een aantal opdrachten, bepaald door sleutelwoorden. We bespreken de twee belangrijkste sleutelwoorden: *zone* en *options*.

options Hierin worden een aantal opties gegroepeerd, waarbij elke optie bestaat uit een sleutelwoord, eventueel een aantal parameters, en tenslotten een afsluitende kommapunt. Een aantal veelvoorkomende opties zijn:

- `directory $PAD`: hierbij wordt de map aangeduid waarin de zonebestanden zich bevinden.
- `forwarders { @ADRESSEN }`: hierin worden de IP-adressen van mogelijke forwarders geconfigureerd. Deze forwarders zijn nameservers die de server zal gebruiken indien hij de DNS-informatie niet in zijn eigen cache vindt, wat de belasting van de rootservers verlaagt. Als ook die servers geen antwoord op de vraag kunnen bieden, zal de server toch contact op nemen met de rootserver.
- `allow-transfer { @ADRESSEN }`: deze lijst van duidt aan welke clients zonetransfers kunnen uitvoeren (indien deze optie niet aanwezig is, kunnen alle clients dat). Dit wordt gebruikt door secundaire nameservers, alsook door `nslookup` indien de `ls` parameter meegegeven is. Hierbij kunnen ook netwerkadressen met prefixlengte gespecificeerd worden.

zone Deze opdrachten configureren de effectieve zones waarvoor de geconfigureerde DNS-server moet instaan (hetzij primair of secundair). Na dit sleutelwoordt volgt de naam van de zone binnen dubbele quotes, eventueel gevolgd door een blok met zone-specifieke configuratie. De zone-specifieke configuratie hangt af van het type server, bepaald door het sleutelwoord `type` in het configuratieblok:

- `master`: een primaire nameserver.
  - `file $BESTAND`: de naam van het zonebestand
  - `allow-update $ADRES`: een optionele optie die bepaalt welke clients dynamische updates mogen uitvoeren (het adres kan hierbij ook een netwerkadres met prefixlengte zijn).
  - `allow-transfer $ADRES`: een optionele optie die bepaalt welke clients zonetransferts mogen uitvoeren (het adres kan hierbij ook een netwerkadres met prefixlengte zijn).
- `slave`: een secundaire nameserver.
  - `masters ADRESSEN`: een optie die specificeert welke nameservers verantwoordelijk zijn voor deze zone.
  - `file $BESTAND`: het bestand waarin de opgehaalde data zal opgeslagen worden (dit wordt eventueel eerste aangemaakt).
- `stub`: een secundaire nameserver die enkel NS-records dupliceert van de opgegeven master servers.
- `forward`: een nameserver die louter requests doorstuurt naar een antere nameserver.
- `hint`: een speciaal type server, enkel mogelijk bij de "." zone, die gebruikt wordt door de nameserver om bij het opstarten de namen van echte root-servers te vinden.
  - `file $BESTAND`: het bestand waarin de rootservers neergeschreven staan.

```
zone ‘‘.’’ {
  type hint;
  file ‘‘named.ca’’;
}
```

```
@                    3600000 IN NS A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.  3600000    A  198.41.0.4
```

Stel het configuratiebestand en alle zonebestanden op van volgende DNS servers. Gebruik relatieve DNS namen waar mogelijk. Gebruik noch forwarders, noch de $ORIGIN opdracht!

2.6 Configuratie van reverse DNS onder Linux

De figuur in bijlage stelt een intranet voor bestaand uit een aantal Linux computers. Alle computers hebben een IP-adres van de vorm 172.x.y.z . De getallen x en y lees je af rechts van de naam van de computer. Het getal z lees je af links van de naam van de computer. De computers staan gegroepeerd in een tabel met als header de naam van het domein waarin ze zich bevinden. De recht hoeken die domeinen groeperen stellen dan weer een zone voor. Stippel lijnen duiden op een domein/sub domein relatie. De pijlen laten toe om de primaire name server van elke forward DNS zone te achter halen. De reverse DNS van de volgende adresruimten wordt gedelegeerd aan: ... . Geen enkele zone heeft een secundaire nameserver.

Wat wordt beoogd met reverse DNS? Hoe realiseert men dit (§3.3.2)?

Reverse DNS is de technologie die toelaat om de domeinnaam op te zoeken voor een gegeven IP-adres. Dit wordt vaak toegepast als beveiligingscontrole. De DNS-records verantwoordelijk voor deze omgekeerde omzetting zijn van het type PTR.

Om de naam van de machine met adres w.x.y.z op te zoeken, moet het PTR-record van de autoritaire server met die informatie opgevraagd worden. Aangezien we de domeinnaam van deze server niet kennen, kunnen we die niet zomaar contacteren. Om dit op te lossen bestaat er een pseudo-domein met de naam *in-addr.arpa*. Elk IP-adres heeft vervolgens een geassocieerde naam in dit speciale domein, zo zal onze op te zoeken machine overeen komen met *z.y.x.w.in-addr.arpa*. Met dit adres kunnen we nu een hiërarchische aanvraag doen die begint bij de server verantwoordelijk voor het *in-addr.arpa* adres, en uiteindelijk uitkomt bij diegene verantwoordelijk voor de machine die we zoeken. Dit werkt aangezien de numerieke opdeling meestal wel deels overeenkomt met de effectieve hiërarchie. Wanneer tenslotte de hiërarchie niet op te delen is in subnetten waarbij de prefixlengte deelbaar is door 8, zal de uitspenderende server verschillende aanliggende specifiekere blokken moeten uitdelen (*supernetting*).

Om reverse DNS toe te laten op eigen machines, moet er dus een primaire nameserver geconfigureerd worden voor het overeenkomstige subdomein van *in-addr.arpa*. Dit realiseren we binnen een zonebestand als volgt (voorbeeld voor de nameserver verantwoordelijk voor 192.168.16/24):

- Identiek SOA-record (behalve het volgnummer).
- Enkel PTR records in de zone.
- Alle namen in het bestand zijn relatief ten opzichte van 16.168.192.in-addr.arpa.
- Getallen in de linkerkolom PTR records duiden op ip-adressen (relatief ten opzichte van 192.168.16), terwijl de namen in de rechterkant absoluut zijn.

Tenslotte moet de nameserver verantwoordelijk voor het 168.192.in-addr.arpa duidelijk gemaakt worden dat het 16 subdomein aan bovenstaande nameserver moet gedelegeerd worden.

Stel het configuratiebestand en de zonebestanden op van alle servers die een rol spelen bij de reverse DNS resolving van IP-adressen behorend tot de adresruimten ... en ... . Je hoeft hierbij enkel de configuratie informatie te vermelden die noodzakelijk is voor reverse DNS. Gebruik relatieve DNS namen waar mogelijk. Het gebruik van forwarders is hier niet toegelaten!

