# Computernetwerken: Opgeloste vragen

Kenneth Hoste

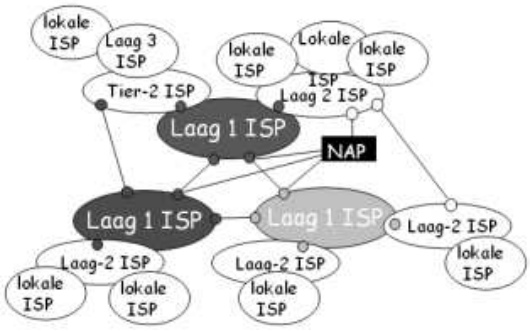
academiejaar 2003-2004

# Hoofdstuk 1

# Inleiding

1.1 ***Bespreek de structuur van het Internet als "network of networks".***

Het internet bestaat uit een groot aantal netwerken die onderling verbonden zijn. De toegangsnetwerken aan de rand van het internet zijn verbonden met de rest van het internet via een gelaagde hiërarchie van **ISP's (internet service providers)**. Bovenaan bevinden zich een klein aantal **eerstelaags ISP's**, die vergelijkbaar zijn met een gewoon netwerk: ze bestaan uit links (die meestal sneller zijn dan 622 Mb/s, bij grote eerstelaags ISP's is dat zelfs tussen de 2,5 en 10 Gb/s) en routers, en zijn verbonden met andere netwerken. Ze hebben enkele kenmerken gemeen: ze zijn rechtstreeks verbonden met *alle* andere eerstelaags ISP's, zijn verbonden met een groot aantal tweedelaags ISP's en andere klantennetwerken en ze omspannen verschillende landen. Eerstelaags ISP's worden ook **internetbackbones** genoemd. Voorbeelden: UUNet, Sprint, AT&T, ... Een **tweedelaags ISP** bestrijkt gewoonlijk een regio of een land, en heeft




verbindingen met slechts enkele eerstelaags ISP's. Een tweedelaags ISP is dus een *klant* van de eerstelaags ISP waarmee die verbonden is, en een eerstelaags ISP is een *provider* voor die klant. Veel grote bedrijven en instituten zijn rechtstreeksverbonden met een dergelijk tweedelaags ISP, en zijn dus klant van die ISP. Om het onderling uitwisselen van data tussen 2 tweedelaags ISP te realiseren zonder tussenkomst van een eerstelaags ISP, kiezen die ISP's er soms voor om een rechtstreekse verbinding te hebben

met elkaar.
Onder de tweedelaags ISP's bevinden zich ook nog **derdelaags ISP's**, en ook nog **toegangs-ISP's**. Bovendien zijn sommige eerstelaags ISP's ook nog tweedelaags ISP, omdat ze rechtstreeks internettoegang aanbieden aan eindgebruikers, op dezelfde manier als lagergelegen ISP's dat doen. Als twee ISP's met elkaar rechtstreeks verbonden zijn, worden ze een **peer** van elkaar genoemd.
Het punt binnen het netwerk van een ISP waar die ISP verbonden is met een andere ISP, wordt een **Point of Presence (POP)** genoemd. Een dergelijke POP is een groep routers waarmee de routers van een ander ISP kunnen communiceren. Als een klant-ISP dan een verbinding wil maken met een provider, verbindt die een van zijn eigen routers rechtstreeks met een router van het POP van de provider. ISP kunnen peer zijn van elkaar wanneer ze twee POP's van elkaar met elkaar verbinden.
Naast niet-openbare verbindingen van ISP's kunnen ISP's ook verbonden zijn via **Network Access Points (NAP's)**, die eigendom kunnen zijn (of beheerd worden door) een telecombedrijf of een internetbackboneprovider. Dergelijke NAP's zijn complexe snelle schakelnetwerken en verwerken enorme hoeveelheden dataverkeer tussen een groot aantal ISP's. Momenteel worden NAP's meestal gebruikt voor de rechtstreekse verbindingen tussen tweedelaags ISP's onderling en verbindingen tussen eerstelaags en tweedelaags ISP's, terwijl POP's gebruikt worden voor de onderlinge rechtstreekse verbindingen tussen eerstelaags ISP's.

1.2 ***Leg uit wat een protocol is.***

In een **protocol** worden de indelingen en de volgorde van de berichten vastgelegd die tussen twee of meer communicerende entiteiten worden uitgewisseld en de reacties bij het verzenden en/of ontvangen van een bericht of een andere gebeurtenis.
Alle activiteiten op het internet, tussen twee of meer communicerende entiteiten op afstand, worden uitgevoerd volgens een protocol. Er bestaan verschillende protocollen, die elk gebruikt wordt voor verschillende applicaties, bvb. voor bestandsoverdracht (FTP), email (SMTP), webbrowsing (HTTP), ...

1.3 ***Geef de verschillende lagen van het TCP/IP referentiemodel. Geef bij elke laag een voorbeeld (leg kort uit).***

**applicatielaag**:
is verantwoordelijk voor de ondersteuning van netwerktoepassingen, en werkt met een groot aantal protocollen, zoals HTTP voor webbrowsing, SMTP voor e-mail en FTP voor bestandsoverdracht

**transportlaag**:
zorgt voor het transport van berichten van de applicatielaag tussen de client en server van een toeassing; op het internet worden er twee gebruikt: TCP (connection-oriented, betrouwbaar, flow control, congestion control) en UDP (connectionless, weing extra functionaliteit)

**netwerklaag**:
is verantwoordelijk voor het bepalen van het pad van datagrammen van de ene host naar de andere; bestaat uit 2 basiscomponenten: IP-protocol

(slects één, waar velden in IP-datagram gedefinieerd worden, en bepaald wordt hoe eindsystemen en routers ermee omgaan) en routingprotocollen (groot aantal, bepalen het pad van datagrammen tussen bron en bestemming); alle internetcomponenten met een netwerklaag moeten het IP-protocol gebruiken, en kunnen eender welk routingprotocol gebruiken

**datalinklaag**:
transporteert complete frames van het ene netwerkelement naar het andere; voorbeelden zijn Ethernet en PPP; omdat datagrammen bijna altijd verschillende links moeten passeren om aan te komen bij de ontvanger, kan een datagram door verschillende datalinklaagprotocollen afgehandeld worden bij verschillende links op het pad

**fysieke laag**:
moet afzonderlijke bits in een frame van een node naar een volgende node verplaatsen; deze protocollen zijn linksafhankelijk en worden bepaald door het fysieke transmissiemedium van de link; Ethernet heeft verschillende protocollen in de fysieke laag: een voor twisted-pair koperdraad, een ander voor coaxiale kabel, nog een ander voor glasvezel, enz. . . ; telkens wordt een bit op een speciale manier getransporteerd

1.4 ***Bespreek de algemene werking van FTP.***

Als een bestand over FTP aangevraagd wordt, wordt er eerst een TCP controle connectie opgezet. Via die controle connectie, wordt de gebruiker gevraagd om in te loggen met een login en een paswoord. Eens de login geverifieerd werd door de FTP server, wordt (nog steeds over de controle connectie) de directory aangevraagd, m.a.w. een lijst met de beschikbare files. Daarop wordt een TCP transfer (data) connectie opgezet, waarlangs de gevraagde directory wordt verstuurd. Eens die directory binnen is, zal de client, opnieuw langs de controle connectie, een bestandsaanvraag doen, waarop opnieuw een transfer connectie wordt opgezet, waarlangs het bestand dan verstuurd zal worden. Voor elk bestand dat opgevraagd (of aangeboden) wordt, wordt een aparte data connectie opgezet.
Het FTP-protocol wordt *out-of-band* genoemd, omdat het gebruik maakt van een aparte verbinding om controle-informatie te versturen. HTTP is daarentegen *in-band*.
Een FTP-server dient, in tegenstelling tot een HTTP-server, per client de status van die client bij te houden, onder andere om de huidige directory van die client bij te houden. Daardoor kan de FTP-server slechts een beperkt aantal clients bijhouden.

1.5 ***Bespreek algemeen de eigenschappen van TCP en UDP. Leg uit en vergelijk.***

**eigenschappen van TCP**:

- connectie georienteerd (virtuele verbinding in software)
- 3-way handshake protocol
- point to point, full duplex (dataoverdracht kan in 2 richtingen tegelijk)
- uitwisseling van segmenten (eenheid van data): tijdens een bestandsoverdracht zal het bestand in stukken gedeeld worden: segmenten

- betrouwbare overdracht: gebruik van acknowledgement, retransmissie, klokken, enz...
- flow controle (zender zal de ontvanger niet overladen met segmenten)
- congestion controle (zender zal transmissiesnelheid verlagen als het netwerk overbelast is)
- segmenten worden verstuurd over een netwerk van routers in de IP-laag
- wordt typisch gebruikt voor http, ftp, smtp, pop, ...

**eigenschappen van UDP**:

- zeer eenvoudig protocol
- alternatief voor complexe TCP protocol
- onbetrouwbaar en connectionless
- geen flow controle of congestion controle
- een-richtings-verkeer
- geen extra vertragingen door acknowledgements
- typisch gebruikt voor real-time and controle applicaties

1.6 ***Bespreek de algemene werking van het internetprotocol (IP) en de typische eigenschappen.***

Het IP-protocol zal aan ieder segment dat dient verstuurd te worden, een IP header gaan toevoegen, waarin vermeld is welk protocol dient gebruikt te worden om het pakket, eens aangekomen bij de ontvanger, te verwerken, alsook de bron (source) en de bestemming (destination) van het pakket. Dat IP datagram wordt dan verstuurd over de links en routers naar de bestemming.

*eigenschappen*

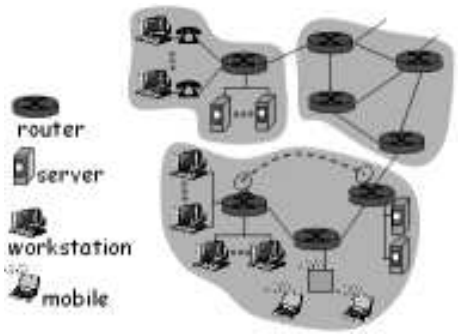
- een-richtingsverkeer
- datagram gebaseerd opslaan en doorsturen
- connectionless
- flexibiliteit wordt geleverd door netwerkelementen (routers)
- best effort: geen garantie over vertraging, aflevering, ...
  (geen Quality of Service (QoS))
- de IP-laag is de serverlaag van de TCP-laag, en de clientlaag van de datalinklaag

1.7 ***Bespreek het principe van encapsulatie.***

Encapsulatie betekent dat een pakket van een laag (bvb. de transportlaag) zal dienen als data voor een onderliggende laag (bvb. de netwerklaag). Een TCP-pakket zal door de netwerklaag beschouwd worden als data, en het IP-protocol zal een IP-header toevoegen, met extra informatie, zodat de ontvanger aan de hand van de verschillende headers in het pakket weet welk protocol het moet gebruiken, waar het vandaan komt, enz... Telkens een header ingelezen is, zal het resterende pakket doorgegeven worden aan een hogere laag, waar opnieuw de bijhorende header ingelezen wordt, en dit gaat verder totdat men de eigenlijke data bekomt.

1.8 ***Leg het verschil uit tussen een host en een router.***

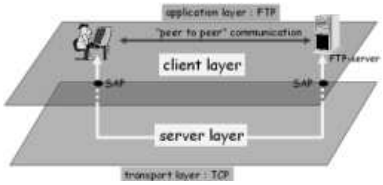
- **hosts** (workstations, servers) bevinden zich aan de rand van het internet, terwijl **routers** zich in de kern van het internet bevinden.




- een **host** zal applicatie programma's hebben, zoals een webbrowser, e-mail programma, enz... Een **router** draait geen applicatie programma's, en heeft enkel een routeringstabel.

1.9 ***Leg uit: client en server laag. Geef een voorbeeld. Vergelijk met client/server applicaties.***

Bij ieder communicatie is er een client laag, en een server laag. De server laag doet op zijn beurt dienst als client laag voor een ander, lager protocol, enz... Als voorbeeld bekijken we onderstaande figuur.




In dit voorbeeld doet de applicatielaag dienst als clientlaag voor FTP, terwijl de transportlaag dienst doet als serverlaag. De applicatielaag zal de diensten van de transportlaag gebruiken, die garandeerd dat de FTP pakketten foutloos, en in de juiste volgorde zullen aankomen. De transportlaag zal dan op zijn beurt clientlaag zijn voor de netwerklaag, die zal dienen als serverlaag, en dus zal de transportlaag gebruik maken van de diensten die de netwerklaag levert.

1.10 ***Bespreek identificatie in de applicatie-, transport- en netwerklaag.***

**applicatielaag**:

poortnummer, bvb. FTP: 20 (data), 21 (controle)

**transportlaag**:

protocol nummer, bvb. TCP: 6, UPD: 17

**netwerklaag**:

ip adres, bvb. 157.193.122.1

1.11 ***Hoe noemt men de informatieblokken in de applicatie-, transport-, netwerk- en datalinklaag?***

**applicatielaag**: message (bvb. een e-mailboodschap met header)

**transportlaag**: segment (bvb. een TCP-segment)

**netwerklaag**: datagram (bvb. een IP-datagram)

**datalinklaag**: frame (een reeks bits)

1.12 ***Wat is: IETF, RFC, ISP? Geef kort uitleg.***

**IETF**: *Internet Engineering Task Force*: de organisatie die de internetstandaarden ontwikkelt, die gepubliceerd worden in documenten die RFC genoemd worden.

**RFC**: *Request For Comment*: de documenten die de IETF publiceerd, waarin de internetstandaarden staan uitgelegd. Ze waren aanvakelijk bedoeld als algemeen verzoek om commentaren, en zien formeel gezien geen standaarden, maar worden wel zo bekeken. De documenten zijn over het het algemeen erg technisch en zeer gedetailleerd, en beschrijven protocollen zoals TCP, IP, HTTP en SMTP.

**ISP**: *Internet Service Provider*: een bedrijf dat internettoegang en diensten verleent aan zijn klanten. Er zijn verschillende klassen ISP's (eerstelaags-, tweedelaags-, derdelaags- en toegangs-ISP's).

# Hoofdstuk 2

# Applicatielaag

2.1 ***Bespreek het client-server principe op applicatieniveau.***

Bij iedere netwerktoepassing zijn er steeds 2 componenten: een **client-component** en een **servercomponent**. De clientcomponent op een eindsysteem communiceert met de servercomponent op een ander eindsysteem. Bij veel toepassing implementeert een host zowel de client- als de servercomponent van een toepassing, bvb. bij FTP: beide hosts die betrokken zijn in een FTP-sessie kunnen een bestand kopiëren naar de andere host. Er is dus ook in beide richtingen communicatie mogelijk.
Men noemt een client "active open", terwijl een server "passive open" is. De eigenschappen van beide zijn hieronder weergegeven.

**client (*active open*)**

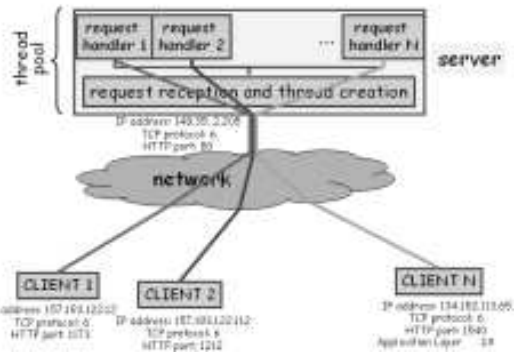
- wordt direct door de gebruiker geinitieerd
- lokaal op de computer van de gebruiker
- initieert *actief* zelf het contact met de server
- één sessie per keer
- kan toegang hebben tot meerdere services indien nodig
- eenvoudige hard- en software
- bvb.: Eudora, Internet Explorer

**server (*passive open*)**

- speciaal programma voor een bepaalde service
- kan meerdere clients op afstand tegelijk aan
- draait op een gedeelde computer
- wacht *passief* op contact van een eender welke client op afstand
- krachtige hardware en een gesofisticeerd besturingssysteem
- het serverprogramma wordt ook *daemon* genoemd
- bvb. Apache, sendmail

2.2 ***Bespreek het concept van "threads". Geef een voorbeeld.***

Een server beschikt over een *thread pool*, en kan op die manier, via verschillende *request handlers* meerdere aanvragen van verschillende clients tegelijk afhandelen. De clients maken telkens verbinding op dezelfde poort van de server, die ervoor zorgt dat die een bepaald aantal requests (evenveel als er request handlers zijn in de thread pool) tegelijk kan afhandelen, door voor elke request die binnenkomt, een thread op te starten.




In bovenstaand voorbeeld zien we hoe verschillende client met een HTTP server verbinding maken. Alle requests komen binnen op poort 80 van de server, die ervoor zorgt dat er N requests tegelijk kunnen afgehandeld worden.

2.3 ***Welke transportdiensten kan een applicatie vereisen? Geen enkele voorbeelden.***

**Betrouwbare gegevensoverdracht**

Sommige toepassingen hebben een absolute betrouwbare gegevensoverdracht nodig, m.a.w. er mag géén gegevensverlies optreden. Dergelijke toepassingen zijn bvb. e-mail, bestandsoverdracht en financiële toepassingen. Andere toepassingen zijn dan weer verlies-tolerant, vooral multimediatoepassingen zoals realtime audio/video. Als er gegevensverlies optreedt bij dergelijke toepassingen, resulteert dit enkel in een mindere weergavekwaliteit.

**Bandbreedte**

Sommige toepassingen moet gegevens met een bepaalde snelheid kunnen versturen om te kunnen werken. Men spreekt dan over een *bandbreedtegevoelige toepassingen*. Vooral multimediatoepassingen zijn bandbreedtegevoelig, maar toekomstige multimediatoepassingen zullen wellicht coderingstechnieken gebruiken die reageren op de beschikbare bandbreedte. Toepassingen die kunnen werken met de bandbreedte die beschikbaar is, of dat nu veel of weinig is, worden *elastische toepassing* genoemd. Voorbeelden zijn e-mail en bestandsoverdracht.

**Timing**

Vooral interactieve realtime-toepassingen, zoals internettelefoon, virtuele omgevingen en multiplayergames stellen strenge eisen aan timing voor het bezorgen van gegevens, omdat ze anders niet kunnen functioneren. Veel

van deze toepassingen eisen bvb. een end-to-end vertraging in de orde van grootte van enkele honderen ms of zelfs nog kleiner. Lange vertragingen bij bvb. multiplayergames resulteren in een lange vertraging tussen het uitvoeren van een handeling en de respons van de omgeving, wat resulteert in een veel minder realistische ervaring.

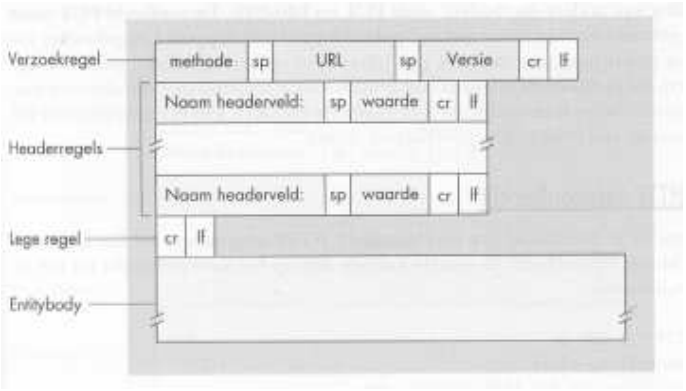
In onderstaande tabel zijn de eigenschappen van enkele internettoepassingen weergegeven.

| Toepassing | Gegevensverlies | Bandbreedte | Timinggevoeligheid |
|---|---|---|---|
| Bestandsoverdracht | Geen verlies | Elastisch | Nee |
| E-mail | Geen verlies | Elastisch | Nee |
| Webdocumenten | Geen verlies | Elastisch (enkele Kbps) | Nee |
| Realtime audio/video | Verliestolerant | Audio: Enkele Kbps-1 Mb<br>Video: 10 Kb - 5 Mb | Ja, honderden ms |
| Opgeslagen audio/video | Verliestolerant | Als hierboven | Ja, enkele seconden |
| Interactieve spellen | Verliestolerant | Enkele Kbps-10 Kbps | Ja, honderden ms |
| Direct messaging | Geen verlies | Elastisch | Ja en Nee |

2.4 ***Bespreek het HTTP protocol en de belangrijkste protocolboodschappen.***

In het HTTP protocol zijn er 2 soorten berichten: *verzoekberichten* en *antwoordberichten*.

