Glen De Scheerder Kevin Verfaille

Modelvragen reeks D

D1. OSPF (§1.8.2 & §1.8.3) -> reeks D 1

a. Bespreek de noodzaak, het doel en de identificatie van *OSPF area's*. Bespreek de verschillende type area's en de eventuele OSPF routers met specifieke functies die hiertoe ingesteld moeten worden.
b. Beschrijf de uitwerking van het *algoritme van Dijkstra*.
c. In welke opzichten is de door OSPF routers gehanteerde procedure, op basis van het algoritme van Dijkstra, fundamenteel verschillend van de gevolgde werkwijze door een verzameling bridges, die diverse netwerksegmenten tot één enkel subnetwerk groeperen ?
d. Pas het algoritme van Dijkstra toe om de routingtabel op te stellen van router X in het internetwerk van volgende figuur ...

A)
Grotere internetwerken → RIP niet toepasbaar → gebruik van OSPF.
Toch 3 factoren die maximale grootte belemmeren:

- Uitwisselen van LSAs → belasting v/h netwerk
- Compileren LSDBs → veel geheugengebruik van routers (want alle LSAs van alle routers in het routingdomein in LSDB)
- Berekening SPF Tree met Dijkstra → processor intensief

Impact van factoren, kwadratisch met grootte van routing domein.
Elke AS → opdelen in routing domeinen → OSPF routing domeinen indelen in OSPF areas.
Hierdoor, impact 3 factoren beperken (LSA, update pakketten,… beperkt tot grens area).

Areas (elk met eigen LSDB) geïdentificeerd door 32bit getal (*area ID*) in dotted decimal notatie, meestal eenvoudige opeenvolgende nummering genomen.
Area ID in elk Hello-pakket. Hello-pakketen met ander area ID dan ontvangende interface worden genegeerd.
Elk routing domein → *backbone area* (0.0.0.0, daarom ook naam area 0), elke aanvullende area verbonden met backbone area → alle verkeer tussen 2 areas via backbone area → meestal 1 netwerk met grote bandbreedte als backbone.

Meerdere areas → **A**rea **B**order **R**outer**s** → maken deel uit van verschillende areas, bevatten voor elke area LSA, LSDB en SPF Tree. ABR steeds deel van area 0. Vermelden in LSAs ook alle routes buiten area of zelfs buiten domein (extrenal LSAs) die rechtstreeks of onrechtstreeks via hen bereikbaar zijn → aggregatie (sumary LSAs) → beperkt route flapping.
LSDB area 0: gedetailleerde LSAs backbone zelf als summary LSAs andere areas.
Soms reductie vdoor ABR tot één enkele default route LSA (cte metriek) → injecteert deze in area (=stub area) die verbonden is met backbone (=transit area). In praktijk enkel als één ABR stub area met backbone verbindt of om het even welke ABR kan gebruikt worden om area externe routers te bereiken.
Creatie stub areas → interfaces ABRs configureren.
Voordeel stub area: minimale LSDB, nadeel:functionaliteit beperkt (geen ASBRs in area).
Stub areas kunnen geen rol vervullen als transit area.

Virtual links tussen twee routers van dezelfde intermediaire area → verplichting fysieke verbinding met backbone afgezwakt, wel één router met backbone verbonden. Gebruik enkel virtual links als praktisch niet mogelijk is voor fysieke connectie.

B)Doel: Op basis van de internetwerktopologie een spanning tree (OSPF Tree) te construeren met de routers als knopen.
Geg: figuur met routers en verbindingen tussen routers (netwerken) met een cijfer bij elk netwerk dat de kost voorstelt om ervan gebruik te maken. LSDB van het AS.
Gevr: bereken SPF tree voor router X en haalt daaruit routingtabel van router X
Opl:

- Start tree met X als wortel
- LSA van X om kandidaat subknopen te bekomen → elke subknoop krijgt waarde van knoop erboven, vermeerderd met kost van verbinding
- Iteratie 3 stappen:
    1. Van alle niet geselecteerde subknopen → neem deze met laagste cijfer op in SPF Tree
    2. Uitbreiden met kandidaat subknopen op basis van overblijvende LSAs van de corresponderende router (routes al in SPF Tree niet in rekening)
    3. Indien routers meerdere keren → verwijder hoogste waarde.
- Iteratiestop als alle routers effectief in SPF Tree zijn opgenomen.
- Dan nog niet alle netwerken opgenomen (1 minder dan # routers) → te vinden door op deze netwerken routers op te sporen met de minimale kost.

C) Bij een bridge verkiest men een rootbridge en dan gaat elke niet rootbridge de kortste weg naar de rootbridge gaan bepalen. De poort met de kleinste kost, wordt dan de rootport van die bridge. Dan gaat elke bridge kijken welke poort hij andere bridges zal aanspreken om een bepaald subnetwerk te bereiken (die met de laagste kostprijs), dit zijn dan de designated poorten. Elke poort die niet designated of root is wordt geblokkeerd.

Bij een OSPF gaat elke router melden aan op welke netwerken hij rechtstreeks is aangesloten via LSA's (Link State Advertisements). Deze berichten worden verstuurd naar alle buren die het ook opnieuw doorsturen naar hun buren(flooding). Elke router houdt hiervan een inventaris (router-subnetwerk) bij in een LSDB (link state database). Eens die compleet is, kan de router op basis hiervan zijn volledige routetabel gaan opstellen, hiervoor stelt hij een spanning tree op met algoritme van Dijkstra. Daaruit kan hij dan alle routes naar de beschikbare routes gaan bepalen.

We zien dus dat bridges zelf hun informatie opzoeken/nagaan/berekenen en dit enkel bij hun buren. Terwijl bij OSPF men gebruik maakt van wat alle andere routers doorgeven van informatie. Toch is dit enigszins beperkt bij OSPF omdat men daar per subnetwerk met een designated router werkt. Het komt er dan op neer dat wanneer een router een wijziging detecteert, deze dit enkel zal melden aan de designated router, die het dan op zijn beurt doorgeeft aan zijn buren. Op deze manier wordt er veel minder netwerkbelasting voor administratie veroorzaakt en dient de herberekening enkel te gebeuren indien er effectief iets wijzigt in de opstelling. Ook zie je bij OSPF dat elke router zelf een spanning tree gaat opbouwen, terwijl bij bridging er eigenlijk een spanning tree gecreëerd wordt op een hoger niveau en er geen zelf berekend.