Hoofdstuk 3

Reeks C

3.1 Netwerkbeheer

Waarom is netwerkbeheer noodzakelijk?

Financieel Men moet pro-actief het aantal incidenten, waardoor bepaalde diensten onbruikbaar worden, proberen te verminderen. Dit leidt tot een hogere producitiviteit en lagere globale kosten, niettegenstaande men ook moet investeren in hulpmiddelen en personeel voor het netwerkbeheer zelf.

Technisch Mogelijke probleemsituaties en punten van verbetering zijn in een strict beheerde omgeving veel gemakkelijker op te sporen. Hierdoor kan men sneller reageren op veranderingen in gebruik of op capaciteitsproblemen.

Beveiliging Er voor zorgen dat (enkel) de juiste personen toegang hebben tot de bepaalde diensten, door de volledige verzameling van bronnen correct te inventariseren. Indien toch een inbreuk zou gebeuren, is het herstel tot gecontroleerde toestand veel sneller dan in een onbheerde omgeving.

Professioneel De toekomstperspectieven van het globale bedrijf alsook de netwerkbeheerder zelf hangen voor een groot deel af van een goed beheerde informaticastructuur. Ook de jobvoldoening van de netwerkbeheerder is bepalend voor de mate waarin ze het gevoel hebben de componenten onder controle te hebben.

Aan welke randvoorwaarde moeten oplossingen voor netwerkbeheer beantwoorden?

Men moet netwerkbeheer in de ruimste zin van het woord interpreteren. De grens tussen systeembeheerder en netwerkbeheerder is zeer vaag geworden. Alle componenten die een essentiële bijdrage leveren tot de diensten die aan de gebruikers aangeboden worden, komen in aanmerking voor nauwgezet beheer:

- Netwerkcomponenten
- Systemen
- Hardware elementen
- Software toepassingen
- Alle niveau's van het OSI/Internet referentiemodel

Netwerkbeheer moet uitgaan van een uniforme, gecentraliseerde aanpak maar met mogelijkheid tot meer gedistribueerde aanpak opdat het NMS zelf geen *bottleneck* zou worden. Dit biedt het voordeel dat het netwerkbeheer niet zelf hoeft onderbroken te worden.

De oplossingen voor netwerkbeheer mogen geen te grote impact hebben op de kosten en belastingen. Daarom moet men de principes alsook de implementatie ervan eenvoudig houden.

Oplossingen dienen ook modulair implementeerbaar te zijn. Dit vergemakkelijkt onder andere toevoeging en afschaffing van hulpmiddelen voor netwerkbeheer.

De oplossingen moeten zoveel mogelijk gebruik maken van de huidige protocolstack. Ze mogen dus zo weinig mogelijk afhankelijk te zijn van de hogere protocollagen.

Beschrijf het model dat de functionele eisen voor netwerkbeheer vastlegt. Geef van elke categorie de meest typische aspecten.

Configuratiebeheer Dit mag in geen enkele omgeving ontbreken en omvat twee deelgebieden. De configuratie van hardware infrastructuur en de configuratie van software.

Hardware configuratie komt vooral neer op een gedetailleerde inventaris van alle systemen en hun componenten. Dit kan deels geautomatiseerd worden door middel van *discovery* procedures.

Waar hardware zich vooral toespitst op de structuur van elementen, heeft *software configuratie* vooral aandacht voor het functionele aspect. Software configuratie is ook complexer, aangezien ze meer frequent wijzigt, en meer rekening moet houden met de verschillende relaties tussen de componenten. Er zijn twee belangrijke aandachtspunten:

- *Change managment* zorgt voor onmiddellijk beschikbare en preciese informatie over alle wijzigingen aangevuld met de mogelijkheid om dezelfde configuratiewijziging op meerdere componenten en tegelijkertijd uit te voeren. Dit laat uiteindelijk toe het netwerk steeds beter te optimaliseren.
- *Asset managment* houdt een overzicht bij van de componenten die mogelijk aan vervanging toe zijn en ook de planning die men moet volgen om de verouderde elementen te vervangen door nieuwe technologie.

Accounting-beheer Dit kent twee subfuncties:

- *Value assesment* kent aan iedere netwerkcomponent een kwantitatieve waarde toe, kenmerkend voor de belang ervan bij dienstverlening. Deze waarde laat toe bij crisissituaties de aandacht te besteden op de elementen met hoogste prioriteit. Ook bij vervanging van componenten kan dit een indicatie zijn.
- *Usage audits* houden precies bij wie welke netwerkbronnen gebruikt op elk ogenblik. Zo kan men onder meer de kosten verhalen voor een specifieke afdeling of individu binnen de onderneming. Opvolging van gebruik steunt op dezelfde technieken die men aanwendt bij *auditing* van bronnen om veiligheidsredenen en wordt er dan ook vaak mee gecombineerd.

Performantiebeheer Dit heeft als doel een netwerkinfrastructuur de hoogst mogelijke niveaus van betrouwbaarheid, beschikbaarheid en *throughput* aan te bieden en impliceert een hele reeks taken: uittesten van media, simulatie, tuning, planning, opsporen van *bottlenecks* en vooral *baselining*. Dit betekent het periodiek meten van de belasting van de diverse componenten over een bepaald tijdsinterval. Hierdoor kan men trends en evoluties vaststellen alsook een goed beeld krijgen inzake (over)belasting. Dikwijls worden performantiegegevens bekomen door middel van *probes* in hardware, die continue het verkeer op pakket niveau analyseren. Rapporteringstools spelen een belangrijke rol.

Security-beheer Is vooral gericht om ervoor te zorgen dat de diverse netwerkbronnen enkel door geautoriseerde gebruikers kunnen benaderd worden en dat inbreuken of pogingen tot onmiddellijk gerapporteerd en hersteld kunnen worden.

- *Value assesment* kent aan iedere type date een kwantitatieve waarde toe.
- *Risk managment* schat de gevolgen van eventuele blootstelling of corruptie in.

Naast deze voorbereidende stappen worden *audits* uitgevoerd om gaten in de beveiliging op te sporen.

Foutopvolging Het pro-actief vermijden van problemen en deze zo snel mogelijk detecteren alsook rapporteren. Rapporteren van fouten kan door middel van *events*, *trouble tickets* of een *alarm*. In sommige gevallen is het beheerssysteem zelf in staat een actie te ondernemen om fouten te herstellen.

Naast de detectie en rapportering van fouten dient men ook een analyse van de oorzaak en van de eventuele alternatieve remedies te maken. Dit om analoge fouten in de toekomst te vermijden.

Foutopvolging is voor velen misschien wel het belangrijkste aspect van netwerkbeheer maar is voor efficiënte uitvoering volledig afhankelijk van de vier andere domeinen.