**HTTP-verzoekberichten**

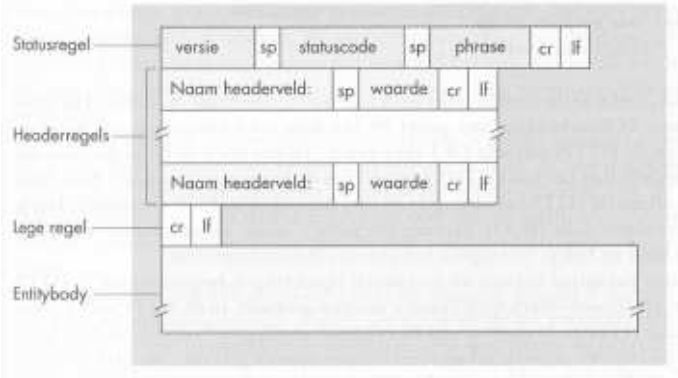



De eerste regel van een HTTP-verzoekbericht wordt de *requestregel* genoemd, de volgende regels noemt men *headerregels*. Daarna volgt ook nog een *entity body*, die leeg kan zijn. De requestregel bevat drie velden: het methodeveld, het URL-veld en het veld met het versienummer van HTTP. Het methodeveld kan verschillende waarden hebben, zoals *GET* (browser vraagt object op bij server), *POST* (entity body bevat ingevulde gegevens van formlier) en *HEAD* (wordt vooral gebruikt bij opsporen van fouten, toont enkel eerste lijnen van het bericht, zonder het entitybody)

bij HTTP/1.0 en ook nog *PUT* (uploaden van een object naar specifiek pad op server) en *DELETE* (object verwijderen op de webserver) bij HTTP/1.1. Het URL-veld bevat steeds het pad van het object dat bvb. opgevraagd wordt. In het veld met versienummer staat weergegeven welk HTTP-versie de browser implementeert (HTTP/1.0 of HTTP/1.1). De headerregels bevatten bijkomende informatie. We geven enkele voorbeelden:

- *Host*: geeft aan op welke host het gevraagde object staat
- *Connection*: als dit attribuut de waarde 'close' heeft, wordt aangegeven dat er geen persistente verbinding gebruikt wordt
- *User-agent*: geeft aan welke user-agent (browser) gebruikt wordt; vb'n: Internet Explorer, Mozilla, Netscape
- *Accept-language*: geeft aan wat de voorkeurstaal is van de gebruiker (mogelijk bestaan er verschillende versies van dezelfe pagina op de server)
- *If-modified-since*: wordt gebruikt bij conditional GET, server zal enkel object opnieuw sturen als die gewijzigd werd sinds de gegeven datum, anders wordt het object uit de cache gehaald van de client

**HTTP-antwoordberichten**




Een HTTP-antwoordbericht bestaat ui drie componenten: de *statusregel*, enkele *headerregels* en een *entity body*. Het entity body bevat het opgevraagde object. De statusregel bevat drie velden: een veld met de protocolversie, een met de statuscode en een met het bijhorende statusbericht. We bekijken enkele veelvoorkomende statuscodes en de bijhorende berichten:

- *200 OK*: verzoek is uitgevoerd en informatie is verzonden in een antwoordbericht
- *302 Moved Permanently*: opgevraagd object is permanent verplaatst; de nieuwe URL wordt gegeven in het attribuut Location (clientsoftware zal de nieuwe URL automatisch opvragen)
- *304 Not Modified*: het object werd niet aangepast sinds de meegegeven datum, dit wordt gebruikt bij conditional GET

- *400 Bad Request*: algemene foutcode, server kan verzoek niet interpreteren
- *404 Not Found*: opgevraagde object is niet aanwezig op deze server
- *505 HTTP Version Not Supported*: versie van het opgevraade HTTP-protocol is wordt niet ondersteund door de server

We bekijken nu ook enkele headerregels:

- *Connection*: als dit attribuut de waarde 'close' heeft, wordt aangegeven dat de verbinding zal gesloten worden nadat het bericht is verzonden
- *Date*: bevat de tijd en de datum waarop het HTTP-antwoordbericht werd gemaakt en verzonder door de server
- *Server*: geeft aan dmv welke server het bericht gegenereerd werd; vb'n: Apache, IIS
- *Last-Modified*: geeft aan wanneer het object laatst aangepast werd, dit is belangrijk voor cache-doeleinden
- *Content-Length*: bevat het aantal bytes dat het opgevraagd object groot is
- *Content-Type*: bepaalt wat er in de entity-body zit (bvb. HTML-tekst), het object-type wordt officieel niet bepaald door de extentie, maar door deze headerregel

Er bestaan nog verschillende headerregels, onder andere voor cookies.

### 2.5 *Waarvoor staat: HTTP, URL, HTML? Geef kort uitleg.*

**HTTP** *Hypertext Transfer Protocol*: het applicatielaagprotocol voor het web, berichten die verzonden worden tussen een browser en een server maken gebruik van dit protocol

**URL** *Uniform Resource Locator*: adres van een bepaald object; een URL bestaat uit twee componenten: de *hostnaam* van de server waarop het object is opgeslagen (bvb. www.someSchool.edu) en het *pad* van het object op die server (bvb. /someDepartment/picture.gif)

**HTML** *Hypertext Markup Language*: opmaaktaal die gebruikt wordt voor webpagina's, browser interpreteert de HTML om de pagina weer te geven

### 2.6 *Bespreek de verschillende HTTP connectiemogelijkheden.*

HTTP kan zowel met non-persistenten als persistente verbindingen werken. HTTP/1.0 maakt standaard gebruik van non-persistente verbindingen, terwijl HTTP/1.1 standaard persistente verbindingen gebruikt, maar de clients en servers die HTTP/1.1 ondersteunen kunnen ook geconfigureerd worden om gebruik te maken van non-persistente verbindingen.

**Non-persistente verbindingen**

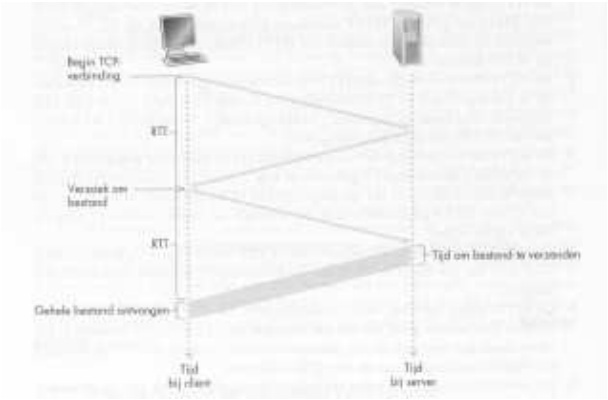
Bij elk HTTP-verzoekbericht wordt een TCP-verbinding geinitieerd. Via de socket die gekoppeld is aan die TCP-verbinding verstuurt de client het verzoekbericht naar de server. De server ontvangt het verzoekbericht, verpakt het object in een HTTP-antwoordbericht, verzendt het antwoordbericht via de socket en geeft daarna opdracht om de TCP-verbinding te

verbreken. De client zal het antwoordbericht ontvangen, en pas daarna de TCP-verbinding verbreken. De client zal dan het verpakte HTML-bericht gaan lezen, de verwijzingen erin gaan bekijken (bvb. naar afbeeldingen), en voor ieder object waar na verwezen wordt opnieuw een aparte TCP-verbinding gaan opstarten en afhandelen. Het is mogelijk om de TCP-verbindingen om die objecten op te halen parallel te laten verlopen, standaard gebruiken de meeste browsers vijf tot tien parallelle verbindingen, die elk een verzoek-antwoordtransactie verwerken.

**Persistente verbindingen**

Bij persistente verbindingen zal de server de verbinding open laten nadat die aan antwoordbericht verstuurd heeft. Op die manier kunnen opeenvolgende HTTP berichten tussen dezelfde client en server verstuurd worden over dezelfde verbinding. Er zijn 2 soorten persistente verbindingen: met *pipelining* en *zonder pipelining*. Bij een persistente verbinding zonder pipelining zal de client wachten met het versturen van een nieuw verzoek tot een antwoord ontvangen werd. Als wel pipelining gebruikt wordt, dit is de standaard instelling bij HTTP/1.1, zal de client verzoekberichten versturen op het moment dat het een verwijzing tegenkomt, en zal dus niet eerst wachten tot het vorige opgevraagde object eerst ontvangen werd.

2.7 ***Bespreek een eenvoudig model voor responstijd bij HTTP.***




We willen een berekening maken voor de tijd tussen het moment dat een client een verzoek doet en het moment dat het gevraagde object wordt ontvangen door de client. We noemen deze tijd de **round-triptijd (RTT)**. De RTT bestaat onder andere uit propagation delays en queuing delays van het pakket over de tussenliggende routers en switches en de processing delay.
We bepalen eerst de totale responstijd bij een non-persistente verbinding: er wordt eerst een TCP-verbinding tot stand gebracht tussen de browser en de webserver. Daarvoor is een drievoudig handschaking-protocol nodig: de client verzendt een klein TCP-bericht naar de server, de server antwoordt met een klein TCP-bericht als bevestiging, en daarop stuurt de client eveneens een klein TCP-bericht met een bevestiging. De tijd tussen de eerste twee handelingen kost al één RTT. Daarna zal de client reeds het HTTP-verzoekbericht versturen, samen met de bevesting die de 3-

way-handschaking afrondt. De server zal antwoorden met een HTTP-antwoordbericht dat het gevraagde object bevat. Dit kost dus ook nog eens een RTT. De totale responstijd is dus twee RTT's, samen met de transmissietijd van het bestand op de server.
Bij persistente verbindingen zonder pipelining zal er een RTT gebruikt worden om de verbinding op te zetten, en dan nog een RTT per object dat gevraagd wordt. Dit is dus beter dan bij non-persistente verbindingen, waar er per object een aparte verbinding nodig is, die telkens 2 RTT's kost (samen met de transmissietijd van het object zelf).
Indien er wel pipeling gebruikt wordt, zal er een RTT zijn om de verbinding op te zetten, en dan één RTT voor alle gekoppelde objecten, omdat de verzoeken direct na elkaar gebeuren en de objecten ook direct na elkaar verstuurd worden.

2.8 ***Bespreek het principe van cookies.***

Cookies zijn een manier om samen met het toestandsloos protocol HTTP toch bepaalde dingen bij te houden over de gebruikers. De technologie bestaat uit 4 componenten: een cookie headerregel in het HTTP-antwoordbericht, een cookie headerregel in het HTTP-verzoekbericht, een cookiebestand op het eindsysteem van de gebruiker en een back-enddatabase op de server.
Eerst zal een HTTP-antwoordbericht een headerregel bevatten met 'Set-cookie' en een waarde. Daarop zal de client een cookiebestand bijhouden, dat door de browser beheerd wordt. Elke keer als de gebruiker dan een bestand zal opvragen van de server, zal de browser ervoor zorgen dat het unieke nummer dat ontvangen werd van de server, in het HTTP-verzoekbericht staat dat verstuurd wordt, d.m.v. een headerregel met 'cookie' en een waarde. Op die manier kan de server bepaalde 'beslissingen' nemen aan de hand van dat nummer en zijn eigen back-enddatabase.
Cookies kunnen voor verschillende dingen gebruikt worden, waaronder authorisatie, winkelwagentjes, status bijhouden (bvb. bij webmail), enz...

2.9 ***Bespreek conditional GET en waarvoor is dat nuttig?***

Bij *conditional GET* wordt een headerregel met 'If-modified-since' en een waarde toegevoegd aan het HTTP-verzoekbericht. De server zal dan kijken of het gevraagde bestand gewijzigd werd sinds de datum die in dat attribuut meegegeven werd. Indien die gewijzigd werd, zal de server antwoorden met een bericht dat identiek is als wanneer een gewoon verzoek gedaan zou zijn. Indien er echter niets gewijzigd werd aan het gevraagde bestand, zal de server antwoorden met een HTTP-antwoordbericht dat als statuscode 304 heeft not modified). Op die manier weet de client dat die het bestand uit zijn cache mag halen.
Conditional GET (of voorwaardelijke GET) is een methode die toelaat om de gebruikte bandbreedte te verlagen, doordat het bestand niet telkens opnieuw dient doorgestuurd te worden, maar rechtstreeks uit de cache gehaald kan worden.

2.10 ***Geef een overzicht van de verschillende e-mail protocols (afkorting + korte uitleg wat het doet).***

**SMTP**

*Simple Mail Transfer Protocol*: wordt gebruikt om e-mail berichten te 'pushen' naar de eigen mailserver, en om e-mail berichten te versturen van de ene mailserver naar de andere

**POP3**

*Post Office Protocol 3*: wordt gebruikt om e-mail berichten op te halen van de eigen mailserver, en om bijvoorbeeld berichten te verwijderen op de mailserver (gebeurt in 3 fases: autoriasatiefase, transactiefase en updatefase)

**IMAP**

*Internet Message Access Protocol*: wordt voor hetzelfde gebruikt als POP3, maar is meer geavanceerd; laat toe om de berichten op te slaan in mappen op de mailserver, en is daardoor ook veel complexer dan POP3; IMAP maakt het ook mogelijk om mappen te doorzoeken op berichten doe voldoen aan bepaalde criteria, en om afzonderlijke delen van berichten (header, enkel figuur uit multi-part bericht) op te halen

2.11 ***Bespreek het gebruik van naam en adres. Geen een voorbeeld.***

Hosts op het internet kunnen aangeduid worden m.b.v. een hostnaam of een IP-adres. Een hostnaam is makkelijker voor gebruikers, maar routers kunnen er niet mee om. Daarom moeten hostnamen via DNS omgezet worden in IP-adressen.
Een IP-adres bestaat uit 4 bytes, een heeft een hiërarchische structuur. Het wordt genoteerd door 4 getallen tussen 0 en 255, telkens gescheiden door een punt (.). Een IP-adres is hiïarchisch om we als we het van links naar rechts lezen, steeds meet informatie over de plaats van de host binnen het netwerk van netwerken (het internet).
Een voorbeeld: *hostnaam*: 'relay1.bar.foo.com' ; *IP-adres*: 145.37.93.126

2.12 ***Bespreek de DNS hierarchie. Geef een voorbeeld.***

Alle informatie die een hostnaam koppelt aan een IP-adres, is verdeeld over verschillende nameservers die verspreid zijn over de hele wereld. Er zijn 3 soort nameservers, die zowel onderling als met verzoekende hosts informatie uitwisselen:

**Lokale nameservers**

Elk ISP heeft een lokale nameserver. Die bevat DNS-records voor alle hosts die verbonden zijn met dat bepaalde ISP. Als een host een IP-adres opvraagt die verbonden is met dezelfde ISP, zal de lokale nameserver dus het gevraagde IP-adres kunnen leveren. (vb. host 'surf.eurecom.fr' vraagt IP-adres van 'baie.eurecom.fr', lokale nameserver 'dns.eurecom.fr' van Eurecom zal antwoorden)

**Root nameservers**

Als een lokale nameserver geen antwoord kan geven op het verzoek van een host, zal die zelf DNS-cliënt worden, en een verzoek versturen naar een van de (tientallen) root nameservers (als recursieve verzoeken gebruikt

worden). Als de root nameserver de gevraagde hostnaam in zijn tabel terugvindt, zal die een DNS-antwoordbericht terugsturen naar de lokale nameserver, die het zal doorsturen naar de verzoekende host. Als de root nameserver geen DNS-record bevat voor de gevraagde hostnaam, zal die wel het IP-adres kennen van de verifiërende namesever (of een tussenliggende namserver, die de weg weet naar een verifiërende namesever), die zeker een verwijzing naar het IP-adres van de gevraagde host moet bevatten. (vb. host 'surf.eurecom.fr' vraagt IP-adres van 'gaia.cs.umass.edu', en root-nameserver X bevat DNS-record met die hostnaam, dan zal X antwoorden via de lokale nameserver van Eurecom)

**Verifiërende nameservers**

Elke host is aangemeld bij een verifiërende nameserver (standaard een nameserver van de lokale ISP van de host). Op die manier zal een verifiërende nameserver die een DNS-verzoekbericht ontvangt van een root nameserver antwoord geven aan die root nameserver, die zal antwoorden via de lokale nameserver. Er kunnen nog verschillende nameservers tussen een root nameserver en een verifiërende nameserver liggen (vb. tussenliggende nameserver: 'dns.umass.edu', verifiërende nameserver: 'dns.cs.umass.edu'). (vb. host 'surf.eurecom.fr' vraagt IP-adres van 'gaia.cs.umass.edu', en root nameserver X bevat geen DNS-record met die hostnaam; de root nameserver zal het verzoek doorsturen naar de gekende verifiërende nameserver 'dns.umass.edu' die zal antwoorden via de root nameserver en de lokale nameserver)

2.13 ***Bespreek het iteratief en recursief mappen in DNS.***

Er zijn 2 mogelijkheden om een host van een DNS-antwoordbericht te voorzien. Ofwel kan de nameserver die het verzoek binnenkrijg, het verzoek doorsturen naar een andere nameserver, die dan uiteindelijk een DNS-antwoordbericht zal terugsturen, en waardoor de nameserver die het verzoek binnenkreeg, de host kan voorzien van antwoord (recursief), ofwel kan de nameserver een DNS-antwoordbericht versturen met de volgende nameserver in de ketting, die mogelijk een antwoord weet.
Meestal zullen alle verzoeken recursief gebeuren, behalve die naar root nameservers, die zullen iteratief gebeuren (omdat root nameservers veel verzoeken moeten verwerken, en iteratieve verzoeken de root nameservers minder belasten (er hoeft niets bijgehouden te worden i.v.m. de verzoeken van hosts of andere nameservers).

2.14 ***Wat is (in de context van DNS): RR, A, NS, CNAME, MX (geef ook een voorbeeld).***

**RR**

*Resource Record*: de verschillende nameservers die samen de databank van DNS vormen, bevatten bronrecords (resource records) voor de hostnaam/IP-adres combinatie. Elk DNS-antwoord zal verschillende dergelijke DNS-records bevatten. Een bronrecord is een 4-tupel:

```
(Naam, Waarde, Type, TLL)
```

TTL is de levensduur (Time To Live) van de bronrecord; de betekenis van de velden Naam en Waarde hangt af van de waarde van het veld

Type. Hieronder zijn de verschillende mogelijke waarden van het veld Type vermeld, telkens met uitleg over de velden Naam en Waarde.