D) oefening op **B)**

Glen De Scheerder Kevin Verfaille

D2. DNS

a. Geef een overzicht van de belangrijkste DNS *concepten* en *terminologie*. (§3.1)
b. Bespreek op welke niveaus DNS van *caching* gebruik maakt. (§3.1 & figuur 3.2)
c. Wat wordt beoogd met *reverse DNS* ? Hoe wordt dit gerealiseerd ? (§3.3.2)
d. Bespreek de *integratie* DHCP/DNS. (§2.1.4)
e. Bespreek hetzij de diagnoseopdracht *nslookup,* hetzij de diagnoseopdracht dig (doel, opties,parameters, output,interactief gebruik,…) (§3.5)

A)

Concepten:

- Domain **N**ame **S**ystem → namen gebruiken ipv IP-adressen. Namen zijn uniek en bestaan uit alfanumerieke gedeelten gescheiden door een punt. Hiërarchisch, hoogste niveau achteraan, eerste gedeelte naam van computer, rest duid een domein aan.
- Elke organisatie kan zelf zijn naamstructuur kiezen → geen rekening met topologie,…
- Gebruikt een gedistribueerde databank: namen van alle computers verdeeld over verschillende primaire nameservers (elke organisatie draait en onderhoudt eigen nameservers en kan zelfstandig beslissen subdomeinen te creëren).
- Nameservers houden DNS-records van verschillende types bij (ip-adressen, info over zone, mail exchange, …). Elke nameserver bevat ook nodige informatie die de nameserver koppelt aan andere nameservers, en hoe deze kunnen bereikt worden.
- Naamresolutie:
    1. computer spreekt willekeurige DNS-server (v/h domein waartoe hij behoort of op het netwerk) aan, DNS-server is verplicht te antwoorden met IP-adres bestemming of foutmelding, als de DNS-server geen autoriteit is voor de zone waartoe de op te zoeken DNS-naam behoort moet hij beroep doen op andere nameservers.
    2. DNS-server weet niet zomaar wie autoriteit is voor het opgegeven domein dus hij zendt aanvraag naar de root-servers.
    3. DNS-server zal vervolgens doorgestuurd worden naar andere nameservers die telkens autoriteit zijn voor een zone, dieper in de domeinhiërarchie.
    4. Dit iteratief doorsturen stopt als autoritiet van op te zoeken zone (waartoe op te zoeken DNS-naam behoort) gevonden is.
- Secundaire nameservers: houden zelfde info bij als de primaire nameservers en verlichten hun taak. Als primaire uitvalt kan secundaire gepromoveerd worden. Secundaire zoekt regelmatig contact met primaire om up te daten (want wijzigingen enkel bij primaire!).

Terminologie:

Domein: een volledige groep computers. Elk domein maakt deel uit v/ precies één zone.
Root-domein: ".", domein van het hoogste niveau → daaronder top-level domeinen (.be, .com, …).
Zone: groepering van domeinen die door dezelfde nameserver worden beheerd. Een zone kan meerdere domeinen bevatten.
Nameserver: computer die autoriteit is voor één of meerdere zones. (autoriteit voor root en belangrijke top-level domeinen (gov, arpa, mil, org, net, …) zijn de root-servers).
Naamresolutie: proces waarbij een willekeurige computer de naam van een andere computer poogt om te zetten naat het overeenkomstig IP-adres.
Recursiviteit: proces van het contacteren van opeenvolgende nameservers door de DNS-server, met de bedoeling een IP-adres te vinden.

B)
Om recursiviteit te versnellen → nameservers houden naam en ip-adres van aanvragen bij, ook van aanvragen zonder resultaat (negatieve caching). Op alle records in de cache is ttl-waarde van toepassing → hoe lang bewaren.

Cliënt: DNS-resolver cache. Wordt eerst geraadpleegd vooraleer DNS-aanvraag te verzenden naar DNS-server. Afkomstig uit 2 bronnen:

1. bij opstarten DNS Client Service wordt alles uit \windows\system32\drivers\etc\hosts bestand in cache geladen (enkel als bestand geconfigureerd is).
2. van eerdere aanvragen bij DNS-server (zit in DNS-servercache).

Server: DNS-server cache. Aangevuld door resultaten van aanvragen aan deze DNS-server.

C)
Reverse DNS = domeinnaam opzoeken voor een gegeven IP-adres. Meestal om beviligingscontrole uit te voeren. De DNS-records zijn van het type PTR.

Naam w.x.y.z? PTR-record opvragen van autoritaire server die deze informatie bevat, kennen domeinnaam machine niet → weten niet waar deze server te zoeken. Invoeren pseudo-domein: in-addr.arpa. Elk netwerkadres heeft een geassocieerde naam in dit speciale domein, naam = z.y.x.w.in-addr.arpa. We zoeken nu dus PTR record van deze naam. Hiërachische structuur van domein in-addr.arpa komt ongeveer overeen met die v/h netwerk.
Reverse DNS toelaten op machines eigen netwerk → primaire nameserver configureren voor overeenkomstige subdomein in-addr.arpa domein.
→ Vb in cursus: toestellen iii.hogent.be allemaal in 192.168.16/24
#cat /var/named/192.168.16
SOA-record hetzelfde (behalve serienummer), PTR records. Alle namen in bestadn relatief t.o.v. 16.168.192.in-addr.arpa .Getallen in linkerkolom PTR records duiden op ip-adressen (relatief), namen rechterkant zijn absoluut.
→ reverse DNS voor deel v/h loopbacknetwerk 127/8
#cat /var/named/named.local
kunnen ongewijzigd zonebestand gebruiken dat standaard tijdens de installatie van BIND software geconfigureerd wordt.

```
@       IN SOA localhost. Root.localhost. (
                …
                )
        IN NS localhost.
1       IN PTR localhost.
```