3.2 SNMP protocol

Belangrijke opmerking: alhoewel niet expliciet gevraagd, word je bij de beantwoording van de SNMP vragen geacht om de globale structuur en de meest typische knooppunten en objecten van volgende componenten te kennen:

- De IP groep van MIB-II (§4.3.1.4 en voorbeelden in §4.2.4).
- De host resources MIB (§4.3.3 en voorbeelden in §4.2.4).

Geef de ASN.1 codering van een (niet-TrapResponse) SNMP bericht, en een korte uitleg bij elke TLV.

Niet-TrapResponse berichtne (zoals GetRequest, GetResponse en SetRequest) volgen allemaal hetzelfde stramien:

```
Message ::= SEQUENCE {
   version INTEGER
   community OCTET STRING
   protocolDataUnit SEQUENCE {
      requestID INTEGER
      errorStatus INTEGER
      errorIndex INTEGER
      varBindList SEQUENCE OF {
         varBind SEQUENCE {
            name OBJECT IDENTIFIER
            value ObjectSyntax } } } }
```

Met elke lijn komt in de ASN.1 komt een *Tag-Length-Value* triplet overeen. Volgende scalaire TLVs zijn aanwezig:

- *version*: bevat voor SNMPv1, SNMPv2c en SNMPv3 respectievelijk de waarden 0, 1 of 3.
- *community*: wordt ingevuld met onversleuteld wachtwoord.
- *requestID*: deze waarde wordt random of incrementeel gegenereerd. Dient om SNMP antwoorden aan de juiste vraag te correleren, en om eventueel geduplicerde berichten te elimineren.
- *errorStatus*: bevat de foutcode wanneer een fout is opgetreden. Heeft als waarde 0 voor een *Request* bericht.
- *errorIndex*: deze waarde kan enkel geïnterpreteerd worden als *errorStatus* niet 0 is, en toont op welk object in *varBindList* de fout betrekking heeft.

- *name*: dit is het OID van het object. Elke component wordt door een byte gecodeerd, behalve de eerste twee (*iso.org*), die samen genomen worden. Indien waarde component groter of gelijk is aan 128 (80h) wordt de component in de zeven laagste bits van twee bytes gecodeerd. Eerste byte krijgt als hoogste bit een om de afwijking aan te tonen.
- *value*: in dit veld wordt de waarde van het object gestuurd. Bij een *Request* bericht is dit niet gekend en gebruikt men de waarde 05, *NULL* datatype.

Daarnaast zijn er nog enkele TLVs om het bericht te structureren:

- *varBind*: deze vormen samen een *name value* koppel (bij een Request is de value gelijk aan *null*).
- *varBindList*: groepeert een aantal *varBind* TLVs om zo meerdere objecten tegelijkertijd op te vragen met een *Request* bericht.
- *protocolDataUnit*: dit is een overkoepelende TLV die in zijn *tag* aanduidt om welk soort SNMP bericht het gaat.

```
a0 GetRequest
a1 GetNextRequest
a2 GetResponse
a3 SetRequest
a4 TrapResponse
a5 GetBulkRequest
```

Indien de lengte van het *value* veld groter of gelijk is aan 128 (80h), dan wordt de lengte gecodeerd door zoveel bytes als vereist. Dit wordt voorafgegaan door een extra byte die in de hoogste bit de afwijking aanduidt en in de andere bits het noodzakelijk aantal bytes aanduidt. Dit verschilt van de techniek die toegepast wordt als een veld uit het OID groter is dan 128: hierbij zal men de waarde encoderen als een reeks bits, waarbij 7 waarde-bits steeds voorafgegaan wordt door een indicatie-bit dat enkel bij de laatste relevante byte 0 is.

Dit is de algemene structuur van een *GetRequest* bericht. Naast dit type bericht zijn er ook andere met gelijkaardige structuur.

GetResponse Een *GetRequest* bericht wordt beantwoord met een *GetResponse* bericht. Heeft exact dezelfde structuur. Maar nu wordt uiteraard de inhoud van *protocolDataUnit* en van de *value* TLVs aangepast. Nu kunnen fouten opgeslagen worden in *errorIndex* en *errorStatus* indien de waarde van een van de objecten niet kan opgezocht worden.

```
0 noError
1 tooBig
2 noSuchName
5 genErr
```

SetRequest Wordt gebruikt om de waarde van een object te wijzigen of om een nieuwe rij toe te voegen aan een tabelobject. De structuur is identiek aan die van een *GetRequest* bericht. Wordt beantwoord met *GetResponse* bericht met relevante foutcodes.

```
3 badValue
4 readOnly
```

GetNextRequest Laat toe de exacte structuur van een MIB boom te achterhalen. Heeft net dezelfde structuur als een *GetRequest* bericht.

GetBulkRequest Vraagt aan de agent om een *GetResponse* bericht met zoveel mogelijk informatie terug te sturen. Twee TLVs hebben echter een volledig andere interpretatie:

- *errorStatus* wordt nu *nonRepeaters*
- *errorIndex* is vervangen door *maxRepeaters*

Interpreteer de inhoud van volgende (hexadecimale) dump [...] van een SNMP bericht. Geef ondermeer aan om welk soort bericht het gaat.

Indien het om een response bericht zou gaan, construeer dan de meest efficiënte oplossing voor de bijhorende request. Indien het om één of andere vorm van een request bericht zou gaan, construeer dan de bijhorende response.

3.3 Overgang naar IPv6 autoconfiguratie

Bespreek de belangrijkste motieven die uiteindelijk een overgang naar IPv6 zullen veroorzaken (§5.1).

Hoewel IPv4 relatief goed geschaald heeft, kent het verschillende beperkingen die de overgang naar IPv6 nodig maken:

- Beperkte adresruimte: door slecht 32 bit als adres te gebruiken, kent de adresruimte maar 4 miljard adressen wat gezien de te verwachten opkomst van clients onvoldoende is. Door bovendien gebruik te maken van een hiërarchisch adresseringssysteem (CIDR) heeft men weliswaar geen nood aan een centrale autoriteit maar wordt er slechts een klein gedeelte van de adresruimte ook effectief gebruikt (in tegenstelling tot een sequentiële nummering). Daarom wordt heden ten dage veel gebruik gemaakt van *Network Address Translation* (NAT), wat deze beperking gedeeltelijk omzeilt.
- Grote routingtabellen: hoewel het hiërarchische CIDR aggregatie van routes toelaat, is dit te laat geïntroduceerd waardoor veel tier-1 ISPs verantwoordelijk zijn voor meerdere niet-aggregeerbare blokken en hun *default-free* routers (ASBRs die

normaal geen default routes maar enkel entries voor andere tier-1 ISPs zouden moeten bevatten) gigantische routingtabellen kennen.