**A**

Het veld Naam bevat een hostnaam, het veld Waarde bevat een IP-adres. Een bronrecord van Type A levert dus de standaard hostnaam/IP-adres combinatie.
vb.: `(plinius.intec2.ugent.be,157.193.122.4,A,100)`

**NS**

Het veld Naam bevat een domeinnaam, het veld Waarde bevat de hostnaam v/e verifiërende nameserver die weet hoe de hostnaam/IP-adres combinaties voor het domein kunnen gevonden worden. Dit type DNS-record wordt gebruikt om het pad van DNS-verzoeken in de ketting te bepalen.
vb.: `(ugent.be,ugdns1.ugent.be,NS,100)`

**CNAME**

Het veld Naam bevat een alias voor een hostnaam, het veld Waarde bevat een canonieke hostnaam voor de alias. Uit dit type DNS-records kunnen verzoekende hosts de canonieke naam voor een hostnaam halen.
vb.: `(mail.intec2.ugent.be,plinius.intect2.ugent.be,CNAME,100)`

**MX**

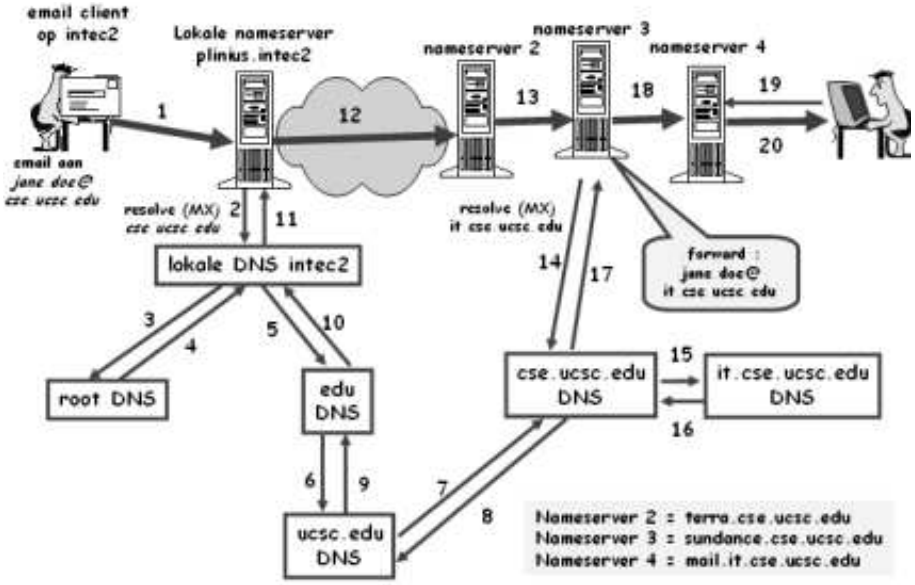
Het veld Naam bevat een domeinnaam, het veld Waard bevat een hostnaam van een mailserver met als alias de gegeven domeinnaam. Dit type DNS-record kan gebruikt worden om de canonieke naam van een mailserver te achterhalen. Dit type record kan ook 'preference' parameters in het Waarde veld bevatten, die de primaire, secundaire, ... mailservers aangeven voor het gegeven domein.
vb.: `(intec2.ugent.be,preference=40 plinius.intec2.ugent.be,MX,100)`
`(intec2.ugent.be,preference=20 mserv.ugent.be,MX,100)`

2.15 ***Op het examen wordt een voorbeeld gegeven (bvb. MIME header, HTML file, DNS request) en er wordt gevraagd dat te bespreken.***

verschillende mogelijkheden, geen oplossing (zie verschillende voorbeelden)

2.16 ***Bespreek het voorbeeld DNS + e-mail (op het einde van het hoofdstuk). De figuur wordt opgegegeven.***

Een gebruiker op het intec2 domein wil een mail sturen naar jane.doe@ csc.ucsc.edu. Daarvoor zal de user agent die door de gebruiker gebruikt wordt, het e-mail bericht in de wachtrij plaatsen van de e-mail server. Daarvoor moet ofwel het IP-adres gekend zijn van de mailserver (zoals hier het geval is), ofwel de hostnaam ervan. In het laatste geval zal eerst het IP-adres van de mailserver moeten bepaald worden via de lokale nameserver.
De lokale mailserver, die dienst doet als client van het SMTP protocol, zal eerst het IP-adres van de mailserver moeten bepalen, die zal zorgen dat het e-mail bericht aankomt bij de ontvanger. Dit gebeurt via DNS: de mailserver zal een DNS-verzoekbericht van het type MX sturen naar de lokale nameserver, waarmee een aanvraag gedaan wordt om voor




csc.ucsc.edu de hostnaam van de mailserver op te zoeken. Die lokale namesever stuurt een iteratief verzoek naar een root nameserver, die als antwoord een IP-adres geeft van een tussenliggende nameserver voor het domein 'edu'. De lokale nameserver stuurt het verzoek dan opnieuw, maar deze keer naar de nameserver van het 'edu' domein. Die zal het verzoek doorsturen naar een andere tussenliggende nameserver van het domein ucsc.edu. Die zal tenslotte het verzoekbericht doorsturen naar de verifiërende nameserver van het domein csc.ucsc.edu. Die zal via de tussenliggende nameservers het antwoordbericht, dat de DNS-records bevat met het IP-adres van de mailserver terra.csc.ucsc.edu, doorsturen naar de lokale nameserver. Op die manier weet de mailserver die client is het IP-adres van de mailserver die server is, en wordt het e-mail bericht via het SMTP-protocol naar de mailserver verstuurd. Die mailserver is zo geconfigureerd dat alle berichten naar een @csc.ucsc.edu geforward worden naar een andere mailserver van. Het IP-adres van die mailserver is gekend, en dus wordt het e-mail bericht via SMTP doorgestuurd naar die mailserver, sundance.csc.ucsc.edu.
Die mailserver weet dat een e-mail bericht naar Jane Doe moet doorgestuurd worden naar mailserver van het domein it.csc.ucsc.edu. Blijkbaar kent die mailserver het IP-adres van die mailserver niet, want die moet opgevraagd worden via DNS. De mailserver zal het DNS-verzoek doorsturen naar de lokale nameserver, die het op zijn beurt zal doorsturen naar de verifiërende nameserver van het domein it.csc.ucsc.edu. Die zal een antwoord terugsturen, dat via de lokale nameserver van csc.uscs.edu de mailserver zal bereiken. Op die manier zal de mailserver het IP-adres van de mailserver mail.it.csc.ucsc.edu weten waarnaar het e-mail bericht verstuurd moet worden.
Als Jane Doe dan zijn mail zal controleren, zal die het e-mailbericht downloaden op zijn computer vanaf de mailserver mail.csc.ucsc.edu via een access-protcol (POP3 of IMAP).

# Hoofdstuk 3

# Transportlaag

3.1 ***Bespreek multiplexering (connection-oriented and connectionless).***

Een *socket* doet dienst al een deur waardoor gegevens van het netwerk naar het proces en omgekeerd worden verplaatst. De transportlaag op de ontvangende host levert de gegevens niet rechtstreeks aan een proces, maar aan een tussenliggende host.
Het verzamelen van gegevens van verschillende toepassingsprocessen op de verzendende host, het verpakken en toevoegen van header-informatie (die zal gebruikt worden bij het demultiplexen) om segmenten te maken en deze segmenten vervolgens door te geven aan de netwerklaag, wordt *multiplexen* genoemd.
Een ontvangende host leidt een inkomend transportlaagsegment naar de juiste socket, en gebruikt daarvoor een aantal velden. Dit wordt *demultiplexen* genoemd.
De speciale velden die daarvoor gebruikt worden, zijn het *bronpoortnummerveld* en het *bestemmingspoortveld*.
Een *poortnummer* is een 16-bit getal dat een waarde tussen 0 en 65535 heeft, waarbij de poortnummers tussen 0 en 1023 gereserveerd zijn voor toepassingsprotocollen, en *well-known poortnummers* genoemd worden.

**Connectionless multiplexen en demultiplexen**

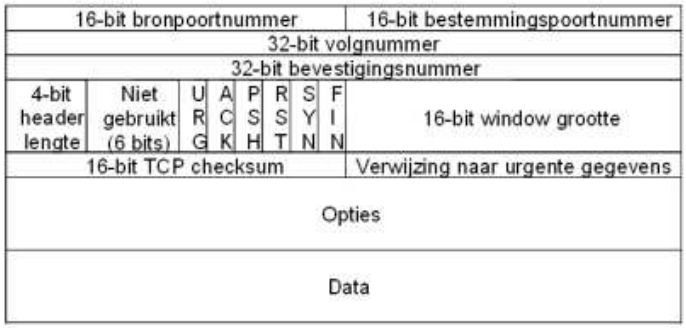
Een voorbeeld van sockets die connectionless multiplexen en demultiplexen gebruiken, zijn UDP sockets. Ze worden aangemaakt met een specifiek poortnummer. Een dergelijke socket wordt geidentificeerd door een twee-tupel dat bestaat uit een IP-bestemmingsadres en een bestemmingspoortnummer. Daardoor kunnen verschillende IP-datagrammen, die verschillende IP-bronadressen en/of bronpoortnummers, maar hetzelfde IP-bestemmingsadres en bestemmingspoortnummer hebben, doorgestuurd worden naar dezelfde socket.

**Connection-oriented multiplexen en demultiplexen**

Een voorbeeld van sockets die connection-oriented multiplexen en demultiplexen, zijn TCP sockets. Ze worden aangemaakt met een IP-adres

en een poortnummer. Een dergelijke socket wordt geidentificeerd door een vier-tupel dat bestaat uit een IP-bronadres, een bronpoort, een IP-bestemmingsadres en een bestemmingspoortnummer. Een server kan verschillende sockets tegelijk beheren, iedere socket is geidentificeerd door zijn eigen vier-tupel. Aks de server een segment van een client met een verzoek tot verbinding binnenkrijgt via de 'welkomstsocket', zal de server een verbindingssocket aanmaken voor die verbinding. Op die manier zal ieder binnenkomend datagram met een vier-tupel dat overeenkomt met het vier-tupel die hoort bij die verbindingssocket, ook doorgestuurd worden naar de verbindingssocket. Bij niet-persistente verbindingen zal ieder voor ieder datagram een nieuwe verbinding opgezet worden, die telkens weer verbroken wordt nadat het datagram doorgegeven werd aan het proces.

3.2 ***Bespreek de verschillende velden van een TCP segment (het segment zelf wordt gegeven op het examen).***




**Bronpoortnummer en bestemmingspoortnummer**

Deze velden (beide 16 bit) zijn nodig voor het multiplexen en demultiplexen van een naar de toepassingen op de bovenste laag.

**Volgnummer**

Dit 32-bits veld duidt het bytestreamnummer aan van de eerste byte in het segment, m.a.w. het geeft aan de hoeveelste byte uit de volledige stroom de eerste byte in het segment is.

**Bevestigingsnummer**

Dit 32-bits veld duidt het volgnummer aan van de volgende byte die de host verwacht te ontvangen. M.a.w. dit nummer duidt aan welke bytes reeds succesvol ontvangen werden. TCP maakt gebruik van cumulatieve bevestigingen, wat betekent dat eens een bevestiging ontvangen is, alle voorgaande bytes zeker in orde zijn.

**Headerlengte**

De headerlengte duidt aan hoe lang de header van dit segment is. Meestal is dat een vast aantal bytes (20), maar indien het optieveld niet leeg is, kan dit meer zijn.

**URG, ACK, PSH, RST, SYN, FIN en verwijzing naar urgente gegevens**

De ACK-bit wordt gebruikt indien het TCP-segment dat verstuurd wordt een bevestiging bezit.
De RST-, SYN- en FIN-bits worden gebruikt om de verbinding tot stand te brengen en te verbreken.
Als de PSH-bit ingesteld is, moet de ontvanger de gegevens direct bezorgen bij de bovenliggende laag.
De URG-bit geeft aan dat dit segment gegevens bevat die door de verzendende entiteit als dringend aangeduid werden. De plaats van de laatste byte van deze gegevens wordt aangegeven met de 16-bit urgente-datapointer.

**Window grootte**

Dit veld wordt gebruikt bij flow controle, en geeft aan hoeveel bytes de ontvanger bereid is om te ontvangen.

**Checksum**

Dit veld wordt, net als bij UDP, gebruikt voor foutcontrole van het ontvangen segment.

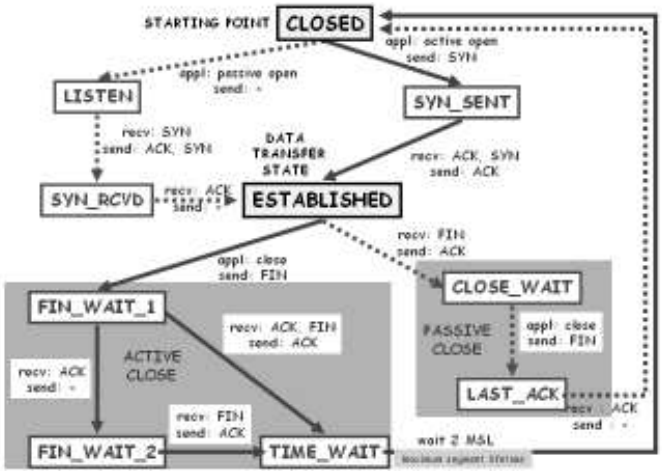
**Optieveld**

Dit veld, dat een variabele lengte heeft, bezit optionele gegevens, bvb. wanneer een verzender en ontvanger onderhandelen over de maximale segmentgrootte (MSG) of de vensteromvang, of om het segment te dateren.

**Gegevensveld**

Dit veld bezit de eigenlijke data die men wenst te versturen. Bij een ACK-bericht is dit veld leeg.

3.3 ***Bespreek het TCP toestandsdiagramma (een skeletfiguur wordt opgegeven, zonder enige tekst). Maak bij de uitleg tevens gebruik van een tijdsverloop (tijdsas client- en serverzijde aangeven en welke boodschappen er uitgewisseld worden).***




Een TCP-verbinding start altijd in 'gesloten' toestand. We maken een onderscheid tussen *active open* en *passive open*:

*active open*: We zeggen dat een toepassing active open is, als die zelf de verbinding initieert: de client zal een SYN-bericht sturen, en zo in de toestand SYN_SENT terecht komen. De server zal antwoorden op dat SYN-bericht met een ACK-bericht, dat op zijn beurt ook een SYN-bericht bevat. De client stuurt een ACK-bericht terug, en op die manier is de verbinding tot stand gebracht. We zien hier het 3-way-handshake-protocol aan de client-zijde.

*passive open*: We zeggen dat een toepassing passive open is, als die wacht tot dat iemand wil verbinding maken met die toepassing; typipsch: server is passive open, wacht totdat client verbinding maakt. Als de server een SYN-bericht ontvangt, zal die een ACK-bericht, met daarin eveneens een SYN-bericht, terugsturen. Op die manier komt de server in de SYN_RCVD toestand. Als die dan ACK-bericht ontvangt van de client, is de verbinding opgezet.

Eens de ESTABLISHED toestand bereikt is, zullen langs beide kanten de nodige buffers en variabele voorzien zijn. De client kan een bestand opvragen, en zal de server antwoorden. Het aangevraagde bestand wordt dan stuk per stuk verstuurd via TCP-segmenten.
Bij het afsluiten van de verbinding maken we opnieuw een onderscheid, deze keer tussen *active close* en *passive close*:

*active close*: Bij active close bekijken we de kant die de sluiting aanvraagt. Dit gebeurt door een FIN-bericht te versturen, op die manier komt de verbinding in een FIN_WAIT_1 toestand. Nu zijn er twee mogelijkheden: ofwel krijgt men een ACK-bericht binnen, waarin de FIN-vlag reeds aan staat, ofwel krijgen we een gewoon ACK-bericht, waarbij de FIN-vlag nog uit staat. In het eerste geval komen we direct in de TIME_WAIT toestand, in het tweede geval komen we eerst nog in een tussentoestand terecht, waarin we wachten tot er een FIN-bericht toekomt voor we overgaan naar de TIME_WAIT toestand. In beide gevallen zullen we bij het overgaan naar de TIME_WAIT toestand een ACK-bericht versturen, waarmee we aangeven dat we begrepen hebben dat de verbinding afgesloten is. We blijven een aantal seconden in die toestand, totdat de verbinding definitief gesloten wordt.

*passive close*: Bij passive close bekijken we de kant die een FIN-bericht ontvangt, omdat de andere kant de verbinding wil afsluiten. Bij het binnenkrijgen van zo'n FIN-bericht, versturen we een ACK-bericht, waardoor we in een CLOSE_WAIT toestand terechtkomen. Daarin wordt de eigen kant van de verbinding afgesloten, en wordt een FIN-bericht gestuurd naar de andere kant. We komen dan in de LAST_ACK toestand, waar we wachten op een ACK-bericht. Eens die ACK ontvangen is, kunnen we de verbinding definitief afsluiten.

Nadat de verbinding afgesloten is, zijn de buffers en variabelen die gebruikt werden, opnieuw vrijgegeven.

3.4 ***Bespreek: acknowledgment, timeout retransmit, duplicate reception, piggybacking, delayed ack, accumulated ack, selective retransmit, fast retransmit, retransmission timer, retransmission time-out, measured round trip time.***

**acknowledgment**

Een acknowledgement (ACK) is een bericht dat aangeeft dat een verstuurd bericht van de andere kant aangekomen is. TCP maakt gebruik van cumulatieve bevestigingen, waardoor het bevestigingsnummer aangeeft dat alle voorgaande bytes goed ontvangen zijn (geen gaten).

**timeout retransmit**

TCP maakt gebruik van een timeout mechanisme. Indien voor een verzonden segment geen ACK-bericht ontvangen werd binnen de timeout, dan wordt het segment opnieuw gestuurd.

**duplicate reception**

Een duplicate reception is een segment dat 2 keer ontvangen werd. Dit kan gebeuren doordat een ACK-bericht verloren ging, waardoor een timeout retransmit gebeurde. Een dergelijk segment wordt genegeerd, en het ACK-bericht wordt opnieuw verstuurd.

**piggybacking**

Een ACK-bericht dat 'meereist' met een segment wordt een piggybacked ACK genoemd. Bij de 3-way-handshake kan het laatste ACK bericht meereizen met het eerste segment dat verstuurd wordt.

**delayed ack**

Een delayed ACK is een ACK-bericht waarmee men even wacht om het te versturen, omdat er mogelijk nog andere segmenten binnenkomen. Door de cumulatieve bevestigingen die gebruikt worden door TCP is het voldoende om enkel een ACK-bericht te versturen voor het laatst verkregen segment. Een dergelijke ACK wordt maximaal 200 msec uitgesteld, omdat dit anders teveel vertraging oplevert.

**accumulated ack**

Een accumulated ACK is een ACK-bericht dat na een bepaald aantal ontvangen segmenten direct verstuurd wordt, en is dus een delayed ack die niet langer wacht dan max. 200 msec, of niet langer dan dat aantal binnengekomen segmenten.

**selective retransmit**

Men spreekt van selective acknowledgement indien een TCP-ontvanger in staat is om segmenten die in verkeerde volgorde binnenkomen selectief te bevestigen i.p.v. cumulatief. Een selective retransmit is dan het opnieuw verzenden van segmenten die geen nog geen bevesting ontvingen (cumulatieve of selectieve bevestiging).

**fast retransmit**

Indien de TCP-verstuurder meerdere berichten na elkaar stuurt, en de ontvanger merkt dat een van de tussenliggende segmenten niet aangekomen is, zal die 3 ACK-berichten na elkaar sturen, met als bevestigingsnummer het volgnummer van het segment dat verloren ging. Op die manier

herkent de verzender dat een segment niet aangekomen is, en zal die onmiddelijk het segment dat verloren ging opnieuw sturen. Als dat bericht dan wel aangekomen is, zal men (als men de vorige segmenten bijgehouden heeft), kan men een accumulated ACK sturen, om aan te geven dat alle segmenten aangekomen zijn.

**retransmission timer**

Via de retransmission timer beslist men hoelang men wacht tot men een segment opnieuw stuurt, als men binnen de timeout geen ACK-bericht ontvangen heeft.

**retransmission time-out**

Met retransmission timeout (RTO) bedoelt de tijd dat men wachten totdat een segment opnieuw gestuurd wordt, indien er binnen die timeout geen ACK-bericht toegekomen is. Om steeds een goede waarde voor de RTO te hebben, wordt die dynamisch aangepast volgens een formule.

**measured round trip time**

Measured round trip time is de tijd die gemeten wordt totdat de verzender een acknowledgement krijgt voor een verstuurd segment (M in de formule voor RTT).