D)
Telkens niet-geregistreerde computer vrij adres uit DHCP-scope krijgt → configuratie van de DNS nameserver van de zone waartoe deze computer behoort aanpassen. Vroeger manueel, nu via dynamische updates. Tabel met verbanden DNS-naam/ip-adres telkens aangepast indien nodig → alle naam/adres toewijzingen van de computer blijven op een eenvoudige manier steeds gesynchroniseerd. Opm: oude machines die geen weet hebben van dynamische updates weten geen raad met deze info.
Dynamische updates kunnen door cliënt verzonden worden. Meeste DNS-cliënts kunnen echter niet rechtstreeks dynamische updates bij DNS nameserver van hun zone aanvragen → DHCP-server zodanig geconfigureerd dat ze voor hun DHCP-cliënts op DNS-nameservers, die dynamische updates ondersteunen, wijzigingen uitvoeren (zowel A als PTR records). Hiervoor wordt optie 81 gebruikt → cliënt geeft **F**ully **Q**ualified **D**omain **N**ame door aan DHCP-server + instructies voor verwerking dynamische DNS-updates.
Vroegere versie: DHCP-cliënt zendt aanvraag IP-lease, DHCP-server antwoord met bevestiging, DHCP-server zendt DNS dynamische updates van A- en PTR-record.
Nu, windows 2000, XP: DHCP-cliënt zendt aanvraag IP-lease, DHCP-server antwoord met bevestiging, DHCP-cliënt zendt dynamische update A-record, DHCP-server zendt dynamische update van PTR-record.

E)
nslookup
Doel: DNS informatie opvragen bij een willekeurige DNS-server. Ook handig om rechtstreeks allerhande informatie via het DNS-systeem te weten te komen.

Opties, parameters, output:

- **computernaam**: opvragen IP-adres computer. *Output*: naam en ip adres van computer (ook van gebruikte DNS-server).
- **computernaam + server**: analoog
- **-query=MX + naam**: opzoeken van MX-record voor de opgegeven naam. *Output*: naam en ip gebruikte DNS-server, ale MX-records voor de naam
- **-type=PTR ip-adres**: reverse DNS-lookup naar IP-adres. *Output*: naam en ip gebruikte DNS-server, naam computer, nameservers domein, ip-adressen nameservers
- **-d2**: ook via >set d2: schakel debug modus in. Vooral gebruikt als je het gedrag wil laten simuleren van een nameserver en niet zoals default ingesteld van de lokale resolver.
- **-norec** en **–nosearch**: ook via >set norec en >set nosearch: iteratieve aanvragen ipv recursieve

Interactief: Programma oproepen zonder argumenten, opdrachten na >-promt opgeven. Nslookup contacteert voor alle opdrachten slechts één DNS-server, de eerste server in een nameserver-lijn die op de eerste aanvraag van de interactieve sessie een antwoord terug stuurt.

> set type=NS
> hogent.be
Vraag aan default ingestelde DNS-server v/d werkpost wie volgens hem nameserver is voor het domein hogent.be. *Output*: naam en ip gebruikte DNS-server, naam en ip gevonden nameserver. Non-authorative answer → antwoord uit cache gehaald.

> server elvis.hogent.be
We gebruiken deze nameserver voor volgende opdrachten. *Output*: ip en naam DNS-server. Alternatief is lserver: zoekt ip-adres op via default DNS-server <> server zoekt ip-adres opgegeven nameserver via huidige nameserver.

> set type=A
> gonzo
Ip-adres willekeurige computer achterhalen. *Output*: naam en ip-adres gebruikte DNS-server en computer.

> ls –t Mx hogent.be, > ls –t ANY hogent.be > hogent.be
ls simuleert zone transfers en toont alle records van een specifiek type. *Output*: naam gebruikte DNS-server, gevonden records. Ook mogelijk output om te leiden naar willekeurig bestand (bekijken via view opdracht).

> exit
of control-d: programma stoppen.

> set rec, >set search, >set nod2
default gedrag van nslookup herstellen, debug modus uitzetten

> help
help ivm nslookup

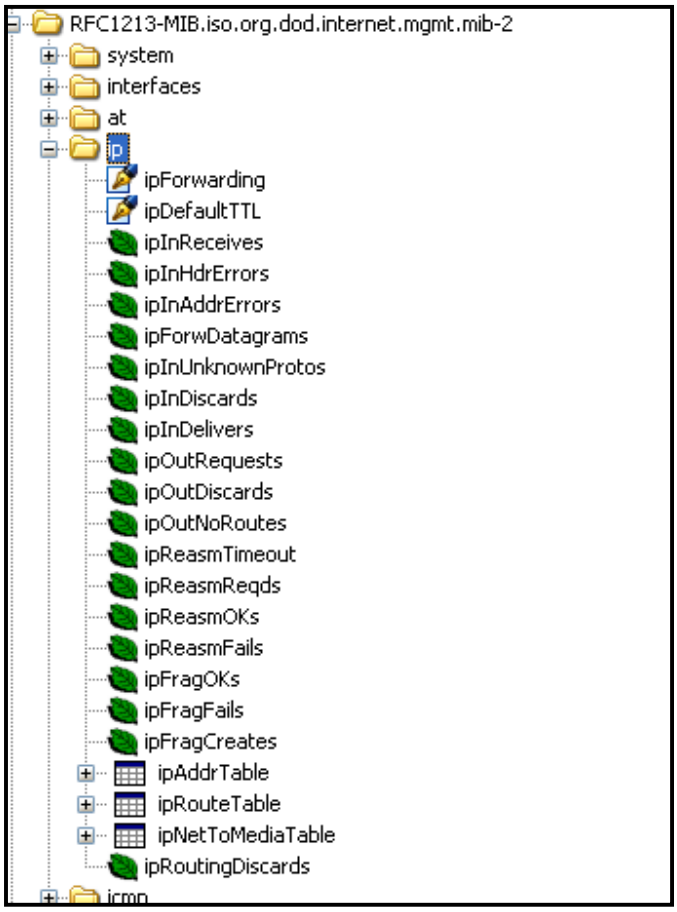
Opm:
Zowel op Linux als Windows standaardsoftware en bijna dezelfde syntax, output.
Bij opstarten foutboodschap: Can't find server name → nslookup wil reverse DNS uitvoeren op IP-adres van ingestelde DNS-server, soms lukt dit niet. Vermijden door reverse lookup zone in te stellen met PTR records voor alle DNS-servers.