- Pakketstructuur: IPv4 kent opties van variabele-grootte als onderdeel van de header, waardoor routers verplicht die opties moeten verwerken (wat de performantie negatief beïnvloedt). Bovendien kunnen pakketen gefragmenteerd worden, wat op globale schaal vertragend werkt.

- Beveiliging: dergelijke functionaliteit is niet geïntegreerd in het IPv4 protocol, waardoor gebruikt moet gemaakt worden van hogere protocollagen (zoals SSL). Hierdoor blijft heel wat informatie, zoals de headers, steeds onbeschermd.

- Mobiliteit: IPv4 kent geen faciliteiten ten behoeve van mobiliteit, de configuratie is steeds statisch (met uitzondering op het APIPA mechanisme). Dit moeten dus steeds in de toepassingslaag moeten gerealiseerd worden (zoals DHCP).

Hoewel er voor de meeste van deze problemen oplapmiddelen bestaan, is het duidelijk dat IPv4 de voortdurende groei niet veel langer aankan.

Hoe zal deze overgang worden gerealiseerd? Van welke technieken zal men gebruik maken? Je moet hierbij onder andere tunneling in algemene termen beschrijven, echter zonder dieper in te gaan op de praktische uitwerking ervan (§5.7 zonder subsecties).

Dankzij de gelaagde opbouw van de netwerkstack, blijft de impact vrij beperkt. Zo moet men enkel op subnetwerkniveau de verwijzing naar de hogere protocollaag aanpassen (praktisch: het veld binnen Ethernet frames dat de het type pakket aanduidt). Hogere protocollagen moeten in principe niet aangepast worden, hetzij om rekening te houden met de viervoudige lengte van de IP adressen (vb de checksum binnen de TCP header).

Bij het implementeren van IPv6 worden gerelateerde protocollen binnen de internetlaag ook aangepast worden: zo mogen ARP en IGMP verwijderd worden, en zal ICMP moeten aangepast worden (ICMPv6). Ook zullen hogere protocollen die toch sterk gebonden zijn met de IP-implementatie (zoals DHCP en DNS) aangepast moeten worden.

Rekening houdende met het enorme aantal netwerken en knooppunten dat aangepast zal moeten worden, zal de omschakeling geleidelijk aan moeten gebeuren met een aantal randvoorwaarden:

- IPv4 knooppunten moeten op een willekeurig moment kunnen geüpgradet worden.

- Alleen-IPv6 knooppunten moeten onafhankelijk van andere knooppunten kunnen toegevoegd worden.

- Bestaande alleen-IPv4 knooppunten moeten hun IPv4 adres kunnenblijven gebruiken (door IPv6 knooppunten gemapt als ::FFFF:w.x.y.z adres).

- Alle IPv6 knooppunten moeten interoperabel blijven met IPv4 systemen.

IPv6 zal lange tijd moeten samenleven met IPv4, vermoedelijk in de vorm van steeds groter wordende IPv6 eilandjes. Om deze overgang eenvoudig te realiseren wordt meestal gebruik gemaakt van een combinatie van drie strategieën:

1. Dual-IP of IPv6/IPv4 systemen: deze systemen ondersteunen beide protocollen en kunnen dus knooppunten van beide types op een subnet bereiken. Dit wordt gerealiseerd met *dual-stack* systemen, waarbij meestal ook de transportlaag gesplitst is.

2. IPv6 over IPv4 tunneling: hierbij worden IPv6 datagrammen in IPv4 pakketten ingekapseld alsof het data was van een hogere protocollaag (met protocolidentificatie 41). Het uiteindelijke IPv6/IPv4 knooppunt verwijdert vervolgens de IPv4 header om de IPv6 inhoud te verwerken (geen permanente tunnel dus). Daarom moeten zowel begin- als eindpunt dual-stack systemen zijn. Deze techniek is essentieel in de beginfase van de omschakeling daar het een *end-to-end IPv6 service* biedt zonder een eventuele IPv4 backing.

3. Protocol of Header Translation: hierbij zetten IPv6/IPv4 routers IPv4 headers om in een IPv6 header, door gebruik te maken van IPv4-gemapte adressen. Dit zal vooral belangrijk zijn in de eindfase van de omzetten, om de laatste IPv4 systemen connectiviteit te bieden.

Geef de alternatieve mogelijkheden voor autoconfiguratie in IPv6. Bespreek hierbij de opeenvolgende stappen (§5.6).

Los van het type configuratie, kan men steeds identificeren in welke toestand een client zich bevindt:

- Tentatief: tijdens deze fase is het adres vastgelegd, maar nog niet geverifiëerd. Deze controle gebeurt door *Duplicate Address Detection*, dat gecontroleert of het nog niet in gebruik is. Hierbij kan de client enkel multicast verkeer ontvangen.

- Valid: eenmaal een uniek adres zijn geldigheid bevestigd is, kan de client ook unicast data verzenden en ontvangen. Aangezien de levensduur van een adres beperkt is, bevindt de client zich steeds in de *preferred* hetzij *deprecated* toestand (waarbij het knooppunt verantwoordelijk voor het adres een nieuw adres moet uitdelen, of het bestaande adres vernieuwen).

- Invalid: deze fase komt voor wanneer de levensduur van een adres verstreken is. Hierbij kan de client unicast noch multicast verkeer verwerken.

IPv6 kent twee grote configuratiemechanismen, *stateless autoconfiguratie* en *stateful autoconfiguratie*.

Stateless autoconfiguratie Hierbij zal een client autonoom zijn configuratie kunnen bepalen, zonder daarvoor expliciet een server te moeten ondervragen. Daarbij kan het interface-id bepaald worden aan de hand van het *IEEE EUI-64* formaat van een MAC-adres, en kan de netwerkprefix bekomen worden aan de hand van *Router Discovery*. Dergelijke autonome configuratie is echter enkel mogelijk voor niet-routers (routers moeten, buiten hun linklocale unicast adressen, steeds manueel geconfigureerd worden).

Meer in detail verloopt stateless autoconfiguratie als volgt:

- Bepalen tentatief linklocaal unicast adres: dit gebeurt op basis van het MAC adres, en zorgt (na Duplicate Address Detection, waarbij indien falen de configuratie manueel moet uitgevoerd worden) voor communicatiefunctionaliteit op het lokale subnet.
- Initialisatie interface: hierbij wordt het tentatief linklocaal unicast adres gebruikt. Ook wordt het multicast MAC-adres overeenkomstig het *solicited-node* multicast adres geregistreerd (eerste 3 bytes in het MAC-adres vervangen door 33-33-FF).
- Routers detecteren: door drie *Router Sollicitation* berichten te sturen. Bij ontvangst van *Router Advertisment* berichten wordt de configuratie voor de *Hop Limit*, *Reachable Time*, *Retrans Timer* en *MTU* er uit afgeleid. Indien niet wordt overgeschakeld op stateful autoconfiguratie.
- Bepalen nieuw adres: op basis van de ontvangen *Prefix Information* wordt een nieuw tentatief unicast adres berekend (opnieuw met Duplicate Address Detection). Indien de prefixlengte overeen komt met een publiek subnetwerk wordt dit ten behoeve van anonimiteit gebaseerd op een random MAC-adres.
- Stateful configuratie: indien de *Managed Address Configuration* vlag in het *Router Advertisement* bericht aan staat, wordt een statfull autoconfiguratie uitgevoerd om bijkomende adressen te bekomen (of om andere informatie te bekomen indien de *Other Stateful Configuration* vlag aanstaat).