3.5 ***Hoe berekent met de RTO? En hoe meet men de round trip time M?***

Men berekent de RTO (retransmission timeout) met volgende formule:

$$RTO = RTT + 4D$$

waarbij

$$D = \beta D + (1 - \beta)|RTT - M|$$

en waarbij men $\beta$ zelf kiest (meestal waarde tussen 3/8 en 7/8).
De RTT stelt de geschatte round trip time voor, die berekend wordt volgens de formule

$$RTT = \alpha RTT + (1 - \alpha)M$$

. In deze formule is RTT in het linkerlid de oude waarde (initieel bvb. 0s), $\alpha$ een waarde die men zelf kiest (meestal 7/8) en M de round trip time is die wordt bepaald als de tijd die gemeten wordt totdat de verzender een acknowledgement krijgt voor een verstuurd segment.
Als initiele waarden neemt men $RTO = 3s, D = 1.5 sen RTT = 0s$.

3.6 ***Bespreek het principe van flow control in TCP. Leg in detail uit a.d.h.v. diverse "windows". Waarom wordt flow controle gebruikt?***

Flow control is een mechanisme dat ervoor zorgt dat de segmenten die verstuurd worden allemaal kunnen verwerkt worden door de ontvanger, dus dat de buffer van de ontvanger niet overloopt.
Om dit op te lossen zal de ontvanger ervoor zorgen dat de snelheid waarmee de zender de segmenten verstuurd zal aangepast worden, volgens de plaats die de ontvanger heeft in zijn buffer.

De ontvanger zal de mate waarin zijn buffer gevuld is meten, en de zender het aantal vrije plaatsen in zijn buffer melden, via de ACK-berichten die teruggestuurd worden (via het window grootte veld in het TCP-segment). De zender zal dan uitgaande verkeer beperken volgens het aantal vrije plaatsen in de buffer van de ontvanger. De zender zal zijn send window (het aantal bytes dat zal verzonden worden) bij ieder ACK-bericht dat men krijgt, aanpassen volgens de waarde van de window grootte in het ACK-bericht, volgens volgende formule:

$$send\ window = receive\ window$$

$$-aantal\ verzonden\ maar\ nog\ niet\ bevestigde\ bytes$$

waarbij *receive window* het aantal vrije plaatsen is dat wordt aangegeven in het ACK-bericht, en het *aantal nog niet bevestigde bytes* gelijk is aan het verschil tussen de laatst verzonden byte en de laatst bevestigde byte. De ontvanger zal zijn *receive window* aanpassen telkens als de applicatie segmenten uit de buffer haalt. De volgende voorwaarde moet steeds voldaan zijn, opdat de buffer van de ontvanger niet zou overlopen

$$laatste\ verzonden\ byte - laatst\ gelezen\ byte \leq ontvangstbuffer$$

Indien de ontvanger aangeeft dat zijn buffer volledig vol is, zitten we met een probleem: de zender mag niets meer sturen, waardoor de server ook geen antwoorden meer zal geven, en zal de overdracht niet verder gezet worden. Om te vermijden dat een dergelijke situati optreedt, zal de zender blijven segmenten doorsturen, die echter slechts 1 gegevensbyte bevatten. De ontvanger zal die segmenten bevestigen, en zo aangeven wanneer er opnieuw plaats zal zijn in de buffer, en wanneer de zender zijn snelheid mag verhogen.
Flow controle zorgt er enkel voor dat de buffer in de ontvanger niet overloopt, en controleert dus enkel de snelheid op de transportlaag.

### 3.7 ***Bespreek het principe van congestion control in TCP. Waarom wordt dit gebruikt?***

Bij congestion control zal de zender zijn snelheid aanpassen, omdat die merkt dat het netwerk verzadigd is. De transportlaag (TCP dus), zal besluiten dat het netwerk verzadigd is, als er veel segmenten opnieuw gestuurd moeten worden, of als de bevestiging van aangekomen segmenten te lang op zich laat wachten.
De zender bepaalt zijn *send window* met de volgende formule:

$$send\ window = min(receive\ window, congestion\ window)$$

$$-aantal\ verzonden\ maar\ niet\ bevestigde\ bytes$$

De zender maakt gebruik van het congestion-control-algoritme, TCP Reno, om het congestion window te bepalen op basis van de belasting op een bepaald moment. Het algoritme bestaat uit 3 belangrijke componenten:

- *additive-increase en multiplicative-decrease*

Als de zender een driedubbel ACK-bericht ontvangt, dan zal TCP de huidige waarde vna het congestion window halveren, en daarna telkens met één MSG verhogen, totdat het netwerk opnieuw overbelasting zal voortonen. Dit principe wordt additive-increase/multiplicative-decrease (AIMD) genoemd, en wordt ook congestion avoidance genoemd.

- *slow-startfase*

  Slow-start betekent dat het congestion window telkens verdubbeld wordt, totdat er pakketverlies zal optreden. Het begint met een congestion window van één MSG. Als het verstuurde segment bevestigd wordt, zal het congestion window met 1 MSG verhoogd worden. Als de twee segmenten bevestigd worden, zal het congestion window opnieuw verdubbeld worden. Voor elk bevestigd segment zal het congestion window telkens met 1 MSG verhoogd worden.
  Dit mechanisme zal gebruikt worden wanneer de verbinding tot stand gebracht wordt, omdat het lineair doen toenemen van de snelheid te traag zou gaan, omdat het netwerk bij het opzetten van een verbinding dikwijls een veel grotere bandbreedte heeft dan gebruikt wordt.

- *reactie op time-out gebeurtenissen*

  Er is reeds vermeld dat, wanneer de zender een driedubbel ACK-bericht ontvangt, die het congestion window zal halveren. TCP zal ook gebruik maken van een threshold variabele, om het meer complexe gedrag bij een time-out te kunnen regelen. Bij een time-out is het de bedoeling om het congestion window tot 1 MSG te verkleinen, daarna telkens te verdubbelen, tot de helft van de vorige waarde bereikt is, en daarna terug 1 MSG per keer te stijgen (lineaire stijging). Dit gebeurt als volgt: wanneer een packetverliesgebeurtenis optreedt, krijgt de threshold variabele een waarde die gelijk is aan de helft van de waarde van het congestion window. Bij het opstarten van de verbinding krijgt de threshold variabele typisch hoge waarde (meestal 65 kb).
  Men gaat dan als volgt te werk om beslissingen te nemen i.v.m. de waarde van het congestion window:

  - als het congestion window kleiner is dan de threshold, zal de zender zich in slow-startfase bevinden, en neemt de omvang van het congestion window exponentieel toe (telkens verdubbelen)
  - als het congestion window even groot is als de threshold, zal de verzender in congestion avoidance-fase terechtkomen, en neemt de omvang van het congestion window lineair toe
  - als de zender een driedubbel ACK-bericht ontvangt, wordt de threshold ingesteld op de helft van de huidige waarde het congestion window, en wordt de waarde van het congestion window ingesteld op de nieuwe waarde van de threshold
  - als er een time-out gebeurtenis optreedt, wordt de threshold ingesteld op de helft van de huidige omvang van het congestion window, en de omvang van het congestion window ingesteld op de waarde van 1 MSG

Het annuleren van de slow-startfase bij een driedubbel ACK-bericht heeft als reden dat ondanks dat er een pakket verloren gegaan is, de aankomst van drie maal hetzelfde ACK-bericht aangeeft de enkele segmenten (meer bepaald de drie segmenten na het verloren gegane segment) wel aangekomen zijn. Blijkbaar in het netwerk toch in staat om tenminste een paar segmenten af te leveren, zelfs al zijn andere segmenten door congestie verloren gegaan, en dit in tegenstelling tot een time-out. Dit principe wordt *fast recovery* genoemd.

3.8 ***Bespreek het verband tussen: send window, receiver window, congestion window.***

De zender zal de snelheid waarmee hij segmenten verstuurd (of eigenlijk het aantal segmenten dat verstuurd moet worden op een bepaald moment) bepalen aan de hand van de volgende formule:

$$send\ window = min(receive\ window, congestion\ window)$$

$$-aantal\ verzonden\ maar\ niet\ bevestigde\ bytes$$

Het congestion window wordt initieel ingesteld als 1 MSG, het receive window wordt bepaald aan de hand van het ACK-bericht dat ontvangen werd van de server tijdens het 3-way-handshake-protocol, de threshold krijgt typisch een hoge waarde (bvb. 65 kb/s). De waarde van het congestion window wordt aangepast volgens het congestion-control algoritme TCP Reno, en het receive window wordt telkens bepaald aan de hand van het laatst aangekomen ACK-bericht.
Op die manier wordt ervoor gezorgd dat de zender zijn snelheid aanpast aan de hoeveelheid segmenten die de ontvanger nog kan bijhouden in zijn buffer, en aan de belasting van het netwerk (meer bepaald het pad, m.a.w. de tussenliggende routers, naar de ontvanger).

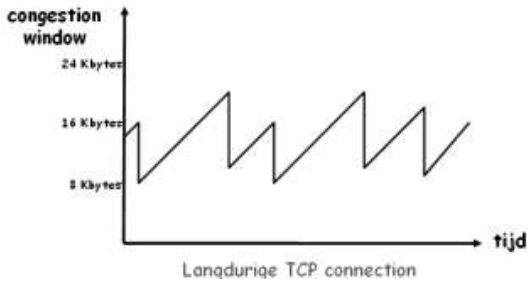
3.9 ***Hoe wordt congestion gedetecteerd en wat is de reactie (geen details)?***

De transportlaag (TCP dus), zal besluiten dat het netwerk verzadigd is, als er veel segmenten opnieuw gestuurd moeten worden, of als de bevestiging van aangekomen segmenten te lang op zich laat wachten.
De zender zal reageren door zijn snelheid van het versturen van segmenten aan te passen volgens het congestion-control algoritme TCP Reno.

3.10 ***Bespreek het principe van slow start en congestion avoidance.***

**congestion avoidance**

Congestion avoidance is een onderdeel van het congestion-control algoritme TCP Reno. Als de zender een driedubbel ACK-bericht ontvangt, dan zal TCP de huidige waarde van het congestion window halveren, en daarna telkens met één MSG verhogen, totdat het netwerk opnieuw overbelasting zal voortonen. Dit principe wordt additive-increase/multiplicative-decrease (AIMD) genoemd, en wordt ook congestion avoidance genoemd. Deze techniek geeft aanleiding tot een zaagtanddiagram:




**slow-startfase**

Slow-start betekent dat het congestion window telkens verdubbeld wordt, totdat er pakketverlies zal optreden. Het begint met een congestion window van één MSG. Als het verstuurde segment bevestigd wordt, zal het congestion window met 1 MSG verhoogd worden. Als de twee segmenten bevestigd worden, zal het congestion window opnieuw verdubbeld worden. Voor elk bevestigd segment zal het congestion window telkens met 1 MSG verhoogd worden.
Dit mechanisme zal gebruikt worden wanneer een TCP-verbinding tot stand gebracht wordt, omdat het lineair doen toenemen van de snelheid te traag zou gaan, omdat het netwerk bij het opzetten van een verbinding dikwijls een veel grotere bandbreedte heeft dan gebruikt wordt.

3.11 ***Bespreek het principe van fast retransmit en fast recovery.***

**fast retransmit**

Indien de TCP-verstuurder meerdere berichten na elkaar stuurt, en de ontvanger merkt dat een van de tussenliggende segmenten niet aangekomen is, zal die 3 ACK-berichten na elkaar sturen, met als bevestigingsnummer het volgnummer van het segment dat verloren ging. Op die manier herkent de verzender dat een segment niet aangekomen is, en zal die onmiddelijk het segment dat verloren ging opnieuw sturen. Als dat bericht dan wel aangekomen is, zal men (als men de vorige segmenten bijgehouden heeft), kan men een accumulated ACK sturen, om aan te geven dat alle segmenten aangekomen zijn. Dit principe noemt men *fast retransmit*.

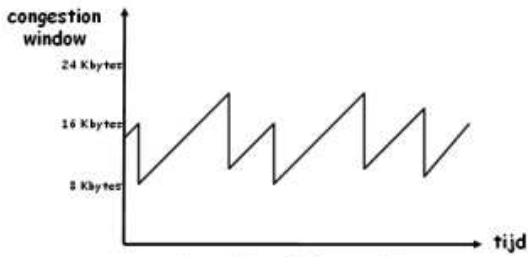
**fast recovery**

Het annuleren van de slow-startfase bij een driedubbel ACK-bericht, in het TCP Reno algoritme voor congestion-control, heeft als reden dat ondanks dat heeft als reden dat ondanks dat er een pakket verloren gegaan is, de aankomst van drie maal hetzelfde ACK-bericht aangeeft de enkele segmenten (meer bepaald de drie segmenten na het verloren gegane segment) wel aangekomen zijn. Blijkbaar in het netwerk toch in staat om tenminste een paar segmenten af te leveren, zelfs al zijn andere segmenten door congestie verloren gegaan, en dit in tegenstelling tot een time-out. Dit principe wordt *fast recovery* genoemd.

3.12 ***Leg uit: AIMD.***

AIMD is een onderdeel van het congestion-control algoritme TCP Reno. Als de zender een driedubbel ACK-bericht ontvangt, dan zal TCP de

huidige waarde van het congestion window halveren, en daarna telkens met één MSG verhogen, totdat het netwerk opnieuw overbelasting zal voortonen. Dit principe wordt additive-increase/multiplicative-decrease (AIMD) genoemd, en wordt ook congestion avoidance genoemd.
Deze techniek geeft aanleiding tot een zaagtanddiagram:



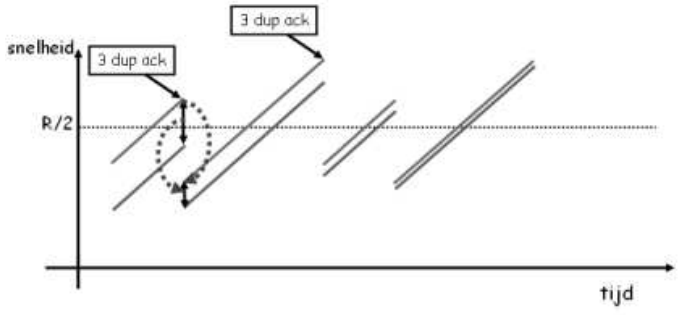
Langdurige TCP connection

3.13 ***Leg uit hoe TCP "fairness" ondersteunt. Geef een voorbeeld hoe men dat kan omzeilen.***

Het doel van *fairness* is dat als $K$ TCP sessies dezelfde bottleneck (een verbinding in het netwerk waarlangs alle TCP verbindingen moeten passeren, en die beduidend minder verzendcapaciteit heeft als de andere links) hebben, die een bandbreedte $R$ heeft, ze dezelfde gemiddelde snelheid $\frac{R}{K}$ hebben.
We bekijken hoe TCP ervoor zorgt dat alle verbindingen ongeveer dezelfde gemiddelde bandbreedte tot hun beschikking hebben. We doen dit aan de hand van een grafiek.
We merken eerst op dat de snelheid van een bepaalde verbinding gegeven wordt door :

$$\frac{CongWin}{RTT}\frac{Bytes}{sec}$$

We merken ook op dat $R$ in de grafiek duidt op de verzendcapaciteit van de bottleneck, en dat er voor de eenvoud slechts 2 TCP sessies veronderstelt worden, en bovendien dat ze dezelfde MSG en RTT hebben, dat ze een grote hoeveelheid gegevens willen versturen, en de link geen andere TCP-verbindingen of UDP- datagrammen verwerkt. We laten ook de slow-startfase buiten beschouwing, en nemen aan dat TCP steeds probeert om overbelasting te voorkomen.




Het blijkt duidelijk uit de grafiek dat, telkens een sessie een te hoge bandbreedte zal gebruiken, waardoor de zender een driedubbele ACK zal ont-

vangen, de bandbreedte van die sessie gehalveerd zal worden door het AIMD-principe. Daardoor het verschil in bitrate tussen beide sessies ook steeds gehalveerd worden, waardoor op den duur de gemiddelde snelheid van beide sessies hetzelfde zal zijn.

3.14 ***Waarvoor worden TCP en UDP gebruikt? Geef enkele voorbeelden.***

**UDP**

UDP is connection-less (geen handshaking) en onbetrouwbaar (segmenten kunnen verloren gaan of niet in volgorde aankomen), buiten dan de beperkte foutcontrole (door de UDP checksum). Dit is voor sommige toepassingen een voordeel, gezien dit de snelheid ten goede komt, en UDP bovendien geen limiet voorziet voor de snelheid, wat wel gebeurt bij TCP. Voorbeelden waarvoor het UDP-protocol gebruikt wordt: streaming multiemedia, DNS, SNMP.

**TCP**

TCP is connection-oriented (3-way-handshake), end-to-end, full-duplex (communicatie mogelijk in 2 richtingen tegelijk), betrouwbaar (acknowledgements, retransmissies, aflevering in volgorde, foutcontrole) en levert flow controle (vermijden dat buffer van ontvanger overloopt) en congestion controle (zorgt ervoor dat netwerk niet constant overbelast is). Voorbeeld waarvoor het UDP-protocol gebruikt wordt: HTTP, FTP, SMTP, Telnet, POP3, IMAP, ...

# Hoofdstuk 4

# Netwerklaag

4.1 ***Bespreek de verschillende IP-adresklassen. Geef enkele speciale adressen.***

We bekijken eerst de vorm van een IP-adres. Een IP-adres bestaat uit 32 bits, en wordt genoteerd als 4 getallen tussen 0 en 255, gescheiden door een punt. Een IP-adres heeft een netwerkdeel en een hostdeel.
We onderscheid 5 IP-adresklassen:


In een IP-adres van klasse A geven de eerste 8 bits (waarbij men de eerste bit vastlegt als 0) het netwerk en de laatste 24 bits de interface binnen dat netwerk aan. In klasse A is er dus ruimte voor $2^7$ netwerken, elk met maximaal $2^24$ interfaces.
In klasse B geven een IP-adres de eerste 16 bits het netwerk aan (waarbij de eerste twee bits steeds de waarden 1 en 0 resp hebben), en de laatste 16 de interface binnen dat netwerk. Er is dus ruimte voor maximaal $2^14$ netwerken, elk met maximaal $2^16$ interfaces.
Een IP-adres van de klasse C, geven de eerste 24 bits het netwerk aan (waarbij de eerste3 bits steeds de waarden 1, 1, 0 resp. hebben), en de laatste 8 bits de interface binnen dat netwerk. In deze klasse is er dus ruimte voor $2^21$ netwerken, elk met $2^8 interfaces$.
Een IP-adres in de klasse D wordt een multicastadres genoemd. De eerste 4 bits van het adres liggen vast (1,1,1,0), de andere 28 bepalen het multicastadres.
Tenslotte is er nog een klasse E, die bedoeld is voor toekomstig gebruik. Een dergelijk adres begint steeds met de bitsequentie 11110.