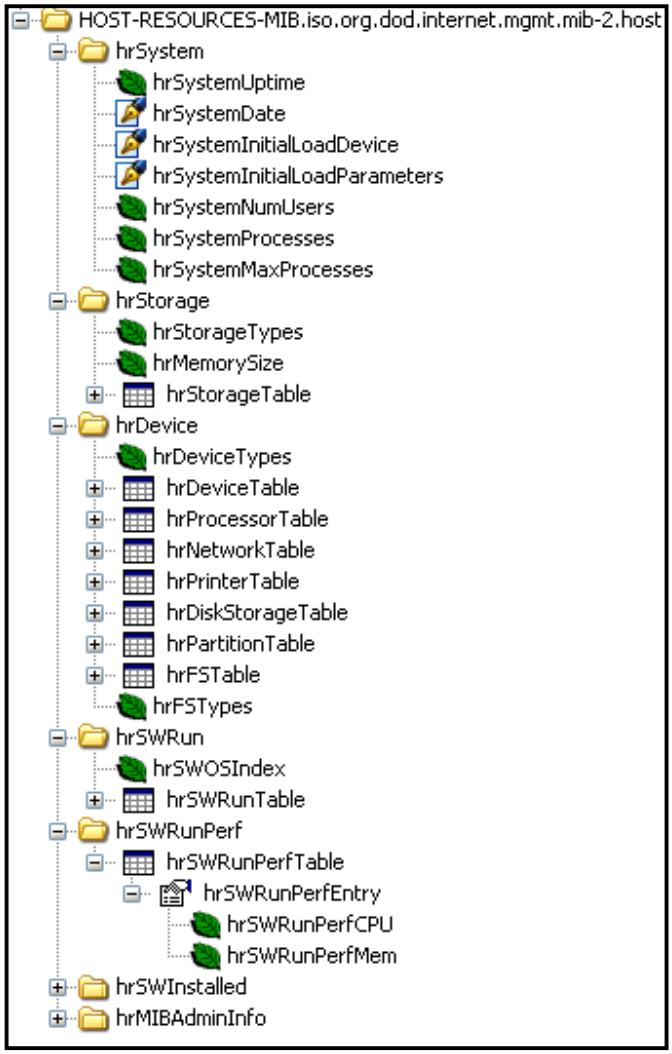
D3. SNMP protocol

a. Geef de ASN.1 codering van een [...] SNMP bericht, en een korte uitleg bij elke TLV.
b. Bepaal de inhoud van de (hexadecimale) dump van een [...] bericht, dat tegelijkertijd [...], [...] en [...] vraagt/meldt.

A)
Conform vraag C1 Die geschrapt was … zie oplossingen reeks C (versie met 2 pdf's)

Wel handig ()mib2ip en mib resources moet je kennen) :





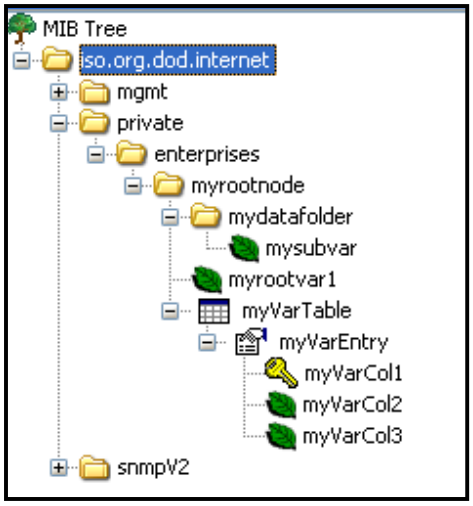
B) Is een oef => zie opgeloste oefeningen labo op gonzo

D4. *SMI*

Als kersvers ondernemer verover je de markt met een *SNMP manageable*apparaat dat toelaat om de toestand van [*een gebruiksvoorwerp*] te achterhalen. Jouw SNMP agent is ondermeer in staat om [*detailfunctionaliteit*].

a. Ontwerp (en teken) een boomstructuur van de knooppunten en objecten (inclusief identificatie) die door jouw agent geïmplementeerd worden. Zorg ervoor dat jouw boomstructuur ingepast wordt in de wereldwijde MIB structuur. Maak zo efficiënt mogelijk gebruik van tabelobjecten. Gebruik er in ieder geval minstens één.
b. Definieer uw volledige MIB met behulp van SMI syntax.