Stateful autoconfiguratie Dit configuratiemechanisme maakt gebruik van een centrale server die toestandsinformatie bijhoudt van alle knooppunten. Dit heeft als voordeel dat er gecontroleerd kan worden wie er een adres ontvangt (om zo eventueel enkel expliciet geautoriseerde clients toe te laten). Ook is er een strikterecontrole over het adresseringsschema daar ook de interface-ID's kunnen vastgelegd worden.

Een dergelijke configuratie kan bekomen worden via het DHCPv6 protocol, dat enkele fundamentele wijzigingen brengt ten opzichte van DHCP:

- Niet BOOTP-compatibel.
- Interactie via multicasting op een aantal permanente adressen.
- Kan op stateless configuratie berusten om stateful configuratie te vereenvoudigen.
- Bepaalde opties worden niet meer ondersteund (zoals optie 3, default routers).

- Er kunnen meerdere adressen toegewezen worden aan eenzelfde interface.
- Toevoeging van het *DHCP-Reconfigure* berichttype om op initiatief van de server configuratiewijzigingen naar clients te sturen.
- Adres deprecation (de toestant waarin een adres op het punt staat te vervallen) kan gebruikt worden om netwerken dynamisch te hernummeren.

3.4 Adressering (§5.2 behalve §5.2.5)

Bespreek de structuur en numerieke voorstelling van IPv6 adressen.

Een IPv6 adres bestaat uit 128 bits (16 bytes), waardoor een enorme adresruimte ter beschikking komt. Hierdoor zijn lapmiddelen zoals NAT af te raden. De numerieke voorstelling is als volgt:

- Basisrepresentatie in de vorm van P:Q:R:S:T:U:V:W.
- Voorloopnullen niet nodig: FFE1:1:... in plaats van FFE1:0001:....
- Opeenvolgende [:]0000[:] strings te vervangen door ::, bijvoorbeeld FFE1::1EFF in plaats van FFE1:0000:0000:0000:0000:0000:0000:1EFF.
- Een willekeurig 2-byte segment mag steeds voorgesteld worden in de *dotted-decimal* notatie: FE80::5EFE:192.168.41.30 in plaats van FE80::5EFE:C0A8:291E.

Elk adres is net zoals bij IPv4 opgebouwd uit een netwerkprefix en een interface-id, maar nij IPv6 kan enkel de prefixlengte syntax gebruikt worden om de grootte van de netwerkprefix aan te duiden: 2001:410:1::/48.

Geef en bespreek de verschillende types IPv6 adressen. Geef onder andere van elk van deze types de structuur, hun interpretatie, relevante voorbeelden en eventuele subtypes.

IPv6 kent 3 types adressen: *unicast*, *multicast* en *anycast*. Bij unicast adressen kunnen we nog het onderscheid maken tussen globaal versus lokaal beschikbare unicast adressen. Verder blijft het concept van unicast adressen echter relatief ongewijzigd ten opzichte van IPv4 (zo moet elke interface er minstens 1 hebben).

Globale unicast adressen Globale unicast adressen, gekenmerkt door de formaat-prefix (eerste drie bits) 001, moeten mondiaal uniek zijn. Interfaces die dit ondersteunen kunnen echter eenzelfde globale unicast adres delen om de netwerkbelasting te splitsen.

Terwijl een IPv4 unicast adres opgedeeld werd in twee variabele delen (netwerkprefix en subnet-ID), is er ruimte genoeg op een IPv6 adres in meerdere hiërarchische niveaus op te delen:

1. *Top-Level Aggregation Identifier* (TLA): een 13-bit veld dat de hoogste niveau in de routinghiërarchie voorstelt (toegewezen aan tier-1 ISPs). De grootte van dit veld zorgt ervoor dat default-free routers ten hoogste 8192 routes met prefixlengte /16 zullen bevatten (er zijn echter 8 reservebits voorzien om dit later eventueel uit te breiden).

2. *Next-Level Aggregation Identifiers* (NLAs): dit 24-bit veld wordt consequent op provider-niveau toegewezen, en vormt samen met de TLA de publieke topologie van het internet. De structuur ervan is echter onzichtbaar voor default-free routers.

3. *Site-Level Aggegation Identifier* (SLA): dit 16-bit veld is voor de klant beschikbaar om de topologie af te stemmen op de structuur van zijn privaat netwerk. Aangezien dit veld 16-bit groot is, kan de oude IPv4 structuur identiek herbruikt worden om de SLA te vormen.

4. *Interface-ID*: dit 64-bit segment tenslotte kan random bepaald worden, of eenduidig afgeleid worden uit het (tevens globaal unieke) MAC-adres van de netwerkkaart.

De omzetting van een 6-byte MAC-adres naar een 8-byte interface-id gebeurt meestal als volgt: men neemt de eerste 3 bytes van het MAC-adres, voegt vervolgens FF:FE in, om vervolgens de laatste 3 bytes van het MAC-adres te gebruiken. Tenslotte complementeert men eventueel de laagste bit van het adres zodat het op 0 staat, wat vereist wordt bij een unicast MAC-adres. Deze voorstelling heet men het *IEEE EUI-64* formaat.

Lokale unicast adressen Deze adressen dienen voor intern gebruik binnen een organisatie, en worden niet doorgestuurd door routers buiten de organisatie. Buiten hun formaatprefix is er niks dat ze van andere geldige IPv6 adressen onderscheid. Er zijn twee soorten lokale unicast adressen:

1. Sitelokale adressen (formaatprefix FEC0:0:0/48): deze adressen bezitten een vrij te kiezen 16-bit SLA veld waardoor subnetting mogelijk is. Ze zijn dan ook te vergelijken met de IPv4 private adresblokken 10/8, 172.16/12 en 192.168/16. Ze kunnen gebruikt worden bij het verwerken van verkeer zonder dat direct versturen naar het Internet mogelijk is.

2. Linklokale unicast adressen (formaatprefix FE80:0:0:0/64): deze adressen worden gebruikt om knooppunten op een enkele netwerkverbinding te nummeren, en laten dan ook geen subnetting toe. Ze zijn te vergelijken met de 169.254/16 APIPA adressen, maar worden steeds automatisch toegewezen aan elke interface (zelfs als die reeds over andere unicast adressen beschikt).