We bekijken nog enkele speciale adressen:

- een IP-adres van de vorm X.Y.0.0 duidt op een netwerk;
  vb$^n$: 15.0.0.0 (klasse A netwerk), 157.193.0.0 (klasse B netwerk), 193.125.97.0 (klasse C netwerk)
- een IP-adres van de vorm 127.X.Y.Z wordt een loopback interface genoemd, die gebruikt wordt voor debugging; in de praktijk wordt vooral 127.0.0.1 gebruikt
- 0.0.0.0 duidt op de host zelf in dit netwerk, en is enkel toegelaten als bronadres
- 0.0.X.Y duidt op een bepaalde host in dit netwerk, en is eveneens enkel toegelaten als bronadres;
- 255.255.255.255 wordt gebruikt als broadcast adres voor dit netwerk; dit adres wordt enkel gebruikt en bestemmingsadres, en forwarding ernaartoe is niet toegestaan
- X.Y.255.255 wordt gebruikt om alle hosts in een ander netwerk aan te duiden, en is enkel toegelaten als bestemmingsadres
- IP-adressen die in volgende intervallen liggen, worden typisch gebruikt in private netwerken, die niet verbonden zijn met het internet: 10.0.0.0 - 10.255.255.255, 172.16.0.0 - 172.31.255.255, 192.168.0.0 - 192.168.255.255

4.2 ***Bespreek het principe van direct connected networks en subnetworks.***

Een groot netwerk kan bekeken worden als verschilende kleinere netwerken, die onderling direct met elkaar verbonden zijn ("direct connected network").
Een betere manier van werken, is met subnetwerken, dat zijn verschillende direct connected networks, die onderling verbonden worden m.b.v. routers, switches, ... Een groot netwerk wordt dan ingedeeld in verschillende subnetwerken, die elk een eigen netwerk adres hebben. Een netwerk met als IP-adres 157.193.0.0 kan dan onderverdeeld worden in subnetwerken, bijvb. 157.193.103.0 , 157.193.60.0, 157.193.234.0, 157.193.40.0, 157.193.227.0. Een dergelijke subnetwerk kan hosts bevatten, of enkel een verbinding vormen tussen 2 routers. Een belangrijk punt is dat subnetwerken nooit mogen overlappen.

4.3 ***Bespreek subnet adressering. Geef een voorbeeld.***

Met subnetadressing duidt men aan welk deel van het IP-adres het netwerk aanduidt, welk deel het subnetwerk en welk deel de host. Dit doet men door een subnetmask mee te geven (typsich in hexadecimale notatie), die bestaat uit een reeks 1 bits gevolgd door een reeks 0 bits. De eerste 0 bit geeft aan waar het hostdeel van het IP-adres begint.
We bekijken een voorbeeld:
We krijgen een IP-adres 158.78.42.64, met als gegeven subnetmask FF.FF.FF.C0. Er wordt gegeven dat het netwerk waarvan dit subnetwerk onderdeel is, aangeduid wordt met 158.78. Als we het subnetmask schrijven is gedeeltelijk binaire notatie ( FF.FF.FF.1100 0000 ) merken we dat er 6 bits vrijgehouden zijn voor het aanduiden van de hosts binnen dat subnetwerk. Er worden dus 10 bits gebruikt voor het subnetwerk,

waardoor er $2^{10} - 2 = 1022$ mogelijke subnetten zijn (0000 0000 00 en 1111 1111 11 zijn niet toegelaten). De hosts gebruiken 6 bits, dus zijn er $2^6 - 2 = 62$ mogelijke hosts (opnieuw, binnen ieder subnetwerk mag een host niet aangeduid worden met 00 0000 of 11 1111).
We merken nog op dat een netwerk, subnetwerk of een host nooit mag aangeduid worden met enkel 0- of enkel 1-bits. Met behulp van een vergelijking tussen de logische AND van het bestemmingadres en het subnetmask, en de logische AND van het eigen IP-adres en het subnetmask, kan men beslissen of de bestemmingshost in het eigen netwerk ligt, of erbuiten (zodat het pakket via een router moet passeren).

4.4 ***Bespreek CIDR. Geef een voorbeeld.***

Een nadeel met het indelen van IP-adressen volgens klassen is dat veel adressen niet gebruikt worden. Een bedrijf dat bijvoorbeeld 2000 adressen nodig had in zijn netwerk, kreeg automatisch een klasse B IP-adres toegewezen, wat een maximale capaciteit heeft van 65534 interfaces. Daardoor gingen veel IP-adressen verloren.
Een oplossing daarvoor is gebruik maken van Classless InterDomain Routing (CIDR). Daardoor is het mogelijk om het de lengte van het netwerkgedeelte in een IP-adres zelf te kiezen. Een IP-adres in CIDR-notatie is van de vorm a.b.c.d/x, waarbij x staat voor het aantal bits in het netwerk gedeelte van het adres.
Voorbeeld: 200.23.16.0/23, dat overeenkomt met:

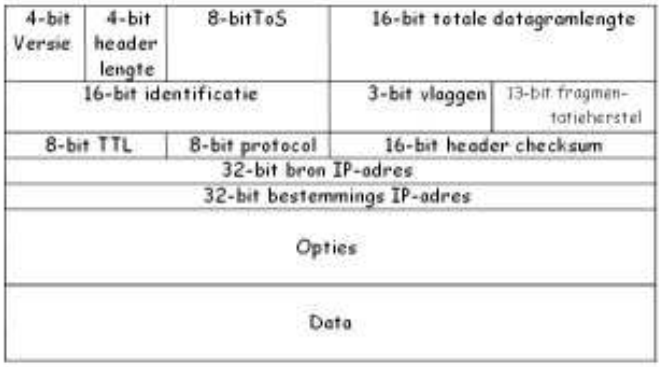
**1100 1000 0001 0111 0001 000**0 0000 0000

waarbij het netwerkgedeelte **vet** gedrukt is.

4.5 ***Bespreek het verschil tussen routing en forwarding.***

Onder **routing** verstaat men het vullen van de routeringstabellen van een router, op basis van een gecentraliseerde of verdeelde berekening van het kortste pad of afstand tussen de router en het bestemmingsnetwerk.
Met **forwarding** daarentegen bedoeld men het doorsturen van pakketjes, gebaseerd op het bestemmingsadres en de inhoud van de routeringstabel.

4.6 ***Bespreek de verschillende velden van een IP-datagram (het datagram zelf wordt gegeven op het examen).***



**Versienummerveld**

Dit veld duidt aan welke protocol versie gebruikt wordt in het datagram (IPv4 of IPv6). Dit is nodig, omdat de verschillende versies verschillende datagramindelingen gebruiken.

**Headerlengteveld**

Omdat een IPv4 datagram een variabel aantal opties kan bevatten, is het met dit veld nodig om aan te geven waar de gegevens juist beginnen in het datagram. Als het datagram geen opties bevat, zal de header 20 bytes lang zijn.

**ToS-veld (Type of Serive)**

Deze bits worden gebruikt om verschillende types datagrammen aan te duiden. Het kan bijvoorbeeld wenselijk zijn om een onderscheid te maken tussen netwerkbeheerdatagrammen en datagrammen die gegevens bevatten (bvb. bij overbelasting van het netwerk), of om een onderscheid te maken tussen realtime-datagrammen en niet-realtime-datagrammen.

**Datagramlengteveld**

Dit veld duidt de totale lengte van het IP-datagram aan. De theoretische maximale afmeting is 65.535 bytes, omdat er 16 bits gebruikt worden voor dit veld, maar in de praktijk zal een datagram slechts 1500 bytes of zelfs slechts 576 bytes bevatten.

**Identificatie-, vlag- en fragmentatieherstelvelden**

Deze velden worden gebruikt bij de IP-fragmentatie.

**TTL (Time-to-Live)**

Dit veld wordt gebruikt om te vermijden dat een datagram eindeloos zou kunnen blijven rondzwerven, bvb. door een routerlus, in het netwerk. Telkens het datagram een router passeert, wordt de waarde van dit veld met 1 verlaagd.

**protocolveld**

Dit veld wordt enkel gebruikt als het IP-datagram de bestemming bereikt heeft, en bepaalt het transportlaagprotocol op de bestemming dat het gegevensdeel van het datagram zal ontvangen. We geven enkele voorbeelden van veel gebruikte waarden in dit veld:

1: ICMP

4: IP-in-IP

6: TCP

17: UDP

89: OSPF

**headerchecksumveld**

Met behulp van dit veld kan een router bitfouten in een ontvangen IP-datagram detecteren. De headerchecksum wordt berekend door elke twee bytes in de header als 1 getalte beschouwen, waarnee deze met het één-complement worden opgeteld. Elke router zal de checksum opnieuw berekenen, en controloren of die gelijk is aan de meegegeve checksum. Indien dit

niet zo is, wordt het datagram meestal verwijderd.
De reden dat zowel in TCP en UDP als in IP aan foutcontrole gedaan wordt, is dat niet alle protocollen die gebruik maken van IP zelf aan foutcontrole doen, en dat TCP en UDP niet mogen rekenen op de routers om de foutcontrole te verzorgen.

**bron- en bestemmingsadres**

Deze velden bevatten de 32-bit IP-adressen van de bron en de eindbestemming van het datagram.

**optiesveld**

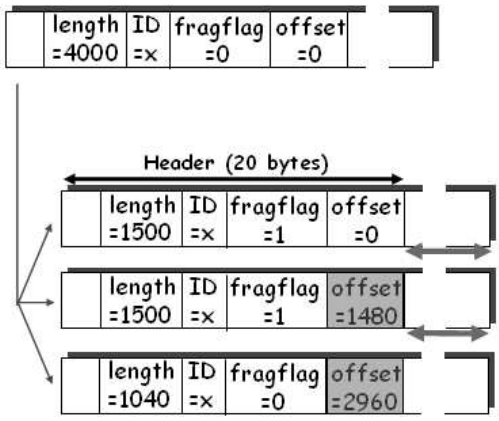
In dit velden worden mogelijke opties vermeld. Dit optieveld zorgt ervoor dat de headerlengte niet constant is, waardoor de verwerkingstijd in de routers sterk kan variëren. Daarom werd het optieveld in IPv6 weggelaten.

**gegevensveld**

Dit veld bevat de eigenlijke gegevens die verstuurd worden met het datagram. Meestal zal dit gaan om een UDP- of TCP-segment dat bij de bestemming moet worden afgeleverd, maar het kan ook gaan om andere gegevens, zoals ICMP-berichten.

4.7 ***Bespreek fragmentatie.***

Iedere netwerklink heeft een waarde voor de MTU (Maximum Transfer Unit) van die link. Die geeft weer hoe groot een datagram maximaal mag zijn om doorgestuurd te worden via de link (niet kleiner dan 576 bytes). Verschillende types van links, zullen verschillende MTU-waarden hebben. Als een groot IP-datagram toekomt, zal de router dat datagram fragmenteren in verschillende kleinere (kleiner dan de MTU-waarde) datagrammen. Op de eindbestemming zullen die fragmenten dan opnieuw samengevoegd worden tot het originele datagram dat verstuurd wordt, waarna het doorgegeven wordt aan de transportlaag.
Opdat de eindbestemming de fragmenten correct (in de juiste volgorde, zonder tekorten) terug zou kunnen samenstellen, maakt men gebruik van de header identificatie-, vlag- en fragmentatieherstelbits.
Wanneer een datagram wordt aangemakt, zal de verzendende host een waarde voor het identificatienummer in de header plaatsen. Die waarde zal voor alle fragmenten waarin dat datagram opgedeeld wordt, hetzelfde zijn. Alle fragmenten zullen een vlagbitwaarde 1 hebben, uitgenomen het laatste fragment, wat een waarde 0 zal hebben voor zijn vlagbit. Op die manier weet de ontvanger dat het om het laastste fragment gaat. Om ervoor te zorgen dat de ontvangende host kan bepalen of er fragment ontbreekt (en om de volgorde van de fragmenten te bepalen), wordt een offset veld gebruikt (fragmentatieherstelveld). Die offset is steeds een veelvoud van 8, en de lengte van de header wordt niet meegerekend.
We bekijken als voorbeeld een IP-datagram van 4000 bytes (20 bytes header), dat door link gestuurd moet worden die als MTU-waarde 1500 heeft:




4.8 ***Wat is ICMP? Geef een voorbeeld bij het gebruik in een redirect en traceroute.***

ICMP (Internet Control Message Protocol) is een protocol dat door hosts, routers en gateways gebruikt wordt om netwerkinformatie met elkaar uit te wisselen, en wordt vooral gebruikt voor het melden van fouten. ICMP is geen onderdeel van IP, want het maakt net zoals TCP en UDP gebruik van IP-datagrammen.
ICMP-berichten hebben een type- en een codeveld en bevatten ook de eerste8 bytes van het IP-datagram dat de oorzaak was dat het ICMP-bericht werd gegenereerd, zodat de verzender kan vaststellen welk pakket de fout veroorzaakte. We bekijken enkele ICMP-berichten met hun type, code en omschrijving:

type: 0, code: 0, omschr.: antwoord (echo) (naar ping)

type: 3, code: 0, omschr.: bestemmingsnetwerk onbereikbaar

type: 3, code: 1, omschr.: ontvangende host onbereikbaar

type: 3, code: 2, omschr.: bestemmingsprotocol onbereikbaar

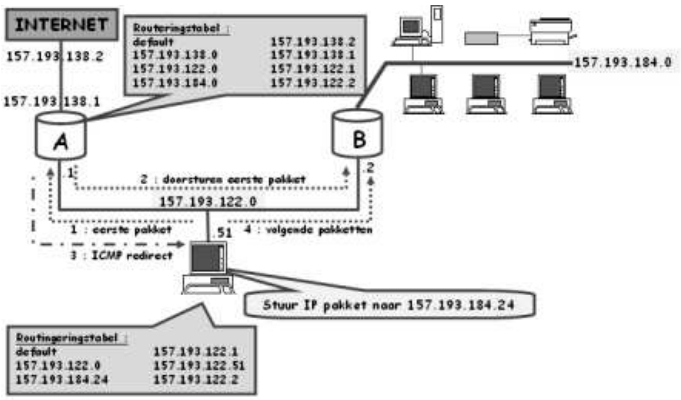
type: 3, code: 3, omschr.: bestemmingspoort onbereikbaar

type: 8, code: 0, omschr.: verzoek om te antwoorden (ping)

type: 11, code: 0, omschr.: TTL bereikt

ICMP wordt gebruikt bij *traceroute* op de volgende manier. Er wordt telkens een IP-datagram verstuurd, de eerste met als TTL-waarde 1, de tweede als TTL-waarde 2, enz... Wanneer een router merkt dat de TTL van het datagram bereikt is, zal de een ICMP-waarschuwingsbericht (type 11, code 0) naar de bron sturen, dat de naam en het IP-adres van de router bevat. Wanneer het ICMP-bericht aankomt op de bron, bepaalt die de round-triptijd aan de hand van een timer die gestart werd toen het datagram verstuurd werd. Op die manier geeft traceroute de route weer die gevolgd wordt naar een bepaalde bestemming.

Een andere toepassing van ICMP is redirect. We bekijken dit aan de hand van een voorbeeld:




In dit voorbeeld zal de host een reeks IP pakketten willen versturen naar het IP-adres 157.193.184.24. Het eerste pakket dat verstuurd wordt, gaat naar router A, omdat dit zo in de routertabel van de host staat vermeld. Die router ziet echter dat hij het binnengekomen pakket moeten verzenden via dezelfde interface waarlangs die ontvangen werd. Daarom zal router A een ICMP redirect boodschap sturen naar de host, waarop die zijn routertabel zal aanpassen. Op die manier worden alle volgende pakketten rechtstreeks naar router B verstuurd, en niet meer via router A (een omweg).

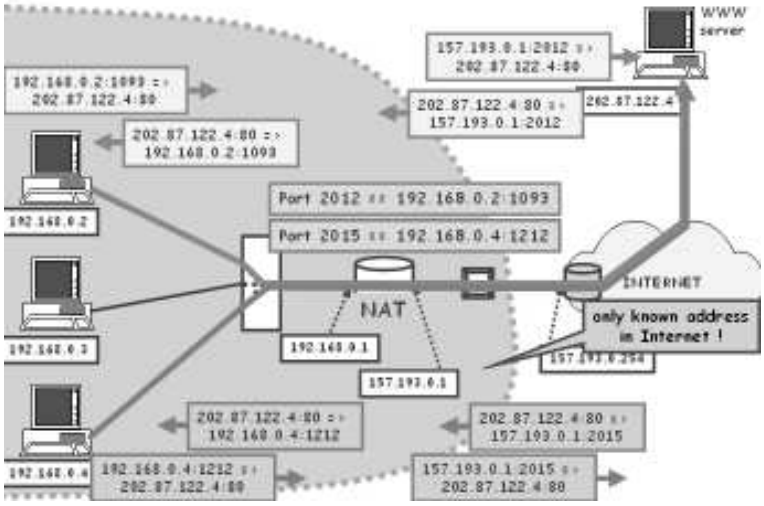
4.9 ***Bespreek NAT. Geef een voorbeeld.***

De bedoeling van NAT (Network Address Translation) is dat een lokaal netwerk slechts 1 IP-adres gebruikt zoals het gezien wordt door de buitenwereld. Dit systeem heeft enkele voordelen:

- het is niet nodig om een interval van adressen aangewezen te krijgen van de ISP, want er wordt slechts één IP-adres gebruikt voor alle hosts
- de adressen van de hosts in het lokale netwerk kunnen gewijzigd worden zonder dat de buitenwereld moet ingelicht worden
- de ISP kan gewijzigd worden zonder dat de adressen van de hosts in het netwerk moeten gewijzigd worden
- de hosts in het lokale netwerk zijn niet expliciet adreseerbaar door de buitenwereld, wat een voordeel is bij beveiliging

Om dit te kunnen verwezenlijken, zal de NAT-router ieder datagram dat buitengaat aanpassen: ieder bronadres zal gewijzigd worden in het adres van de NAT-router, een de poort die gebruikt wordt zal ook gewijzigd worden. De NAT-router zal een tabel bijhouden waarin staat welke poorten gemapt worden op welke poorten op welk (lokaal) IP-adres. Op die manier zal de NAT-router de inkomende datagrammen eveneens aanpassen (poort en bestemmingsadres), zodat die op de juiste lokale host terechtkomen. Doordat de NAT-server gebruikt maakt van een 16-bit poortnummer, is het mogelijk om 65.535 gelijktijdige verbindingen te verwerken met één enkel LAN-side IP-adres.

We bespreken het voorbeeld op de figuur.
De NAT-router heeft als extern IP-adres 157.193.0.1. De tabel, die nu 2 rijen heeft, zorgt ervoor dat de pakketten die toekomen opde NAT-router doorgestuurd worden naar de juiste hosts in het lokale netwerk, en dat pakketten die vanuit de lokale hosts verstuurd worden aangepast worden, zodat antwoorden op die pakketten op de NAT-router terechtkomen.




Een pakket dat verstuurd wordt van de host met lokaal IP-adres 192.168.0.2 van poort 1093 naar het externe IP-adres 202.87.122.4, poort 80, wordt door de NAT-router aangepast naar een pakket met als bron IP-adres 157.193.0.1, poort 2012. De NAT-router zal een rij in zijn tabel aanmaken waardoor alle pakketten die toekomen op poort 2012, doorgestuurd worden naar de host met lokaal IP-adres 192.168.0.2, poort 1093.