Ik heb hier gewoon een voorbeeld tree gemaakt en gedocumenteerd wat voor wat staat.




```
NET-SNMP-EXAMPLES-MIB DEFINITIONS ::= BEGIN

IMPORTS
MODULE-IDENTITY, OBJECT-TYPE, Integer32 FROM SNMPv2-SMI;

--De folder die in deze file beschreven wordt normaal naam van je bedrijf--
myrootnode MODULE-IDENTITY
    LAST-UPDATED "200202060000Z"     --20/06/2002--
    ORGANIZATION "companyName"
    CONTACT-INFO
       "postal:   Wes Hardaker
                    P.O. Box 382
                    Davis CA  95617
        email:    net-snmp-coders@lists.sourceforge.net"
    DESCRIPTION
        "Example MIB objects for agent module example implementations"
    REVISION     "200202060000Z"     --versienr … hier date--
    DESCRIPTION
      "First draft"
   ::= { enterprises 1 }           --onze rootnode onder Enterprises zetten



--next page --
```

```
--zorgen dat we op het juiste niveau zitten voor de bedrijf--
enterprises OBJECT IDENTIFIER ::= { iso org(3) dod(6) internet(1) private(4)
1 }
mydatafolder OBJECT IDENTIFIER ::= { myrootnode 1 }

-------------------------------------------------------------------------
----Er is dus 1 identifier om op het enterprise niveau te raken en dan   ----
----nog 1 extra subfolder ter demo, deze plaatsen we onder de myrootnode ----
----waarvoor deze file gemaakt is.                                       ----
-------------------------------------------------------------------------

--Vars onder rootnode--
myrootvar1 OBJECT-TYPE
    SYNTAX      OCTET STRING
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
      "Een variabele of hoofdniveau"
    ::= { myrootnode 2 }  --onder rootnode 2de plaats, subfolder staat op 1--

--Vars in subfolder mydataFolder--
mysubvar OBJECT-TYPE
    SYNTAX      OCTET STRING
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
      "Een variabele in een subfolder"
    ::= { mydatafolder 1 } -- dit zit onder mydatafolder 1 ste plaats --

-------------------------------------------------------------------------
----Nu gaan we en demotabel aanmaken. Hiervoor heb je 4 objecten nodig:  ----
----   1)Tabelobject die je onder de root zet, bestaat uit rijen         ----
----   2)Een rij moet je zeggen wat erin zit (definitie)                 ----
----   3)Dan moet je een rij object hebben, zodat je er kolommen kan     ----
----     kan aanhangen (specificatie)                                    ----
----   4)Dan moet je nog je kolom object specifiëren                     ----
-------------------------------------------------------------------------

--tabel op rootniveau--
myVarTable OBJECT-TYPE
      SYNTAX SEQUENCE OF MyVarEntry
      ACCESS not-accessible
      STATUS mandatory
      DESCRIPTION
            "De beschrijving van mijn tabel"
      ::= { myrootnode 3 }

--rij moet gedfinieerd worden waaruit hij bestaat--
MyVarEntry ::=
      SEQUENCE {
            myVarCol1   INTEGER,
            myVarCol2   INTEGER,
            myVarCol3   OCTET STRING
}
```

```
--daarnaast heb je nog een rijobject nodig (specificatie)--
myVarEntry OBJECT-TYPE
      SYNTAX MyVarEntry
      ACCESS not-accessible
      STATUS mandatory
      DESCRIPTION
            "mijn info over een rij"
      INDEX { myVarCol1 }
::= { myVarTable 1 }

--kolomobjecten voorzien op rijobject--
myVarCol1 OBJECT-TYPE
      SYNTAX INTEGER
      ACCESS read-only
      STATUS mandatory
      DESCRIPTION
            "Beschrijving kolom1"
      ::= { myVarEntry 1 }

myVarCol2 OBJECT-TYPE
      SYNTAX INTEGER
      ACCESS read-only
      STATUS mandatory
      DESCRIPTION
            "Beschrijving kolom2"
      ::= { myVarEntry 2 }

myVarCol3 OBJECT-TYPE
      SYNTAX OCTET STRING
      ACCESS read-only
      STATUS mandatory
      DESCRIPTION
            "Beschrijving kolom3"
      ::= { myVarEntry 3 }

myTrap TRAP-TYPE      -- ook nog trap ter demo (pg121)
      ENTERPRISE myrootnode
      VARIABLES { myrootvar1, mysubvar }
      DESCRIPTION
            "Een trap is opgetreden, hier geef ik ter demo naast een foutnr
(26) ook nog eens 2 vars mee, die de client dan moet opvragen om de inhoud te
kennen.
            in praktijk maak je dus trapvariabelen, waar je foutwaarden kan
in bewaren en die de client dan kan op vragen"
      ::= 26

END
```

Verder moet je gewoon ALLES van mogelijke waarden blokken op pg 112-115 om te weten wat de opties zijn. Heeft geen nu om hier alles onder te plakken, maar je zal zien dat het nog meevalt … .

Wat kan er staan bij status, syntax … of whatever. In bovenstaant voorbeeld heb ik gebruik gemaakt van basisobject zoals INTEGER, OCTET STRING maar dat kunnen ook objecten zijn zoals vb IpAddress, Counter … of zelfs eigen object dat je dan uiteraard nog dient aan te maken.

D5. SNM protocol interacties

a. Beschrijf de belangrijkste *componenten* van de SNMP standaard (in de ruime zin van het woord).
b. Met welke (minimaal aantal) interacties kan je in SNMPv1 de velden [...], [...] en [...] (en enkel deze velden) van alle rijen van het [*tabelobject*] van een netwerkcomponent te weten komen ? Beschrijf de wisselwerking tussen NMS en agent, waarbij je bij elke vraag en antwoord zowel het type van het bericht als de inhoud van de*varBindList* geeft (je hoeft geen hexadecimale dump te produceren).
c. Geef een antwoord op dezelfde vraag, maar nu met behulp van SNMPv2. Zorg opnieuw voor een minimaal aantal interacties.
d. Geef een kort overzicht van de verbeteringen van SNMPv2 ten opzichte van SNMPv1. Beperk je hierbij tot het protocol aspect van SNMP.