Multicast adressen Dit concept blijft tevens relatief ongewijzigd ten opzichte van IPv4, en kan gebruikt worden om informatie ter beschikking te stellen aan meerdere clients. Die kunnen zich inschrijven op het adres (waarbij tussenliggende routers die inschrijving namens de client moeten doorsturen), en ook data sturen naar het multicast

adres in kwestie. Om dit te verwezenlijken wordt de netwerkinterface geconfigureerd met een lijst MAC-multicast adressen die doorgestuurd moeten worden naar een hogere protocollaag.

Bij IPv6 wordt deze notatie gekenmerkt door het eerste segment gelijk te stellen aan FFvs::

- Vlag veld (v): bepaalt het type adres, waarbij nu enkel de eerste 4 bytes vastliggen:
  - $v = 0$: het adres is een permanent adres (vastgelegd door het IANA), waarbij de overige 7 bits de speciale betekenis vastleggen (alle knooppunten, alle routers, alle OSPF routers, ...). Het multicast adres dat hierbij alle knooppunten adresseert, vervangt het broadcastmechanisme uit IPv4.
  - $v = 1$: het adres is een transient (tijdelijk) adres, dat ad-hoc kan worden gebruikt. Dergelijke adressen van de vorm FF1s::w.x.y.z worden daarbij automatisch gemapt op Ethernet multicast adressen van de vorm 33-33-w-x-y-z.
- Scope veld (s): geeft aan welk bereik het multicast adres heeft (globaal, site-lokaal, subnet, organisatie, etc).

Bij het configureren van een interface wordt naast een linklokaal unicast adres ook een *solicited-node* multicast adres van de vorm FF02::1:255.x.y.z aan de interface toegewezen, waarbij de laatste 3 bytes bepaald worden door de laatste 3 bytes van het unicast IP adres. Hiermee zorgt men dat adresresolutie meestal slechts een enkele client verstoord, in plaats van alle knooppunten op het netwerk zoals bij IPv4.

Anycast adressen Dergelijke adressen kunnen door meerdere knooppunten gedeeld worden, maar zullen slechts door een enkel knooppunt ontvangen worden (in tegenstelling tot datagrammen die naar een multicast adres gestuurd worden). Dit wordt handig toegepast bij het verlenen van diensten, door aan elk type dienst een specifiek anycast-adres te linken. Als client-software vervolgens een bepaalde dienst nodig heeft (zoals een timeserver), zal die het adres van de server niet exact moeten kennen, maar gewoon het anycast adres moeten aanspreken om bij de correcte server terecht te komen. Anycast adressen worden toegewezen uit de unicast-adresruimte, en kunnen er niet van onderscheiden worden. Daarom moet een interface waarop een dienst verleend wordt, expliciet geconfigureerd worden om dat anycast adres te accepteren. Ook de routers moeten op de hoogte zijn van elke locatie van ieder anycast adres.

De interfaces van een router worden ook steeds geconfigureerd met het *subnet-router anycast adres*, gevormd door het netwerkadres aan te vullen met een interface-ID dat uit allemaal nullen bestaat. Hierdoor is het eenvoudig om op een subnet een willekeurige router te bereiken, door als het ware het netwerk te adresseren, wat in IPv4 niet mogelijk was (`ping 192.168.0.0`).

3.5 IPv6 berichtstructuur (§5.3)

Bespreek in detail de structuur van IPv6 berichten

In tegenstelling tot IPv4, kent IPv6 een header met vaste grootte. De opties worden geëncodeerd als een gelinkte lijst van aparte extensieheaders, waardoor efficiënte routing gemakkelijker is (opties moeten enkel onderzocht worden als dat werkelijk nodig is). Zowel de header als de extensieheaders bevatten een *Next Header* veld, dat aangeeft welk type data er volgt: een nieuwe header (met type gespecifieerd), of data van een ander protocol (eveneens met type gespecifieerd, meestal een transportprotocol zoals TCP of UDP). Het geheel van extensieheaders en ingekapselde gegevens noemt men de *payload*.

IPv6 header Ten opzichte van IPv4 kunnen een aandal velden weggelaten worden:

- Header lengte: een IPv6 header heeft nu een vaste grootte van 40 bytes.
- Checksum: dit moet verzorgt worden door het transportprotocol.
- Fragmentatievelden: fragmentatie niet meer mogelijk.

De overige velden zijn meestal analoog aan hun IPv4 tegenhanger:

- Version: ingevuld met waarde 6.
- Traffic Class: analoog aan het TOS veld, geeft mogelijkheid tot een gedifferentieerde dienst (standaard: 0).
- Payload Length: aantal bytes in het datagram, zonder de header (wél extentieheaders).
- Hop Limit: bovengrens aan de levensduur van een datagram, moet in tegenstelling tot IPv4 door hogere protocollagen telkens opnieuw ingesteld worden.
- Bron- en doeladres: uiteraard 16 bytes lang.

Een IPv6 header kent ook een aantal nieuwe velden:

- Flow Label: geeft het verbindingsloze IPv6 protocol enige faciliteit om samenhangende pakketten te detecteren. Als het flow label, dewelke routers in een cache moeten bijhouden, identiek is aan een vooraf-verwerkt pakket, moet de header niet opnieuw volledig verwerkt worden. Zo kunnen pakketten binnen een flow sneller afgehandeld worden. Een flow is ook specifiek voor een bepaalde bron- en eindbestemming, maar er kunnen verschillende flows optreden tussen eenzelfde bron- en eindbestemming.
- Next Header: geeft aan welk type data er volgt.

Geef en bespreek de verschillende soorten extensieheaders die momenteel voor IPv6 gedefinieerd zijn.

Door opties naar extentieheaders te verplaatsen, wordt het gemakkelijker om later nieuwe opties te definiëren. Ook is het niet langer nodig om alle extenties te interpreteren: routers kunnen selectief de extenties verwerken en de rest gewoon doorsturen.

Alle extentieheaders (buiten de ESP) volgen dezelfde TVL-achtige basisstructuur:

- Next Header
- Extension Header Length
- Extension Header Data

De extentieheader moeten ook steeds in een vastgelegde volgorde voorkomen.

Hop-by-Hop Options Deze extentieheader verzamelt alle opties die ook door tussenliggende routers moeten verwerkt worden, zoals:

- Jumbo Payloads optie (type 194): dit maakt het afhandelen van datagrammen groter dan 65535 bytes mogelijk (*jumbogrammen*). Hierbij moet voorkomen worden dat jumbogrammen gestuurd naar netwerken die dit niet ondersteunen, waardoor de optie steeds moet gecontroleerd worden.
- Router Alert optie (type 5): deze optie maakt een router duidelijk dat de informatie in het datagebied ook door de router zelf moet verwerkt worden (bivoorbeeld bij het *Resource Reservation Protocol* of *Multicast Listener Discovery*).
- Opvuloptie (type 1): dit wordt gebruikt om de extentieheader af te lijnen op een veelvoud van 8 bytes. Ook individuele niet TLV-geëncodeerde 0-bytes mogen hiervoor gebruikt worden.