4.10 ***Bespreek DHCP. Geef een voorbeeld.***

Het doel van het DHCP (Dynamic Host Configuration Protocol) protocol, is om een host toe te laten om dynamisch zijn IP-adres te bekomen van een netwerkserver, wanneer die host toetreedt tot het netwerk. Het laat eveneeens toe om IP-adressen te hergebruiken: een IP-adres zal enkel gebruikt worden zolang de host verbonden is, bij een nieuwe verbinding zal een nieuw IP-adres gegeven worden. Op die manier is er ondersteuning voor mobiele gebruikers die willen toetreden tot het netwerk.
Het verloop van het verkrijgen van een IP-adres van een DHCP-server is als volgt:

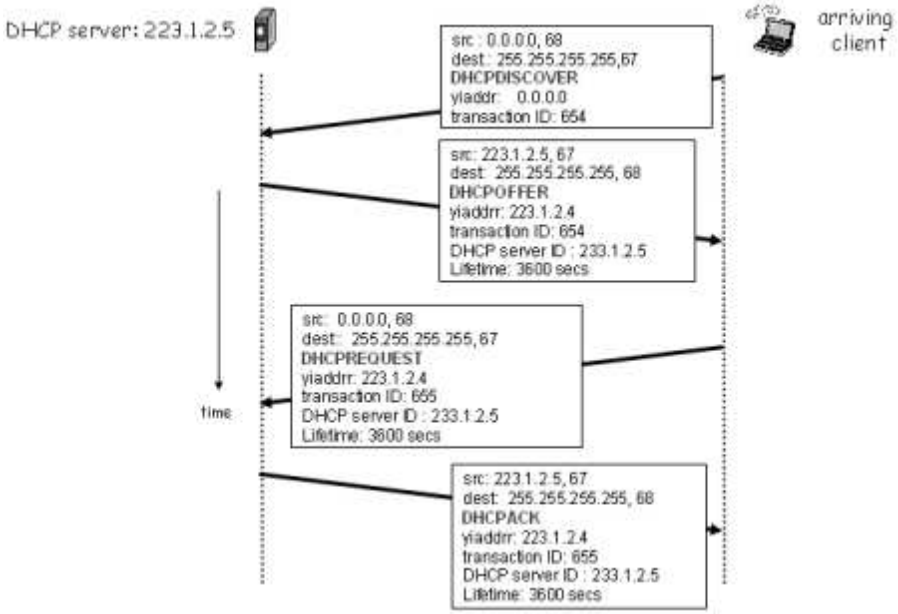
1) De host verstuurd doet verstuurd een DHCP discover-bericht (een UDP-datagram naar poort 67) naar het broadcast-adres 255.255.255.255, en geeft als bronadres 0.0.0.0 (want de host heeft nog geen eigen IP-adres)
2) De DHCP-server reageert op het discover-bericht met een DHCP offer-bericht naar de host die het discover-bericht verstuurde. Er kunnen meerdere servers zijn die reageren op met een offer-bericht, en dus zal de host een DHCP-server moeten kiezen. Het offer-bericht

bevat het transactienummer van het ontvangen discover-bericht, een IP-adres voor de eventueel toehappende client, het netwerkmasker en een leasetijd van het IP-adres (de tijd die het IP-adres gebruikt mag worden, typisch voor een periode van enkele uren tot enkele dagen).

3) De host maakt een keuze tussen de offer-berichten en antwoordt met een DHCP request-bericht naar de gekozen DHCP-server, waarin de configuratieparameters worden gemeld.

4) De DHCP-server beantwoordt het request-bericht met een DHCP ACK-bericht, en bevestigd daarmee de gevraagde parameters.

De standaard poorten voor DHCP zijn 67 en 68.
We bekijken een voorbeeld van het verkrijgen van een IP-adres van een DHCP-server.




De client doet een broadcast, en vraagt op die manier een IP-adres aan. Het discover-bericht wordt verstuurd van poort 68, en krijgt als bronadres 0.0.0.0. Het transaction id van het bericht is 654. De DHCP server antwoord met een offer-bericht waarin een IP-adres voor de host (223.1.2.4, yiaddrr = your internet address) aangeboden wordt, en met hetzelde transaction id als het discover-bericht. De client antwoordt dan met een request-bericht met de nodige configuratie-parameters. Als laatste antwoordt de DHCP server terug met een ACK-bericht, en bevestigd zo de configuratieparameters.

4.11 ***Wat is een AS? Geef 3 types (waarom is het zo belangrijk een onderscheid te maken).***

Een **autonoom systeem (AS)** is een verzameling routers onder hetzelfde beheer die onderling werken met hetzelfde padbepalingsprotocol. Er zijn 3 types autonome systemen:

- *strub AS*: kleine onderneming; één verbinding naar andere autonome systemen, dus kan er geen transit verkeer zijn
- *multi-homed AS*: grote onderneming , verschillende verbindingen naar andere autonome systemen, maar er is geen transit verkeer toegelaten
- *transit AS*: provider, die meerdere autonome systemen met elkaar verbindt, en uiteraard wel transit verkeer toelaat

Het onderscheid moet gemaakt worden omdat niet alle systemen transit verkeer toelaten.

4.12 ***Bespreek het verschil tussen intra- en inter-AS routering.***

**Intra-AS routing** is het routeren dat gebeurt tussen de routers van een autonoom systeem onderling. De beheerder van het autonoom systeem is verantwoordelijk voor de keuze van het routeringsalgoritme dat gebruikt wordt binnen het autonoom systeem. In verschillende autonome systemen kunnen verschillende routeringsalgoritme's gebruikt worden. Voorbeelden van dergelijke algoritme's zijn het Routing Information Protocol (RIP) en het Open Shortest Path First (OSPF) protocol.

**Inter-AS routing** is het routeren tussen autonome systemen onderling. Als routeringsalgoritme werd een standaard afgesproken, het Border Gateway Protocol (BGP).

Er zijn verschillende redenen waarom er verschillende inter-AS en intra-AS padbepalingsprotocollen gebruikt worden, waaronder *policy*-redenen (bij inter-AS routing geven policies de doorslag, het kan bvb. erg belangrijk zijn dat het verkeer van een bepaald AS een ander AS niet passeert), *schaal* (bij inter-AS padbepalingsalgoritmen is het belangrijk dat ze een groot aantal netwerken aankunnen, bij intra-AS algoritmen is dit minder belangrijk, omdat via OSPF een splitsing mogelijk is in verschillende areas) en *prestaties* (bij inter-AS padbepaling is de snelheid van paden van ondergeschikt belang, om die vooral gestuurd wordt door policy's; binnenin een AS streeft men er echter naartoe om de snelst mogelijke verbinding te vinden).

4.13 ***Bespreek het principe van distance vector en link-state routering. Geef een voorbeeld voor beide strategieën.***

**Distance vector routering**

Bij dit soort algoritmen wordt een afstandentabel bijgehouden op elke node. Om de 30s (bvb.) verstuurd elke node zijn afstandtabel naar zijn buren via een advertisement. Een dergelijk bericht bevat gegevens over maximaal 25 bestemmingsnetwerken, en aan de hand daarvan zullen de nodes die het bericht ontvangen eventueel hun tabel aanpassen. Ook wanneer van een bepaalde node gedurende een lange tijd (typisch 180s) geen bericht ontvangen werd, past die zijn tabel aan, omdat dan veronderstelt wordt dat de aangrenzende node niet meer bereikbaar is. Een voorbeeld van een dergelijk algoritme is het Routing Information Protocol (RIP).

**Link-state routering**

Bij een link-state routeringsalgoritme, bezit iedere router een overzicht van het netwerk in de vorm van een databank van kosten van de verschillende

links. Uit deze databank zal iedere router het korste pad bepalen, aan de hand van het algoritme van Dijkstra, naar alle bestemmingen. Aan de hand van die berekening zal de routeringstabel samengesteld worden. Op regelmatige tijdstippen zal een router zijn padbepalingsinformatie bekend maken aan alle routers in het AS. Een voorbeeld van een dergelijk algoritme is het Open Shortest Path First (OSPF) algoritme.

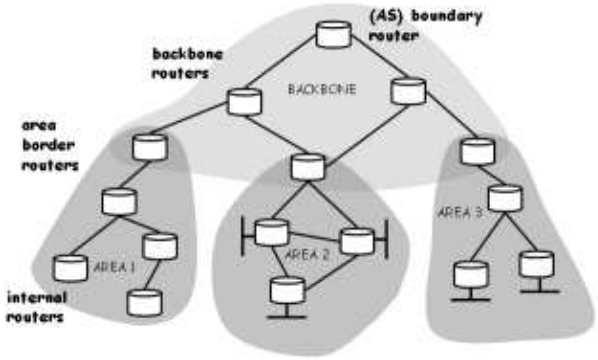
Bovendien heeft het OSPF algoritme nog enkele voordelen t.o.v. het RIP algoritme: beveiliging (alle OSPF berichten worden geauthenticeerd, zodat indringers de tabellen niet kunnen aanpassen), diverse paden met gelijke kosten zijn mogelijk (met OSPF kunnen de datagrammen via verschillende paden (met dezelfde kosten) worden verzonden naar de ontvanger), verschillende kosten kunnen bepaald worden voor datagrammen voor verschillende diensten, geïntegreerde ondersteuning voor unicast en multicast padbepaling, ondersteuning voor hiërarchie binnen een enkel padbepalingsdomein (in OSPF via area's).

4.14 ***Bespreek hierarchical OSPF. Waarom is dat nuttig?***

Bij *hierarchical Open Shortest Path First (OSPF)* wordt een autonoom systeem ingedeeld in verschillende area's. Elke area voert een eigen OSPF link state padbepalingsalgoritme uit waarbij elke router in een area zijn link state bekend maakt aan alle andere routers in die area. De interne gegevens van een area blijven dus onzichtbaar voor alle routers buiten die area. Voor intra-area padbepaling worden enkel de routers binnen dezelfde area gebruikt.
Binnen elke area zijn een of verschillende *area border routers* verantwoordelijk voor de padbepaling voor pakketten met een bestemming buiten de area. Er is één OSPF-area in het autonoom systeem geconfigureerd als *backbone*-area, waarvan de belangrijkste taak bestaat uit het bepalen van het pad voor pakketten die tussen de andere area's moeten verzonden worden. Die backbone-area bevat altijd alle area border routers in het autonoom systeem, en eventueel ook andere (niet-border) routers. Voor inter-area padbepaling binnen het AS moet het pakket eerst naar een area border router worden verzonden, en vervolgens via de backbone naar de area border router van de area waarin de eindbestemming aanwezig is. In een dergelijk autonoom systeem, onderscheiden we 4 soorten OSPF-routers:

- *interne routers*: deze routers in niet-backbonegebieden worden alleen gebruikt voor padbepaling binnen autonome systemen
- *area border routers*: deze routers horen zowel bij een area als bij een backbone
- *backbonerouters (non-border routers)*: deze routers bepalen paden binnen de backbone, maar zijn zelf geen area border routers; de interne routers binnen een non-backbone area krijgen informatie over paden naar andere area's van de backbonerouters binnen hun eigen area.
- *boundaryrouters*: deze routers wisselen padbepalingsinformatie uit met routers die behoren bij andere autonome systemen; ze kunnen

bvb. het Border Gateway Protocol (BGP) gebruiken voor de padbepaling tussen autonome systemen. De andere routers krijgen van dit type router informatie over externe netwerken.



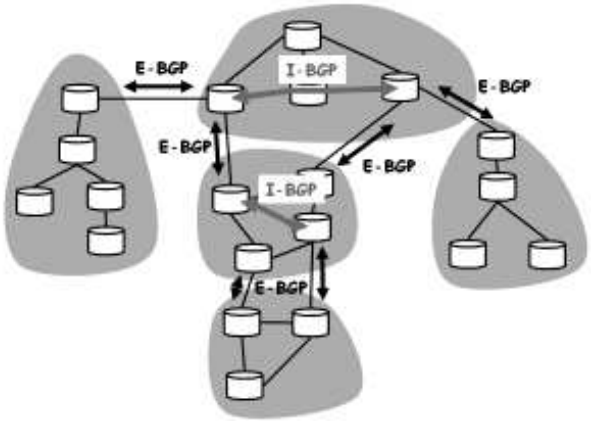

4.15 ***Bespreek een voorbeeld van BGP. Waarom heeft men I-BGP en E-BGP?***

*Border Gateway Protocol (BGP)* maakt de padbepaling tussen autonome systemen mogelijk. BGP is een gedistribueerd protocol omdat BGP-routers alleen communiceren met cd BGP-routers waarmee ze rechtstreeks verbonden zijn. BGP bepaalt enkel paden naar netwerken, want zodra het bestemmingsnetwerk bereikt is, wordt het padbepalingsprotocol van het intra-AS gebruikt om het pad van het datagram naar de eindbestemming ervan te bepalen. Elk autonoom systeem krijgt een uniek *autonoom systeem nummer (ASN)*. De kern van BGP bestaat uit padmeldingsberichten, die door de ene BGP-peer naar een andere BGP-peer verzonden worden via een point-to-pointverbinding. Een dergelijk bericht bestaat uit een netwerkadres(in CIDR) en een reeks attributen die horen bij het pad naar dat bestemmingsnetwerk, waarvan de twee belangrijkste het padattribuut (lijst van alle autonome systemen op het pad naar het bestemmingsnetwerk) en de identiteit van de volgende-hoprouter op dat pad naar het bestemmingsnetwerk.
De werking van BGP is op te delen in 3 activiteiten:

- *Het ontvangen en filteren van padmeldingsberichten van direct aangewezen buren*, die kunnen gezien worden als een belofte dat die buur die het bericht verstuurd, in staat zal zijn om het datagram verder te versturen op een pad naar die bestemming. Een BGP-router kan ook ontvangen padmeldingsberichten filteren (bvb. als het eigen ASN voorkomt in het pad, om lussen te vermijden). Een netwerkbeheerder kan veel invloed uitoefenen op de manier waarop datagrammen worden doorverzonden, omdat het volledig autonoom systeempad wordt gespecifieerd.
- *Het kiezen van een pad.* Een BGP-router kan verschillende padmeldingsberichten voor hetzelfde AS ontvangen, en zal moeten kiezen welk pad het best is. Slechts één volgende-hoprouter voor een bepaalde bestemming zal in de padbepalingstabel opgenomen worden. BGP specifieert echter nie hoe een autonoom de keuze voor een bepaald

pad moet maken, dat is een beslissing die door de netwerkbeheerder moet gemaakt worden. Die kan lokale voorkeuren instellen, zodat bvb. een bepaald aangrenzend AS voorkeur krijgt op een ander. Als er geen lokale voorkeuren ingesteld is, wordt de keuze meestal gemaakt op basis van het aantal autonome systemen op het pad (zo laag mogelijk).

- *Het verzenden van padmeldingsberichten naar aangrenzende autonome systemen.* Ook hier levert BGP enkel een mechanisme, zodat de netwerkbeheerder grote controle over het dataverkeer heeft dat het netwerk inkomt.




De bovenstaande vorm van BGP staat bekend onder de naam *Extern-BGP (E-BGP)*. Er is nog een andere vorm van BGP, *Intern-BGP (I-BGP)*, die gebruikt wordt om padbepalingsinformatie te verspreiden naar routers binnen het autonoom systeem met bestemming autonome systemen buiten het autonoom systeem. I-BGP kan gebruikt worden als optie voor het intra-AS padbepalingsprotocol.
I-BGP en E-BGP hebben dezelfde indeling voor het bericht- en padattribuut, maar hebben ook een aantal belangrijke verschillen. I-BGP routers hebben te maken met beperkingen in de manier waarop ze paden kunnen melden die van ze andere I-BGP routers ontvangen.
We merken nog op dat BGP de de facto-standaard is voor inter-AS padbepaling voor het publiekelijk toegankelijke internet. Het wordt bvb. gebruikt op de grootste *netwerk access points (NAP's)* die de verbinding vormen tussen grote ISP en waarop dataverkeer wordt uitgewisseld.

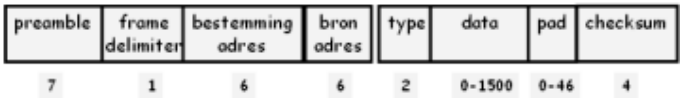
### 4.16 *Wat is een BGP "peering point"?*

BGP-peers zijn routers waartussen E-BGP gebruikt wordt. Een *peerig point* is een verzameling van dergelijke routers (die bijvb. allemaal verbonden zijn via een Ethernet switch). Een voorbeeld van een dergelijk peering point in België is BNIX (Belgian National Internet eXchange). Het is een plaats waar ISP's kunnen 'peeren', en in België zijn meer dan 40 ISP's ermee verbonden. De verbindingen vormen geen full mesh.

# Hoofdstuk 5

# Datalinklaag

5.1 ***Bespreek de verschillende velden van een Ethernet frame (het frame zelf wordt gegeven op het examen).***




Alle Ethernet-technologieën maken gebruik van dezelfde framestructuur.

- **preamble**
  De preamble wordt gebruikt om de klokken van de verzender en ontvanger te synchroniseren. Het bestaat uit 7 bytes, die allemaal hetzelfde zijn: 10101010.
- **frame delimiter**
  Deze byte wordt gebruikt om aan te geven dat bij de volgende byte het eigenlijke frame start. Deze byte maakt eigenlijk ook deel uit van de preamble, en bevat steeds de waarde 10101011.
- **bestemmingsadres**
  Dit veld bestaat uit 6 bytes, die het LAN-adres bevatten van de adapter van de bestemming. Indien een adapter een datagram bevat met een bestemmingsadres dat verschilt van zijn eigen MAC-adres, zal die het frame negeren.
- **bronadres**
  Dit veld (ook 6 bytes) bevat het bronadres van de adapter die het frame naar het LAN verstuurt.
- **typenummer**
  Dit veld bestaat uit 2 bytes die het type van de data bevat dat getransporteerd wordt door het frame. Dit is nodig omdat de adapter moet weten naar welk netwerklaagprotocol die de inhoud van het gegevensveld moet sturen. Ieder datalinklaagprotocol heeft een eigen gestandaardiseerd typenummer.
- **data**

Dit veld bevat de informatie die verstuurd wordt (IP-datagram). De MTU van Ethernet bedraagt 1500 bytes (grotere IP-datagrammen moet gefragmenteerd worden), de minimum lengte van het gegevensveld bedraagt 46 bytes (als het IP-datagram kleiner is, moet het gegevensveld opgevuld worden; het aantal bytes dat eigenlijke data bevat, kan dan bepaald worden via het lengteveld van de header van het IP-datagram).

- **padding**
  In dit veld staan de eventuele bytes die gebruikt worden om ervoor te zorgen dat minimum frame lengte 64 bytes bedraagt. Deze padding bedraagt maximaal 46 bytes.
- **checksum**
  Het checksumveld (4 bytes) wordt gebruikt om de Cyclic Redundancy Check (CRC) code in op te slaan. Aan de hand van die code kan de adapter bepalen of er bitfouten in het ontvangen frame aanwezig zijn. De CRC-code wordt door de versturende adapter berekend aan de hand van de andere bits van het frame (zonder de preamble dus). De ontvangende adapter zal ook die code gaan berekenen, en zal zijn uitkomst vergelijken met de waarde in het checksumveld (CRC check). Indien de test negatief is, weet de host dat er een fout in het frame aanwezig is, en zal die het frame gewoon negeren.

5.2 ***Bespreek het CSMA/CD principe.***

We bekijken eerst de algemene eigenschappen van het CSMA/CD (Carrier Sense Multiple Access / Collision Detection) algoritme:

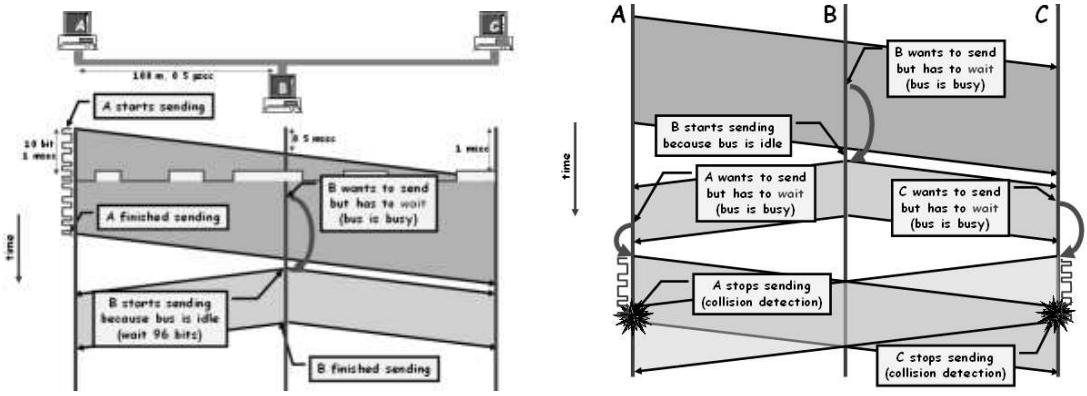
- een adapter kan op elk gewenst moment beginnen met verzenden (er worden geen slots gebruikt)
- een adapter verzendt nooit als een andere adapter bezig is (carrier sensing)
- een verzendende adapter stopt met verzenden zodra deze detecteert dat een andere node ook bezig is met verzenden (collision detectie)
- een adapter probeert pas opnieuw een frame te verzenden na een willekeurige tijd te hebben gewacht

Nu bekijken we hoe een specifieke adapter het CSMA/CD protocol zal uitvoeren:

1. De adapter krijgt een PDU van de netwerklaag van de node, maakt een frame aan, en plaatst het frame in een buffer op de adapter.
2. Als de adapter detecteert dat het kanaal niet in gebruik is (geen signaal), dan begint de adapter met de verzending van het frame, anders (kanaal wel in gebruik) wacht de adapter totdat die geen signaal meer ontvangt, en begint dan met het verzenden van het frame.
3. Tijdens het verzenden controleert de adapter of er een signaal aanwezig is. Als er geen signaal binnenkomt, en het volledige frame verzonden is, wordt het frame als verwerkt beschouwd.