A)

SNMP (Simple Network Management). Richt zich in de eerste instantie op het beheer van alle toestellen met IP als transportprotocol. Er zijn 2 soorten entiteiten:

- *SNMP agent:*
  het is software dat uitgevoerd wordt op alle te beheren netwerkcomponenten. Het houdt alle noodzakelijke informatie bij om de operationele toestand, de performantie en de configuratie van een toestel volledig te beschrijven.
  Een agent kan een programma zijn of ingebouwd zijn in het besturingssysteem van het toestel.
- *SNMP managers:*
  ook NMS (Network Management Stations) genoemd.
  Ze ondersteunen software (NMA's, Network Management Applications) die voor het gehele netwerk beheerstaken kunnen uitvoeren.
  Het is het controlecentrum van alle netwerkbeheer activiteiten. Het zorgt zowel voor de interface met de operatoren en netwerkbeheerders als met de agenten.
  Het is verantwoordelijk voor het opvragen en ontvangen van informatie aan alle SNMP agenten in het netwerk, en voor het uitvoeren van acties op basis van de ontvangen informatie.
  Agenten (=server) en managers (= client)

SNMP operaties (= interactie tussen NMS en agenten)

- *Get request* of *Poll:* is de meest uigevoerde basisoperatie. Een NMS kan een agent vragen naar specifieke informatie. Periodiek opvragen van informatie aan verschillende agenten noemt men *externe polling* (*interne polling* = software van agent die periodiek toestand vraagt van component). Meestal kan men de NMS instellen zodat er een actie gebeurt al er een drempel wordt overschrijden.
- *Set request:* via de agent de toestand van bepaalde instellingen van een netwerkcomponent wijzigen.
- *trap:* manier waarom agent NMS verwittigt dat er iets aan de hand is. Worden asychroon opgestuurd en is geen reactie op een vraag van de NMS.
  Indeling traps:
  - 0: coldstart
  - 1: warmstart
  - 2: linkdown
  - 3: linkup
  - 4: authenticationfailure
  - 5: egpneighborloss
  - 6: enterprisespecific

  Reactie op *generische traps* (0 tot 5) zijn meestal hardgecodeerd in de software van de NMS. *Specifieke traps* bevatten aanvullende informatie (identificatie van organisatie die trap maakte en specifiek trap nummer gegeven door organisatie). Er kunnen ook aanvullende identificaties van objecten en hun waarden opgestuurd worden.

  Polls en traps kunnen onafhankelijk van elkaar verstuurd worden. Er bestaan geen andere operaties dan de 3 vermelde. Het is wel mogelijk om interne opdrachten van de agent te koppelen aan een

toestand. Met een set request kan men die toestand veranderen waardoor een agent procedure wordt uitgevoerd. (zo kan men een netwerktoestel rebooten)

Agent bezit een lijst van gegevens over de netwerkcomponent. De NMS kan deze raadplegen deze raadplegen.

- *Structure of Management Information (SMI):* laat toe om objecten, die netwerkelementen beschrijven, en hun gedrag te definiëren.
- *Management Information Base (MIB):* is een databank op basis van de SMI-syntax van alle objecten die een agent bijhoud. Het legt een symbolische naam vast voor het object en geeft de betekenis ervan weer. De NMS bevat een kopie van de MIB gegevens van de agent, daardoor kan deze een correct geformatteerde en gecodeerde vraag stellen aan de agent omtrent zijn informatie.

De NMS moet voor elk netwerkcomponent die hij moet beheren de MIB laden in een *Manageable Objects Database (MDB)*. Die functioneert als de verzameling vragen die aan een agent kunnen gesteld worden. Het heeft geen zin de MIB's te laden voor elementen waarvoor geen SNMP agenten beschikbaar zijn.

De SNMP standaard omschrijft ook het SNMP- protocol in de applicatielaag. Dit protocol garandeert dat NMS en agents van willekeurige constructeurs kunnen communiceren. Dit protocol is bewust in de applicatielaag gestopt, omdat zo de IP laag er voor zorgt dat NMSen en agents zich in verschillende hetrogene netwerken kunnen bevinden.
SNMP maakt gebruik van het UDP protocol en werkt op poorten 161 (send recieve request) en 162 (traps). Het voordeel van UDP is dat de implementatie eenvoudiger is voor de low-level netwerkcomponenten en ook omdat dit voor een kleine impact zorgt op het netwerk dan TCP. Het nadeel dan weer is dat UDP pakketten verloren kunnen gaan en het SNMP onbetrouwbaar maakt.
Dit valt deels op te lossen door de agent verantwoordelijk te stellen voor het detecteren van verlies (vb time out). Maar voor traps gaat dit niet op. Een trap die verloren gaat komt nooit aan en de agent verwacht geen antwoord, dus weet geen van beide of de transmissie gelukt is.

B)

1. Je stuurt een GetNextRequest met velden uit de kolom die je wenst
2. Je krijgt bericht terug met de OID en Waardes
3. Er wordt opnieuw GetNextRequest met oid gegeneerd (je krijgt dan volgende rij terug)

Voor elke rij uit de tabel dienen we minimaal 2 berichten te gebruiken … en GetNextRequest en een GetResponse bericht.

C)

1. Je stuurt een GetBulkRequest met velden uit de kolom die je wenst
2. Je krijgt bericht terug met de OID en Waardes van x - rijen
3. Er wordt opnieuw GetBulkRequest met oid gegeneerd (je krijgt dan volgende rijen terug)

Voor elke x rijen uit de tabel dienen we minimaal 2 berichten te gebruiken … en GetBulkRequest en een GetResponse bericht.

D)

Daar waar SNMPv1 te kort kwam om veel data in 1 keer op te vragen, onder de vorm van objecten, tabellen, biedt SNMPv2 oplossingen voor dit probleem. Bij SNMPv1 kan men meerdere objecten per request bericht gaan opvrragen, maar indien het pakket/antwoord te groot is voor udp/verweking wordt er geantwoord met foutboodschap en verder niets. Een tabel volledig overlopen gaat relatief goed met de GetNextRequest/GetResponse berichten, maar hiervoor zijn voor elke cel BEIDE berichten nodig, wat voor een serieuze overhead zorgt.