De header moet altijd onmiddellijk na de IPv6 header komen, en is TLV geëncodeerd. Het tag-veld kent daarbij nog een extra betekenis: de twee hoogste bits ervan indiceren wat er moet gebeuren indien de optie onbekend is (fout negeren, ICMPv6 bericht sturen).

Routing header (RH) Deze extensieheader specificeert de route die het pakket moet afleggen, en bevat daartoe een lijst van adressen die het datagram moet bezoeken. Hierbij kan het doeladres in de IPv6 header eventueel slechts een tussenbestemming uit deze lijst zijn. Het *Routing Type* veld beschrijft hoe de routing moet verlopen:

- Type 0: het pakket zal beginnen bij het eerste adres in de adreslijst, van waar het doorgestuurd wordt naar volgende adressen om tenslotte zijn bestemming te bereiken. Hierbij gebruikt men *loose source routing*: tussenliggende routers zijn toegestaan.

Het *Segments Left* veld binnen de RH geeft ook steeds aan hoeveeel adressen nog bezocht moeten worden vooraleer de eindbestemming bereikt wordt.

Fragment Header (FH) In tegenstelling tot IPv4, kan fragmentatie enkel toegepast worden aan de kant van de afzender, door het pakket te splitsen en dezelfde informatie als in de fragmentatieheader van IPv4 in de FH te plaatsen. Een ander verschil is dat het *Fragment Identification* tweemaal zo groot is. Door de beperkte grootte van het *Fragment Offset* veld kunnen jumbogrammen echter niet gefragmenteerd worden.

Tussenliggende hops mogen het pakket niet verder splitsen, en moeten zich dus ook geen zorgen maken over de FH extensieheader waardoor de belasting van routers vermindert. Het nadeel hierbij is dat een *Path MTU Discovery* proces nodig is om de maximale pakketgrootte te bepalen, door het grootst mogelijke datagram te sturen en bij eventuele *ICMPv6 Packet Too Big* foutberichten een kleiner pakket te testen. Om dynamische MTU verhogingen te ondersteunen moet dit proces ook regelmatig opnieuw opgestart worden.

Authentication Header (AH) Dit is een eerste van twee extensies ten behoeve van het *IPsec* mechanisme. De AH bevat daartoe op MD5 gebaseerde handtekening van het datagram, waarmee de ontvanger kan verifiëren of het datagram tijdens transport niet gewijzigd werd. De AH bevat ook een volgnummer dat zorgt voor *antireplay* bescherming: onderschepte datagrammen kunnen later niet opnieuw het netwerk opgestuurt worden.

Encapsulating Security Payload (ESP) Deze tweede extensie laat toe dat men geëncrypteerde datagrammen verzendt, door aan te geven dat alle data vanaf dit punt geëncrypteerd is en genoeg informatie aan te bieden om die ook data ook opnieuw te decrypteren. De ESP kan op twee manieren gebruikt worden:

- Transparante of Transport Mode: hierbij wordt enkel de data geëncodeerd, en blijven headers en extensieheaders ongecodeerd getransporteerd worden.
- Tunnel mode: hierbij wordt het volledige datagram versleuteld en in een ander datagram ingepakt, door het knooppunt dat als *security gateway* werkt. Een security gateway aan de andere kant van de tunnel zorgt vervolgens voor het uitpakken van het datagram. Zo kunnen *Virtual Private Networks* (VPN) gerealiseerd worden zonder daarbij enige informatie te openbaren.

Hoewel de ESP header authenticatiemogelijkheden biedt, wordt aangeraden gebruik te maken van de AH. Deze twee headers komen ook voor bij IPv4, maar worden daar als normale opties aan de header toegevoegd.

Destination Options header Hierin worden alle opties verzamelt die enkel door de eindbestemming moeten worden verwerkt. Deze header is ook de enige die verschillende keren voorkomen in het pakket, dit om te kunnen aanduiden dat een bepaalde router uit de RH deze header ook met verwerken. Hoewel er nu nog geen opties binnen deze header gedefinieerd zijn, zullen ze geformatteerd worden zoals in de Hop-by-Hop options header.

3.6 Tunneling (subsecties §5.7)

Wat is IPv6 over IPv4 tunneling, en waarvoor zal men deze techniek aanwenden?

Bij IPv6 over IPv4 tunneling worden IPv6 datagrammen in een IPv4 datagram geplaatst (met het IPv4 *Protocol Veld* op 41). Datagrammen worden ingekapseld en uitgepakt door de dual-stack eindpunten van de tunnel. Deze techniek is essentiëel bij de eerste fasen van de overschakeling, daar het een end-to-end IPv6 connectiviteit verzekert zonder daar de nodige IPv6 infrastructuur aanwezig voor moet zijn. In latere fasen garanteert het connectiviteit langsheen IPv4-only backbones.

Bespreek in detail de diverse, ook de meest recente, tunnel mechanismen waarop men een beroep kan doen. Bespreek onder andere hun relatieve voor- en nadelen, en de gebruikte addresseringsschema's.

IPv6 over IPv4 tunneling wordt gerealiseerd via verschillende technieken.

Automatische IPv6 over IPv4 tunneling Bij deze vorm van tunneling zal men indien nodig tunnels creëren doorheen de bestaande IPv4 infrastructuur. Of dit gebeurt, hangt af van een aantal criteria:

- Doeladres is IPv4 of IPv4-gemapt: IPv4 wordt gebruikt.
- Eindbestemming bevindt zich op hetzelfde subnetwerk: IPv6 wordt gebruikt.
- Eerste hop van een route naar de eindbestemming is een IPv6 router: IPv6 wordt gebruikt. Hierbij hebben dergelijke IPv6 routes voorrang op automatische tunneling daar het minder overhead veroorzaakt en toelaat de mogelijkheden van het IPv6 netwerk te gebruiken.

Indien niet aan deze criteria voldaan is, en het eindadres is IPv4-compatibel (32-bit IPv4 adres voorafgegaan door 96 0-bits), dan zal gebruik gemaakt worden van automatische tunneling. Daarbij zijn twee types tunneling te onderscheiden:

- Router-to-host tunneling: indien een router op het pad merkt dat de volgende hop geen IPv6 aankan. Deze zal dan automatische tunneling toepassen tot de eindbestemming.
- Host-to-host tunneling: indien de eerst mogelijke hop na de afzender al geen IPv6 ondersteunt, zal de tunnel van afzender naar bestemming lopen.