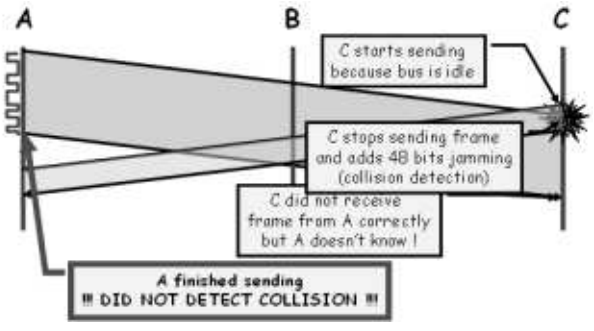
4. Als de adapter wel een signaal detecteert, stopt die met verzenden, en verzendt een jamsignaal van 48 bits (om zeker te zijn dat de andere aangesloten actieve stations ook de botsing detecteren).
5. Nadat dit signaal verzonden is, zal de adapter in een exponential backoff fase komen, waarbij de adapter voor de $n$de keer dat een collision gedetecteerd wordt, een willekeurige waarde K uit $0, 1, 2, \ldots, 2^m - 1$ kiezen ,waarbij m in het bereik $min(n, 10)$ ligt. De adapter zal dan $K \times 512$ bitintervallen wachten, om dan verder te gaan met stap 2.

Onderstaande figuren tonen hoe het CSMA/CD principe werkt.




### 5.3 *Waarom gebruikt men bij Ethernet een minimale framelengte van 64 bytes?*

Een minimale framelengte van 64 bytes is nodig om een succesvolle collision detectie te garanderen. Om zeker te zijn van een succesvolle collision detection moet de frameduur minstens 2 keer groter zijn dan de propagatietijd. Als we 64 bytes kiezen als minimale lengte, hebben we bij een 10 Mbit/sec Ethernet een maximum van $51,2\mu$sex roundtrip delay (5 $\mu$sec per km kabel, dus max. 5 km kabel).
Op onderstaande figuur zien we wat er zou gebeuren als aan deze vereiste niet voldaan wordt.




### 5.4 *Wat is exponential back-off (bespreek)?*

*Exponential back-off* wordt gebruikt bij het CSMA/CD protocol. Voor elke $n$de collision die de adapter detecteerd, kiest die een willekeurige

waarde $K$ uit $0, 1, 2, \ldots, 2^m - 1$ kiezen, waarbij $m$ in het bereik $min(n, 10)$ ligt. De adapter zal dan $K \times 512$ bitintervallen wachten, om dan verder te gaan met stap 2. De waarde 512 bekomt men door de minimale framelengte van 64 bytes te beschouwen. Die vereist een tijdslot van 512 ($64 \times 8$) bit tijden om zeker verzonden te zijn. Het aantal mogelijke tijdsloten wordt beperkt tot 1023 ($2^10 - 1$). De reeks waaruit de waarde voor $K$ gekozen wordt neemt exponentieel toe met het aantal collisions detections (tot de waarde 10 bereikt wordt). Dit gebeurt omdat, wanneer de collision gedetecteerd wordt, de adapter geen idee heeft hoeveel adapters bij de collision betrokken zijn. Als er maar een paar adapters betrokken zijn, is het voldoende om een klein aantal mogelijke tijdsloten aan te bieden, bij een groot aantal betrokken adapters moeten er meerdere tijdsloten zijn (anders blijft er collision optreden).

5.5 ***Bespreek ARP bij Ethernet.***

Het **Access Network Protocol (ARP)** wordt gebruikt om de netwerklaagadressen (IP-adressen) en de datalinklaagadressen (LAN-adressen) met elkaar te koppelen.
Een *LAN-adres* (fysiek adres, Ethernet-adres, MAC-adres) is meestal 6 bytes lang, en wordt voorgesteld in hexadecimale notatie (iedere byte als stel hexadecimale getallen). Het LAN-adres van een *adapter* is permanent en uniek. Een *LAN-broadcastadres* wordt gebruikt om een frame te versturen naar alle adapters in het LAN (Local Area Network), en is van de vorm FF-FF-FF-FF-FF-FF (48 1 bits). Drie van de vier bytes liggen vast voor de maker (bedrijf) van de interface, de laatste byte wordt door ingevuld door de maker zelf.
Als een node een datalinkframe wil versturen naar een andere node in hetzelfde LAN, dan heeft die node het MAC-adres van de ontvanger nodig om als bestemming op te geven in het datagram. De node zal daarvoor eerst gaan kijken in zijn eigen *ARP-tabel*. In de ARP-tabel zijn IP-adressen gekoppeld aan LAN-adressen. De entry's in de tabel hebben allemaal een TTL van typisch 20 minuten. Het is niet noodzakelijk dat die tabel een verwijzing bevat voor iedere node inhet LAN.
Als het LAN-adres van de node niet in de eigen ARP-tabel staat, maakt de node gebruik van het *ARP-protocol*. Er wordt een ARP-verzoekbericht verstuurd, met als bestemming het broadcastadres van het LAN. Dat ARP-pakket wordt in een datalinkframe verpakt, en verstuurd. Iedere node in het LAN zal het pakket ontvangen, dat via zijn adapter doorgeven aan de node. Dan zal elke node kijken of zijn eigen IP-adres overeenkomt met het IP-adres in het verzoekbericht. De node waarvan het IP-adres overeenkomt, zal een antwoordbericht versturen met de gevraagde verwijzing naar de verzoekende node (dit gebeurt niet via een broadcastframe, maar via een standaardframe, omdat de verzender het MAC-adres van de verzoekende node kent) . Die zal dan zijn ARP-tabel bijwerken, en het IP-datagram versturen.
We bekijken nu ook het geval waarbij een node een datagram wil verzenden buiten het LAN. Dat betekent dat het datagram een router zal passeren. Een router heeft meerdere interfaces, en zal voor iedere interface een aparte ARP-module en een eigen adapter hebben. De node verstuurt een ARP-verzoek om het MAC-adres te bekomen van de interface waar het data-

gram naartoe gestuurd moet worden. De router zal in zijn ARP-tabel een verwijzing staan hebben voor een IP-adres dat van dezelfde vorm is als het bestemmingadres. De router zal dus antwoord sturen, met daarin het MAC-adres van zijn interface die in hetzelfde LAN ligt als de verzoekende node. Die node zal het datagram naar die adapter versturen, en de router zal het datagram doorgeven aan zijn netwerklaag. De router zal via zijn padbepalingstabel bepalen via welk van zijn andere interfaces het pakket moet verstuurd worden. Deze interface zal het datagram in een nieuw frame verpakken, het MAC-adres bepalen van de bestemming in het bestemmings-LAN via ARP, en dan het datagram versturen naar de adapter die het MAC-adres heeft dat resulteert uit het ARP-verzoek. Die adapter zal het datagram doorgeven aan zijn node, en zo is het datagram op zijn bestemming aangekomen.

5.6 ***Wat is een Ethernet hub? En een Ethernet bridge?***

Een **hub** is een apparaat dat toelaat om verschillende nodes met elkaar te verbinden. Alle bits die een hub binnenkrijgt op een van zijn interfaces worden doorgestuurd naar iedere node die een verbinding heeft met een van de andere interfaces (point-to-point verbinding tussen node en hub). Hubs verwerken bits, en werken daarom op de fysieke laag. Ze zijn eigenlijk niet meer dan repeaters, met wat extra netwerkbeheerfunctionaliteit (afkoppelen van defecte adapters die constant frames verstuurd).
Men kan nodes op een hierarchishe manier met elkaar verbinden d.m.v. hubs, maar daarbij zijn er verschillende grote nadelen: de afzonderlijke collision domeinen worden samengevoegd tot één collision domein, waardoor de doorvoercapaciteit afneemt (verschillende LAN's apart hebben 10 Mb/s doorvoercapaciteit, eens verbonden met hub moeten ze 10 Mb/s delen met de andere LAN's die met de hub verbonden zijn). Een ander nadeel is dat het niet mogelijk is om voor verschillende LAN's verschillen Ethernet-technologieën te gebruiken, anders kunnen ze niet op dezelfde hub verbonden worden. Bovendien beperken de verschillende technologieën het maximale aantal nodes in een collision domein, de maximale afstand tussen twee hosts in een collision domein, en ook het maximale aantal lagen in een multi-tier structuur. Een voordeel van een dergelijke systeem is dat de maximale afstand tussen 2 nodes toeneemt (omda hub werkt als repeater).

Een **bridge** is een packetswitch op laag 2. Het ontvangt en verstuurd Ethernet-frames op basis van het LAN-adres in de frame header, en maakt bij het forwarden van pakketten gebruik van het CSMA/CD principe. Zijn transparant, omdat de hosts geen weet hebben dat een bridge aanwezig is (er wordt niet gewijzigd aan het doorgestuurde frame, het blijft het LAN-adres van de verzender behouden). Bovendien zijn de plug-and-play en zelflerend: de bridge-tabellen die gebruikt worden, worden automatisch ingevuld, en bridges moeten nie geconfigureerd worden.
De voordelen van bridges t.o.v. hubs is dat ze de verschillende collision domeinen van de LAN's die ermee verbonden zijn behouden, en dat de verschillende LAN's verschillende technologieën kunnen gebruiken. Pakketten kunnen gefilterd (als ze van het LAN komen waarvoor ze bestemd zijn), of geforward worden (als ze voor een ander LAN bestemd zijn). De interfaces van een bridge hebben geen LAN-adres, maar ze kunnen in

de LAN waarmee ze verbonden zijn, wel gezien worden als een adapter (omdat de bridge het CSMA/CD principe gebruikt).

5.7 ***Hoe worden de bridgetabellen ingevuld? En hoe worden ze gebruikt?***

Bridges zijn *zelflerend*. De bridgetabellen worden automatisch ingevuld. Wanneer een node een frame zal versturen dat de bridge bereikt, zal die het bronadres en de interface waarop het frame is binnengekomen opslaan in zijn bridgetabel, samen met het tijdstip waarop dit gebeurde (omdat de entry's in de tabel na een bepaalde tijd verwijderd worden als de node gedurende die tijd geen frame meer verstuurd heeft).
Wanneer de bridge een frame ontvangt, zal die aan de hand van het bestemmings-LAN-adres en zijn bridgetabel bepalen naar welke van zijn interfaces het frame moet verstuurd worden. Indien dit vermeld wordt in de tabel zal de het pakket naar één interface gestuurd worden, in het andere geval dat de bridge niet weet via welke interface het frame moet verstuurd worden, zal het naar elk van zijn interfaces (de uitgangsbuffer ervan) verstuurd worden (uitgezonderd de interface waarop het pakket toegekomen is). Een pakket dat, volgens de bridgetabel, bestemd is voor dezelfde interface als waarop het binnengekomen is, wordt gefilterd (niet verstuurd).

5.8 ***Geef een aantal voor- en nadelen van bridges (versus routers).***

- een bridge is een packetswitch op laag 2, een router is een packetswitch op laag 3
- een bridge is *plug-and-play*, een router moet geconfigureerd worden
- bridges kunnen relatief snel pakketten filteren en doorsturen, routers hebben een langere verwerkingstijd per pakket nodig, omdat ze op laag 3 werken (netwerklaag)
- een bridge gebruikt niet het optimale pad van de ene node naar de andere, omdat het spanning-treeprotocol gebruikt wordt (om te vermijden dat frames onnodig veel gekopieerd worden door de switch), een router zal het pakket wel versturen via het meest optimale pad (d.m.v. een routing algoritme)
- bridges bieden geen bescherming tegen op hol geslagen hosts, routers wel (ze laten ook geen broadcast frames door)
- omdat adressen binnen een netwerk hierarchisch toegekend worden, kunnen routers makkelijker het pad bepalen waarover een pakket moet verzonden worden

Bridges worden meestal gebruikt voor kleine netwerken (enkele honderden hosts), routers voor grotere netwerken (duizenden hosts).

5.9 ***Bespreek PPP.***

Het *Point-to-Point protocol (PPP)* is een protocol dat een point-to-point verbinding mogelijk maakt tussen twee nodes. De belangrijkste vereisten voor PPP zijn:

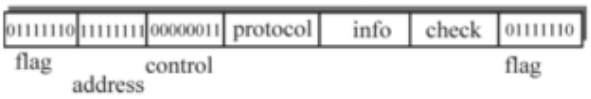
- *pakket framing*: het netwerkdatagram wordt ingekapseld in een datalinkframe, en een dergelijk datalinkframe is in staat om data van eender welk netwerkprotocol te bevatten, en het te demultiplexen naar de netwerklaag
- *bit transparantie*: er mag geen beperking zijn van bepaalde bitsequenties in het verstuurde netwerkdatagram (wordt opgelost door middel van byte stuffing)
- *foutdetectie*: fouten moeten gedetecteerd kunnen worden, maar niet gecorrigeerd
- *connection liveless*: gedecteerde link fouten moeten kunnen gemeld worden aan de netwerklaag
- *netwerklaag adres onderhandeling*: PPP moet een mechanisme hebben om de communicerende netwerklagen te configureren

Volgende zaken zijn dus *niet* vereist:

- *foutcorrectie*
- *flow controle*
- *sequencing* (herorderen van data)

Deze zaken moeten opgelost worden op hogere niveau's.

We bekijken de structuur van een PPP dataframe:




**vlag**

Het vlagveld vooraan en achteraan elk PPP-veld heeft steeds de waarde 01111110 (1 byte).

**adresveld**

Dit veld heeft steeds de waarde 11111111.

**controleveld**

Dit veld heeft steedsde waarde 00000011.

**protocol**

Dit veld is 1 of 2 bytes lang, en duidt het netwerkprotocol aan waarnaar PPP de inhoud van het gegevensveld moet sturen eens het datagram aangekomen is.

**informatie**

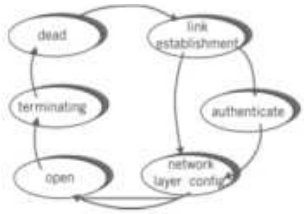
Dit veld bevat de verstuurde informatie. De MTU bedraagt 1500 bytes, maar dit kan anders ingesteld worden wanneer de link geconfigureerd wordt.

**checksum**

Dit veld maakt gebruikvan een twee-byte of vier-byte CRC-methode om bitfouten in een verzonden frame te kunnen detecteren.

Het adresveld en controleveld lijken nutteloos, maar werden ingevoerd om eventueel later andere waarden te krijgen op basis van bepaalde criteria. Dit is echter nog steeds niet gebeurt (deze velden moeten dus niet effectief verstuurd worden).

De data transparantie vereiste zorgt ervoor dat we een oplossing moeten zoeken voor bitsequenties van de vorm 011111110 in het informatieveld van het verstuurde frame. Dit wordt opgelost d.m.v. byte stuffing: na elke 011111110 bitsequentie wordt nog een 011111110 byte geplaatst. Op die manier zal de ontvanger, wanneer twee 011111110 bytes na elkaar voorkomen, 1 byte negeren (de stuffed byte), en verder gaan met het inlezen van de data. Als een enkele 011111110 byte ontvangen wordt, gaat het om een vlagveld van een PPP-frame.




Voordat elke link opgezet wordt, moet die geconfigureerd worden. Dit gebeurt aan de hand van het link-controlprotocol (LCP). De zaken die geconfigureerd moeten worden houden de maximale frame lengte in, authenticatie, ... Voor IP gebruikt een PPP-datagram het IPCP-protocol om IP-adressen te configureren of te achterhalen.

# Hoofdstuk 6

# Multimedianetwerken

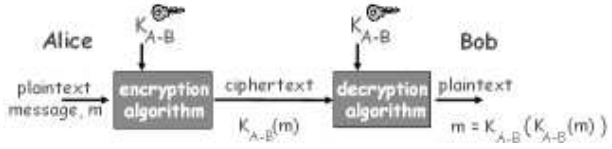
Geen vragen, want niet te kennen.

# Hoofdstuk 7

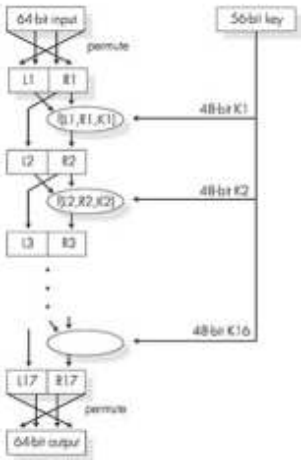
# Beveiliging

7.1 ***Bespreek het principe van cryptografie met symmetrische sleutels.***

Bij cryptografie met *symmetrische sleutels* gebruiken de zender en ontvanger dezelfde symmetrische sleutel $K_{A-B}$. Het bericht (plaintext) dat de zender wenst te versturen, wordt gecodeerd aan de hand van de symmetrische sleutel (die geheim gehouden wordt, enkel de zender en ontvanger kennen de sleutel). Het gecodeerde bericht (ciphertext) wordt naar de ontvanger verstuurd, die het zal decoderen met behulp van dezelfde symmetrische sleutel. Op die manier krijgt die het originele bericht dat verstuurd werd. In de onderstaande figuur staat het principe schematisch.



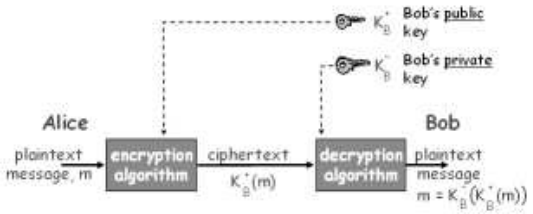

Een voorbeeld van cryptografie met openbare sleutels is het DES (Data Encryption Standard) algoritme. DES maakt gebruikt van een 64-bit plaintext input en een 56-bit symmetrische sleutel. Enkelvoudige DES werd reeds gekraakt in enkele uren, daarom heeft men, om het principe veiliger te maken, een uitbreiding gedaan: het gebruiken van 3 verschillende sleutels na elkaar (3-DES), en het gebruik van cipher block chaining (het geëncrypteerde blok $j$ wordt eerst d.m.v. een XOR 'samengevoegd' met blok $j + 1$, voordat dat laatste blok geïcrypteerd wordt).
We bekijken de werking van DES aan de hand van de figuur. Het 64-bit woord wordt eerst gepermuteerd. Daarna worden 16 keer dezelfde operatie uitgevoerd, telkens met een verschillende 48-bit sleutel die afgeleid wordt uit de 56-bit symmetrische sleutel. Na de laatste operatie met een 48-bit sleutel wordt het gecodeerde woord nog eens gepermuteerd. Om het gecodeerde woord opnieuw te decoderen, wordt het algoritme in omgekeerde volgorde uitgevoerd.




Het probleem bij cryptografie met symmetrische sleutels, is dat de zender en ontvanger een symmetrische sleutel moeten kunnen afspreken. Dit probleem kan opgelost worden door het gebruik van een KDC (Key Distribution Center), een betrouwbare organisatie die als tussenpersoon dienst doet.

7.2 ***Bespreek het principe van cryptografie met openbare sleutels.***

Bij cryptografie met *publieke sleutel* delen de zender en ontvanger geen symmetrische sleutel. Er wordt gebruik gemaakt van een publieke sleutel, die iedereen kent, en een geheime sleutel, die enkel de ontvanger kent. De zender codeert het bericht m.b.v. de publieke sleutel $K_B^+$, verstuurd het bericht naar de ontvanger, en die decodeert het bericht met zijn geheime sleutel $K_B^-$(die enkel hij kent).