In SNMPv2 daarentegen is het nu mogelijk om te werken met het GetBulkRequest bericht, die toelaten om een deel van een tabel ineens op te vragen. GetBulkRequest wordt beantwoord met een GetResponse bericht met zoveel mogelijk data en hier is wel onvolledige data toegelaten. De structuur voor een GetBulkRequest is identiek aan GetNextRequest bericht:

```
GetBulkRequestMessage ::= SEQUENCE {
        version    INTEGER (0),
        community OCTET STRING,
        protocolDataUnit SEQUENCE {
                requestID       INTEGER,
                nonRepeaters    INTEGER,
                maxRepetitions INTEGER,
                varBindList     SEQUENCE OF {
                        varBind SEQUENCE {
                                name  OBJECT IDENTIFIER,
                                value ObjectSyntax } } } }
```

Hier zijn wel TLV's (Overbodige: errorStatus, errorIndex) vervangen.
nonRepeaters Het aantal TLV's (die vooraan moeten zitten) waarvoor geen iteratie nodig is
maxRepetitions Bepaalt hoeveel keer dezelfde vraag (na nonrepeaters) moet herhaald worden

Meestal geeft men bij de TLV's dan kolomobjecten op en bij de maxrepetitions geeft men dan het aantal rijen op dat men wil opvragen. Zijn er meer maxRepetitions dan er rijen zijn dan krijgt men kolominformatie waarin men waarschijnlijk niet geïnteresseerd is.

Verder biedt SNMPv2 ook een SNMPv2Trap en een InformRequest bericht.aan. Deze laatste maakt interactie tussen verschillende NMSen mogelijk (Network Management Stations).

Ook beschikt SNMPv2 over meer (13 extra) precieze foutboodschappen.

Glen De Scheerder Kevin Verfaille

D6. Neighbor Discovery (§5.5 & §5.6)

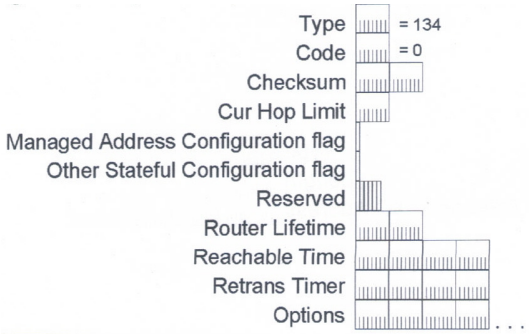
a. Bespreek de structuur van Neighbor Discovery *berichten*. Vermeld alle velden die een rol spelen bij *stateless autoconfiguratie* of één van de Neighbor Discovery processen.
b. Bespreek in detail de verschillende Neighbor Discovery *processen*.

A)

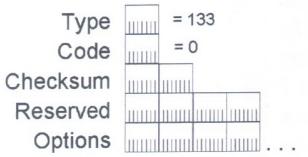
Neighbor Discovery: een verzameling ICMPv6 berichten en overeenkomstige processen die gebruikt worden om informatie te verkrijgen omtrent naburige knooppunten van hetzelfde subnetwerk.
Stateless autoconfiguratie: biedt aan individuele knooppunten de mogelijkheid hun IPv6 configuratie te bepalen, zonder dat ze expliciet een server moeten ondervragen die informatie over elk knooppunt bezit.

Structuur ND-berichten:
Er zijn 5 soorten ND-berichten die allemaal de structuur volgen van andere ICMPv6 berichten. Het datagebied bevat zowel gegevens die uniek zijn voor de individuele berichten, als informatie die in meerdere ND berichten terugkomen. Daarom wordt het datagebied in 2 verdeeld (elk deel een veelvoud van 8bytes):

- **header:** bevat *type*, *code*, *checksum* en *specifieke informatie* over het type van het bericht
  - **Router Advertisement:** indicatie van prefixen, MTU[1] en hop limiet.
    - *Router Lifetime*: heeft aan hoelang de router als default mag ingesteld worden.
    - *Managed Address & Other Stateful Configuration Flag* niet gezet dan zal stateless autoconfiguratie uitgevoerd worden
    - *Retrans Timer:* duid aan om hoeveel miliseconden men *Neighbor Solicitation* berichten moet sturen om bereikbaarheid vast te stellen.
    - *Reachable Timer:* duid geldigheidsperiode aan van ontvangen *Neighbor Advertisements*.



  - **Router Solicitation:** verzoek aan lokale routers om direct een Router Advertisement bericht te versturen.
    - bevat enkel een MAC adres (in optie *Source Link-Layer Adress*)

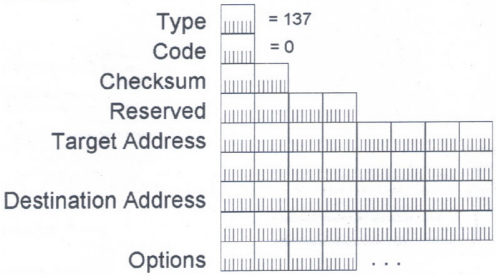


  - **Neighbor Solicitation:** vragen aan een knooppunt naar zijn MAC adres, nagaan of het knooppunt nog bereikbaar is of verifiëren of zijn eigen IPv6 uniek is.
    - *Target Address*: IPv6 adres van de doelknoop

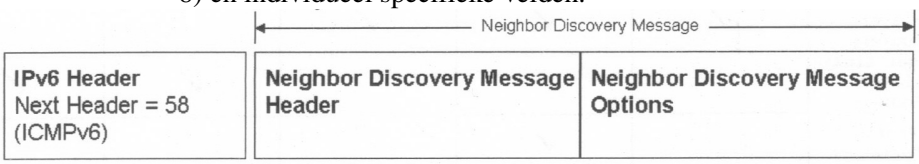
[1] MTU: maximum grootte (in bytes) die verstuurd mogen worden

- **Neighbor Advertisement:** verstuurt als reactie op Neighbor Solicitation of als MAC adres veranderd is.
  - *Solicited flag*: gezet als het een antwoord is op Neighbor Solicitation bericht.
  - *Router flag:* indicatie of knooppunt router is
  - *Override flag:* aanduiding of ontvanger cache moet aanpassen volgens gegevens van optie Target Link-Layer Address.
  - *Target Address:* bevat IPv6 van zender.