Aangezien het aanmaken van een automatische tunnel steeds zo lang mogelijk uitgesteld wordt, zal de tunnel steeds zo kort mogelijk zijn aan de zijde van de afzender. Langs de kant van de ontvanger is de tunnel echter mogelijks suboptimaal: de tunnel wordt altijd gelegd tot aan de eindbestemming van het pakket (bepaald door de 4 laatste

bytes van het IPv4-cmopatibele doeladres), zelfs als er daarbij gebruik gemaakt wordt van subnetwerken die wel IPv6 verkeer aankunnen.

Een ander nadeel is dat het eindpunt steeds IPv4 moet aankunnen, en er zo geen gebruik kan gemaakt worden van de volle 128-bit adresruimte. Deze techniek blijft dan ook beperkt tot kleine netwerkmogevingen.

Geconfigureerde IPv6 naar IPv4 tunneling Bij deze vorm van tunneling worden routers geconfigureerd om bepaalde IPv6 pakketten die over een IPv4 netwerk moeten de versturen via een expliciet geconfigureerde tunnel. Deze tunnel manifesteert zich meestal als een aparte interface, waarbij het eindpunt steeds vastligt. Hierdoor kan een datagram onderweg verschillende keren getunneld als opnieuw gerout worden, afhankelijk van de subnetwerken waarlangs het moet.

Het grote voordeel van deze vorm van tunneling is dat de eindpunten niet IPv4-compatibel moeten zijn, en dus de hele IPv6 adresruimte bruikbaar is.

Hoewel hierbij *router-to-router* tunneling het meest voorkomt, kan het beginpunt van de tunnel ook een standalone IPv6 knoopppunt zijn waardoor *host-to-router* tunneling bekomen wordt. Om deze vorm van tunneling op ISP niveau te implementeren, zou voor elk koppel IPv6 sites minstens 1 expliciet-geconfigureerde tunnel moeten bestaan. Aangezien dit heel wat manuele configuratie vergt, is het te verwachten dat ISPs zullen zorgen voor een IPv6 routing infrastructuur die connectiviteit met alle andere IPv6 netwerken omvat. Hierbij zullen aparte routers slechts 1 router-to-router tunnel moeten configureren om connectiviteit te bekomen.

6to4 tunneling Dit is een speciale vorm van geconfigureerde tunneling, waarbij het IPv4 eindpunt van de tunnel eenduidig kan afgeleid worden uit het IPv6 adres. Daartoe wordt een subset van de IPv6 adresruimte gereserveerd: 2002:w.x.y.z::/48.

Een 6to4 router van een organisatie is een dual-stack border router, met een publiek IPv4 adres voor de tunnelinterface die de toegewezen 6to4 adresruimte in het internet injecteert. Dezelfde router adverteert de globale 6to4 adresruimte 2002::/16 naar het intranet van de organisatie toe. Het geheel van 6to4 routers verbonden met het internet verzekert communicatiemogelijkheden tussen alle 6to4 knooppunten onderling:

- 6to4 knooppunten om eenzelfde site: kunnen rechtstreeks met elkaar communiceren via de private IPv6 infrastructuur. [1]
- 6to4 knooppunten op verschillende sites: bij afgifte van het IPv6 pakket aan de 6to4 router, detecteert die aan de TLA van 2002 dat een 6to4 tunnel moet opgezet worden. Hiertoe wordt het IPv4 adres uit het doeladres geëxtraheerd, en worden ingekapselde IPv6 pakketten naar die host verstuurd. De ontvangende 6to4 router zal tenslotte de IPv4 header strippen en het pakket in zijn lokaal subnet injecteren.

[1] Is dit niet het geval, dan zal een andere vorm van tunneling moeten toegepast worden, bijvoorbeeld ISATAP.

Hoewel we hier continu spreken over 6to4 routers die een subnet beheren, is dit natuurlijk ook toepasbaar bij individuele hosts.

Om connectiviteit te bekomen tussen 6to4 knooppunten en reguliere IPv6 knooppunten moeten *6to4 relay routers* geconfigureerd worden. Dergelijke dual-stack routers bevatten zowel een regulier IPv6 als een 6to4 adres, waardoor 6to4 routers als default route een 6to4 tunnel naar een dergelijke relay router kunnen opzetten. De relay router zal vervolgens de 6to4 IPv4 header strippen, om tenslotte het IPv6 datagram correct door te sturen.

Het grote nadeel aan deze tunnelingtechniek is dat de routingtabellen hun huidige grootte behouden, en een globale hernummering zal nodig zijn om een reguliere IPv6 adresseringsschema te bekomen.

Intra-Site Automatic Tunnel Addressing Protocol (ISATAP) tunneling Dit mechanisme is gelijkaardig aan 6to4, maar biedt een oplossing voor tunneling tussen twee subnetten van eenzelfde private netwerkinfrastructuur (verschillende IPv6 eilanden binnen eenzelfde site). 6to4 faalt hierbij, aangezien beide subnetten dezelfde 6to4 prefix kennen en dus rechtstreeks met elkaar zouden moeten kunnen communiceren, zonder gebruik van een tunnel.

Om dit te verhelpen wordt het IPv4 eindpunt van de tunnel niet afgeleid van de prefix, maar van de laagste 32 bits van een ISATAP adres. ISATAP adressen zijn samengesteld uit een willekeurige globale unicast prefix (64 bit) en een interface-id van de vorm 0:5EFE:w.x.y.z (met w.x.y.z zijnde een willekeurig IPv4 adres dat aan de interface is toegekend).

Het voordeel hiervan is dat ISATAP eenvoudig kan gecombineerd worden met 6to4: de eerste 64 bits van het adres worden bepaald via 6to4, terwijl het interface-id gegenereerd wordt met behulp van ISATAP.

ISATAP knooppunten zijn steeds dual-stack. Communicatie tussen knooppunten verloopt nu als volgt:

- Tussen twee knooppunten van dezelfde site (die niet op hetzelfde IPv6 subnet zitten): hier wordt host-to-host tunneling toegepast, indien ze gebruik maken van ISATAP adressen op basis van de linklokale prefix FE80::/64.
- Tussen twee knooppunten van verschillende sites: hiervoor is extra configuratie benodigd.
  - Knooppunt moet een globale unicast prefix ontvangen: dit kan via uitwisseling van *Router Solicitation* en *Router Advertisement* berichten met de border router, eventueel getunneld indien die zich op gescheiden IPv6 eilanden bevinden.
  - Default route binnen de routetabel van het knooppunt: deze route moet zorgen dat berichten bestemd voor buiten de site, afgeleverd worden aan de border router (hetzij via IPv6 routing, hetzij via ISATAP tunneling).