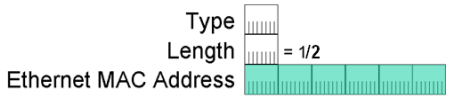
- **Redirect:** naburige knopen vertellen dat ze niet de beste route zijn voor een bepaalde bestemming. Na ontvangst van z'n bericht passen knooppunten hun routingtabel aan.
  - *Destination Address*: doeladres
  - *Target Address*: router voor af te leveren



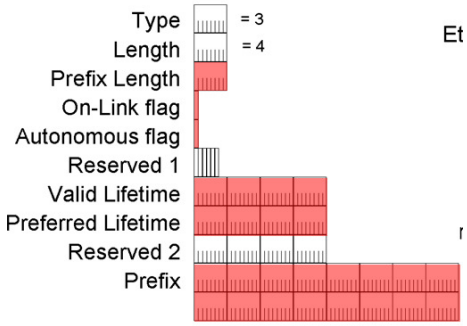
- **opties**: kan 5 verschillende opties bevatten. Ze zijn gecodeerd in TLV formaat. Ze hebben elk een *type* veld (= 1 tot 5), een *length* veld (= totale lengte TLV triplet / 8) en individueel specifieke velden.



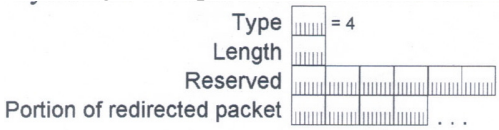
- **Source Link-Layer Address (1):** bevat een ethernet MAC adres
- **Target Link-Layer Address (2):** bevat een ethernet MAC adres



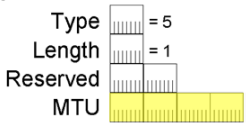
- **Prefix Information (3):** als de *On-Link* vlag op 1 staat, bepaalt de combinatie van *Prefix* en *Prefix Length* velden de prefix van het subnetwerk waarop de interface aangesloten is.



- **Redirected Header (4):** bevat IPv6 datagram dat aan de oorzaak ligt van het ND bericht. Om de totale lengte te beperken kan het een fractie van de datagram bevatten.



- **MTU (5):** bevat aanbevolen MTU voor berichten die buiten het locale subnetwerk moeten gestuurd worden.



B)

Router discovery:

- Gaat op zoek naar de beschikbare routers op een subnetwerk. één van de routers wordt als default ingesteld met een maximale duur = *Router Lifetime* in *Router Advertisement*. Als die niet meer bereikbaar is (= reachability detection) wordt een andere router gekozen.
- Zorgt ook voor parametrisatie van de autoconfiguratie als van andere ND processen.

Na opstarten stuurt elk knooppunt een *Router Solicitation* bericht naar het linklokale *all-routers* multicast adres (= FF02::2). Op Ethernet niveau wordt het doeladres op 33-33-00-00-00-02 gezet en de brongegevens worden ingevuld met de gegevens van de aanvrager. (= om unicast antwoord mogelijk te maken) Als aanvrager nog geen IPv6 adres geeft wordt :: ingesteld.
Na ontvangst van *Router Solicitation* bericht stuurt router een unicast *Router Advertisement* bericht terug. Als aanvrager nog geen IPv6 adres heeft wordt het bericht gemulicast naar het linklokale *all-nodes* multicast adres (= FF02::1) en MAC adres 33-33-00-00-00-01. Gebeurt ook periodiek op initiatief van router.
Op basis van het Router Advertisement bericht kunnen knooppunten hun configuratie bijwerken.

1. **Adres resolutie:**
   Is noodzakelijk om het MAC adres van een bepaald IPv6 adres te achterhalen, om daarna de *neighbor cache* te kunnen aanpassen.
   Aanvrager stuurt *Neighbor Solicitation* bericht naar *solicited-node* multicast adres. Eigen MAC adres ingevuld in optie *Source Link-Layer Address.* Het doeladres, in subnetwerk header, is een mulicast adres. (vorm = 33-33-FF-x-y-z waar x,y en z bepaald worden door laatste 3 bytes van IPv6 doeladres.) Volledige IPv6 adres komt in het *Neighbor Solicitation* bericht.
   Ontvangers van *Neighbor Solicitation* bericht vergelijken IPv6 adres met eigen adres. Ze passen dan de *neighbor cache* aan, met gegevens van aanvrager, en sturen een unicast *Neighbor*

Advertisement bericht terug. Nu kan de aanvrager zijn *neighbor cache* aanvullen met IPV6 en MAC adres.

2. **Duplicate address detection:**
   (bijna identiek aan *adres resolutie*) het *Target Address* (in *Neighbor Solicitation* bericht) is het adres dat men wil controleren. het bronadres is niet gespecificeerd (::).
   Aanvrager beschikt nog niet over IPv6 adres. Dus het *Neighbor Advertisement* antwoord moet gemulticast worden naar linklokale *all-nodes* adres (FF02::1). De *Solicited flag* wordt niet ingesteld.
   Als de aanvrager geen *Neighbor Advertisement* ontvangt, wil dit zeggen dat het IPv6 adres nog niet gebruikt wordt.

3. **Reachability detection:**
   - Achterhaalt of een eindbestemming bereikbaar is
   - of de eerste hop naar een willekeurige eindbestemming bereikbaar is.

   Bereikbaarheid staat vast als recente bevestigd is dat IPv6 datagrammen verwerkt worden.
   Bevestiging kan op 2 manieren:
   - Impliciet: door ontvangt van TCP acknowledgement berichten.
   - Expliciet: met behulp van *Neighbor Solicitation* en *Neighbor Advertisement* berichten. Als *Solicited flag* niet is ingesteld dan is het geen bewijs van bereikbaarheid (= bewijst alleen verbinding in 1 richting).
     Wordt geparametriseerd door 2 timers: *Retrans Timer* & *Reachable Timer*.

4. **Redirect:**
   Wordt verstuurd naar afzender als router merkt dat de afzender niet over de efficiëntste routetabel beschikt.
   Na het doorsturen van het bericht stuurt de router een *Redirect* bericht naar de oorspronkelijke afzender, enkel als die geen router is.
   Als *Redirect* bericht ook de optie *Target Link-Layer Address* bevat kan de ontvanger ervan ook zijn *neighbor cache* updaten.
   Als de afzender niet over de juiste prefix van het eigen subnetwerk beschikt wordt dit ook gecorrigeerd via het *Redirect* bericht.