Uit deze manier van werken volgen 2 vereisten voor de werking ervan:

- $K_B^+$ en $K_B^-$ moeten zo gekozen worden dat $K_B^-(K_B^+(m)) = m$
- Als de publieke sleutel $K_B^+$ gekend is, moet het onmogelijk zijn om de geheime sleutel $K_B^-$ te kunnen bepalen

Een probleem hierbij is hoe de ontvanger zeker kan weten dat die de publieke sleutel van de persoon gebruikt die hij/zij denkt te gebruiken. De oplossing daarvoor is het gebruik van certificate authority's (CA).

Een voorbeeld van cryptografie met openbare sleutels is RSA (Rivest Shamir Adelson). Bij RSA gebruikt de ontvanger volgend algoritme om de 2 sleutels de bepalen:

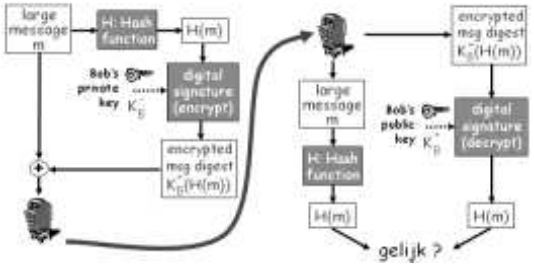
- Kies 2 grote priemgetallen $p$ en $q$ (elk ong. 1024 bits lang)
- Bereken $n = pq$ en $z = (p-1)(q-1)$
- Kies een getal $e$ dat geen gemeenschappelijke factoren heeft met $z$ ($e$ en $z$ zijn relatief priem)
- Kies een getal $d$ zodat $ed - 1$ exact deelbaar is door $z$ (m.a.w. $ed \ mod \ z = 1$)
- De publieke sleutel is de combinatie van $n$ en $e$, de geheime sleutel is $n$ en $d$

Een de publieke sleutel (n,e) en de geheime sleutel (n,d) bepaald zijn, kunnen we een bericht coderen/decoderen: Om een bericht te *coderen* berekenen we $c = m^e \ mod \ n$. Het gecodeerde bericht is dan $c$. Om te decoderen berekenen we $m = c^d \ mod \ n$. Op die manier bekomen we het originele bericht opnieuw.
Een belangrijke eigenschap van RSA is dat

$$K_B^-(K_B^+(m)) = K_B^+(K_B^-(m))$$

### 7.3 *Bespreek het principe van digitale handtekening.*

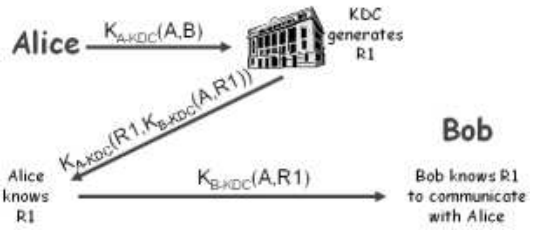
De bedoeling van een *digitale handtekening* is om het bericht dat een zender wil versturen te ondertekenen, zodat de ontvanger van het bericht zeker weet dat het bericht verstuurd werd door die zender. Dit kan gebeuren met behulp van de geheime sleutel van de zender. Als de ontvanger het bericht decodeerd met de openbare sleutel van de zender, weet deze zeker dat: het bericht verstuurd ondertekend door de ontvanger, en door niemand anders, en ook dat het ondertekende bericht niet gewijzigd werd (anders zou de decryptie 'vreemde' resultaten geven of niet lukken).
Een nadeel aan deze methode is dat het gehele bericht gecodeerd moet worden, wat lang duurt. Daarom zal men gebruik maken van *message digests*, die een vaste lengte hebben, en makkelijk te berekenen zijn. Daarbij wordt gebruik gemaakt van een hash functie, die een vaste lengte message $H(m)$ digest berekend voor een bepaald bericht $m$. Aan de hand van de message digest $x$, moet het bijna onmogelijk zijn een ander bericht $m'$ te vinden zodat $x = H(m')$ ook geldig is. Onderstaande figuur toont aan hoe de message digest gebruikt wordt bij het versturen van een ondertekend bericht.



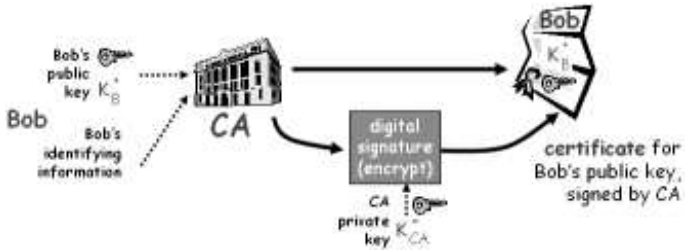

### 7.4 *Bespreek KDC en CA.*

**KDC**

Een KDC (Key Distribution Center) wordt gebruikt om het probleem op te lossen dat optreedt bij cryptografie met symmetrische sleutels. Een KDC is een betrouwbare organisatie die dienst doet als tussenpersoon tussen de zender en ontvanger die een bericht, gecodeerd via een symmetrische sleutel, willen uitwisselen.
Een KDC deelt met al zijn verschillende geregistreerde gebruiker een verschillende geheime sleutel. Als een zender een bericht wenst te versturen naar een ontvanger, moet deze eerst een symmetrische sleutel afspreken met de ontvanger. De zender doet een aanvraag bij het KDC, die stuurt gecodeerd een *sessiesleutel* naar de zender terug, samen met een gecodeerd bericht voor de ontvanger (waarin de identiteit van de zender, en dezelfde sessiesleutel zitten). De zender decodeerd het bericht, en weet zo de sessiesleutel en het gecodeerd bericht voor de ontvanger. Dat gecodeerd bericht verstuurt de zender naar de ontvanger, die, na decoderen met de geheime sleutel die gedeeld wordt met het KDC, de identiteit en de sessiesleutel kent. Daarmee kunnen de zender en ontvanger de sessiesleutel gebruiken om een bericht uit te wisselen.
Onderstaande figuur toont de werking, waarbij $K_{A-KDC}$ de geheime sleutel tussen de zender en het KDC is, $K_{B-KDC}$ de geheime sleutel tussen de ontvanger en het KDC, en $R1$ de sessiesleutel die afgesproken wordt.

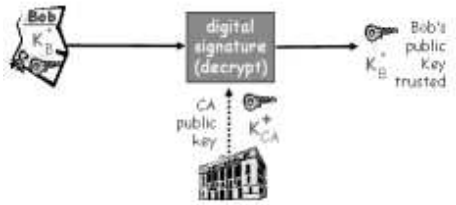



**CA**




Een CA (certificate authority) lost het probleem op bij cryptografie met openbare sleutels: het zeker weten dat een bepaalde publieke sleutel bij een bepaalde persoon hoort.
Een persoon of router $E$ registreert zijn publieke sleutel bij de CA. $E$ moet daarbij zijn identiteit bewijzen aan de CA, die dan een certificaat zal maken, waarbij E gebonden wordt een zijn publieke sleutel. Het certificaat bevat de publieke sleutel vanE, en is digitaal ondertekend door de CA. Bovenstaande figuur toont het proces.

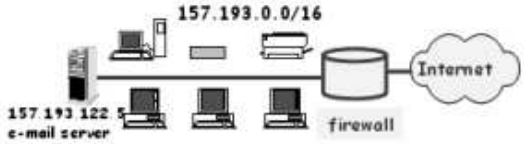
Wanneer een zender nu de publieke sleutel wil achterhalen van een bepaalde persoon haalt die het certificaat op van de ontvanger, past daarop de publieke sleutel van de CA op toe om de digitale handtekening te verifiëren. Op die manier is de zender zeker dat de verkregen publieke sleutel overeenkomt met de persoon waarvan men de publieke sleutel wil.




7.5 ***Bespreek pakketfiltering "packet firewall" en toepassingsgateway "application firewall".***

Een *firewall* zorgt ervoor dat het interne network van een organisatie geisoleerd wordt van de buitenwereld, waarbij sommige pakketten doorgelaten worden, en andere niet. De bedoeling van firewalls is om DoS (Denial of Service) aanvallen te vermijden, het verhinderen van illegale aanpassing en/of toegang van/tot gegevens, en enkel geauthoriseerde toegang tot het interne netwerk toe te laten. We onderscheiden de types:

**packet-filtering**

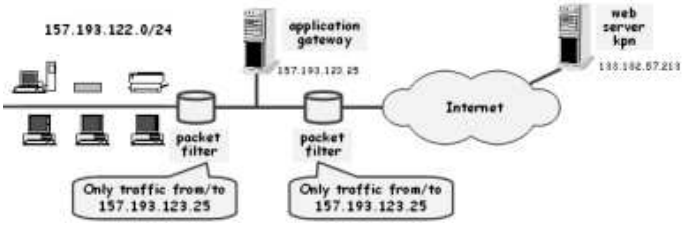



Een intern netwerk is meestal via een router verbonden naar het publieke internet. De router, die dienst doet als firewall, de pakketten filteren, en zal de beslissing om een pakket door te sturen of tegen te houden zal baseren op:

- bron IP-adres, bestemmings IP-adres
- TCP/UDP bron- en bestemmingspoortnummers
- ICMP berichttype
- TCP SYN en ACK bits

**applicatie-niveau**

Een *applicatie gateway* zal uitgaande en inkomende informatie filteren gebaseerd op applicatie-data. Deze gateways worden typisch gecombineerd met pakket filters, waarbij alle inkomende en uitkomende verkeer eerst moet passeren door een pakketfilter, en daarna ook door een toepassingsgateway. Een dergelijke applicatie gateway kan ook dienst doen als cache voor webverkeer, en als e-mail server (waarbij alle mails gescand worden op virussen).




Er zijn echter ook een aantal nadelen aan firewalls en gateways. Bij IP spoofing (het gebruiken van een vals IP-bronadres) kan een router niet weten of de data echt komt van de bron die gegeven wordt. Als meerdere applicaties een speciale behandeling nodig hebben, moeten ze elk hun applicatie gateway hebben. De client software moet weten hoe verbinding te maken met de gateway (instelen van het IP-adres van een proxy in de webbrowser).

7.6 ***Bespreek een aantal mogelijke aanvallen op het internet (en de bijhorende verdedigingen).***

**mapping**

Voor de eigenlijke aanval zal men onderzoeken welke diensten gebruikt worden op het netwerk, en ook hoe het netwerk opgebouwd is. Dit gebeurt door middel van het *ping* commando, om te bepalen welke IP-adressen de hosts in het netwerk hebben, en door poort-scanners te gebruiken, die proberen om een TCP connectie op te zetten naar iedere poort (en te kijken of er een antwoord op komt).
Mogelijke verdedigingen zijn het doorlichten van het verkeer dat binnenkomt in het netwerk, en het sporen naar verdachte activiteiten (IP-adressen en poorten die sequentieel gescand worden).

**packet sniffing**

Bij packet sniffing zal een host alle passerende pakketten lezen (in een lokaal netwerk), en kan op die manier alle niet gecodeerde gegevens lezen. Dit is vooral een groot probleem bij draadloze netwerken.
Mogelijke verdedigingen zijn het installeren van software die regelmatig controleert of een interface in promiscu mode werkt (dit is nodig om alle passerende pakketten binnen te halen), en het gebruik van een broadcast medium vermijden (en gebruik maken van switched Ethernet).

**IP spoofing**

IP spoofing is het gebruiken maken van een "vals" IP-bronadres in pakketten die verzonden worden. De ontvanger kan niet weten of een bronadres gespoofed is.
Een mogelijke verdediging hiertegen is het configureren van routers zodat pakketten van een ongeldig bronadres niet doorgestuurd worden.

**denial of service (DoS)**

Bij een Denial of Service aanval wordt een stroom van gegenereerde pakketten naar de ontvanger gestuurd, met de bedoeling om de ontvanger te overvloeden, zodat die zijn diensten niet meer kan leveren aan de gewone

gebruikers. Een variant hierop is een DDoS aanval, waarbij verschillende bronnen gebruikt worden, die door 1 iemand gecontroleerd worden (dikwijls via een virus).
Mogelijke verdedigingen hiertegen zijn het filteren van pakketten die overvloedig gestuurd worden, zoals bvb. SYN pakketten (nadeel: ook de gewone SYN pakketten zonder slechte bedoelingen worden weggefilterd), en het opsporen van de bron van de stroom van pakketten (dikwijls is dit een onschuldige client, die door een hacket gebruikt wordt).

**andere**

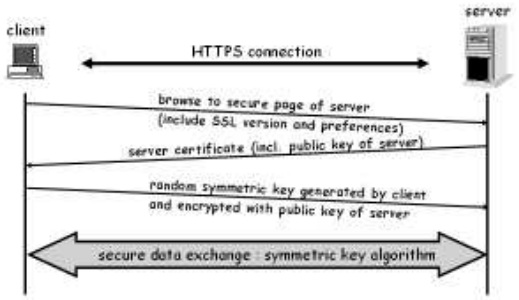
Overige mogelijke zaken die problemen kunnen opleveren zijn het achterhalen van paswoorden, het vervalsen van de identiteit, het installeren van eigen software op een server (trojan horse (controle overnemen), virus (soft- of hardware beschadigen)), virussen, ...
Veel gebruikte veiligheidsmaatregelen zijn een firewall(bescherming tegen buitenwereld), een centrale of lokale virus scanner, goede paswoord bescherming, het in de gaten houden van verkeer, het loggen en analyseren van gegevens, goede softwarekeuzes maken, het installeren van security patches, ...

### 7.7 *Bespreek het principe van PGP.*

*Pretty Good Privacy (PGP)* is een internet e-mailversleutelingsmethode die een de-facto standaard geworden is. Het maakt gebruik van symmetrische sleutel cryptografie, publieke sleutel cryptografie, een hash functie en een digitale handtekening. Het systeem voorziet geheimhouding, zender verificatie en integriteit. Het is mogelijk om met PGP enkel een bericht te ondertekenen, te coderen, of zowel te ondertekenen als te coderen (geheimhouding). Telkens als de geheime sleutel gebruikt wordt, moet het wachtwoord ingevoerd worden. Het PGP-bericht wordt telkens na de MIME-header weergegeven. De openbare sleutels van PGP worden door openbare PGP-servers gedistribueerd.

### 7.8 *Bespreek het principe van SSL.*

Secure Sockets Layer (SSL) biedt transportlaag beveiliging aan aan gelijk welke TCP-gebaseerde applicatie die van SSL gebruikt maakt, en wordt o.a. gebruikt voor webbrowsers voor e-commerce (https). Het biedt server authenticatie, data encryptie en client authenticatie (optioneel) aan. De client authenticatie kan gebeuren door client certificaten.
De figuur toont het opzetten van een https verbinding met een server.



De server authenticatie gebeurt door het aanvragen van de browser van een server certificaat aan een betrouwbare CA. De browser zal dan de publieke sleutel van die CA gebruiken om de publieke sleutel van de server te bekomen uit het certificaat.
Bij een geëncrypteerde SSL sessie genereert een symmetrische sessiesleutel, encrypteert die met de openbare sleutel van de server, en stuurt de gecodeerde sessiesleutel naar de server. Met zijn geheime sleutel, zal die de sessiesleutel decoderen, en op die manier worden de gegeven tussen de client en de server gecodeerd met behulp van de sessiesleutel.
SSL ligt eigenlijk tussen de applicatielaag en de transportlaag. Aan de verzendende kant ontvangt SSL gegevens van een toepassing, codeert de gegevens en verzendt de gecodeerde gegevens naar een TCP-socket. Aan de ontvangende kant leest SSL gegevens uit de TCP-socket, decodeert de gegevens en verzendt de gegevens naar de toepassing.

7.9 ***Bespreek het principe van IPSec (twee modes).***

Het *IP-security protocol (IPsec)* is een pakket protocollen dat beveiliging op de netwerklaag implementeert. Het biedt geheimhouding op de netwerklaag aan (zender encrypteert de data in het IP datagram), verificatie van bronnen (ontvanger kan IP-adres van bron controleren).
IPsec verzorgt een bron-bestemming handshake, die een logische verbinding (security agreement (SA)) vormt. Iedere SA is eenrichting, zodat er twee SA's nodig zijn voor bidirectionele communicatie. Een dergelijke SA wordt uniek gedefinieerd door het gebruikte protocol (AH of ESP), het IP-bronadres en een 32-bit verbindingsid. IPsec bestaat uit 2 basisprotocollen:

**Authentication Header (AH)**

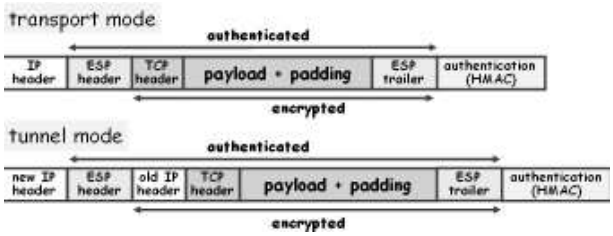
Het Authentication Header (AH) protocol biedt enkel bronidentificatie en gegevensintegriteit, geen geheimhouding. De beveiligde datagrammen die verstuurd worden, eens de logische verbinding opgezet is, bevatten de header van het AH protocol, die tussen de oorspronkelijke gegevens van het IP-datagram en de IP-header geplaatst is. De AH header vergroot het gegevensveld, en dit nieuwe gegevensveld is verpakt als een standaar IP-datagram. Het protocol veld in de IP header zal de waarde 51 hebben, om aan te duiden dat het AH protocol gebruikt wordt.
De header van het AH protocol bevat verschillende velden: een next-headerveld (zelfde rol als protocolveld in gewoon datagram, aanduiding naar welk transportlaagprotocol de gegevens moeten verstuurd worden), en SPI-veld (Security Parameter Index) (32-bits waarde, die gebruikt wordt voor unieke identificatie van de logische verbinding), een volgnummerveld (32-bits veld met volgnummer voor elk datagram, op deze manier kunnen playback- en man-in-the-middle-aanvallen voorkomen worden) en een Authentication Data veld (variabele lengte met ondertekende message digest (DES, MD5 of SHA) voor dit datagram, die berekend wordt over het originele IP-datagram; hierdoor worden bronverificatie en integriteit gewaarborgd).

**Encapsulation Security Payload (ESP)**

Dit protocol biedt bronhsotverificatie en geheimhouding op de netwerklaag. Een beveiligd datagram, dat verstuurd kan worden over de tot

stand gebrachte logische verbinding, wordt gemaakt door de oorspronkelijke gegevens in een IP-datagram te omhullen met header- en trailervelden, en dit opnieuw in een IP-datagram te verpakken. Voor het protocolveld in de IP-header wordt de waarde 50 gebruikt om aan te duiden dat het ESP protocol gebruikt wordt. De geheimhouding wordt gerealiseerd door het gebruik van DES-CBC-versleuteling. DeESP-header bestaat uit een 32-bits veld voor de SPI en een 32-bits veld voor het volgnummer. De trailer bevat het next-headerveld zoals in het AH protocol. Omdat de trailer samen met de oorspronkelijke gegevens gecodeerd wordt, kan een indringer niet bepalen welk transportprotocol gebruikt zal worden. Na de trailer volgt een authentication-dataveld, met hetzelfde doel als bij AH.

Het IPsec protocol bevat 2 modes: tunnel mode (tussen verschillende routers, firewalls) en transport mode (tussen end points(terminals en servers). Tunnel mode wordt gebruikt om bvb. tussen verschilende locaties van eenzelfde bedrijf velig te houden d.m.v. IPsec. Het IP-datagram groeit aan telkens een router of firewall gepasseerd wordt (zie figuur).




De IPsec-verificatie maakt gebruik van HMAC (Hashed Message Authentication Code), om en gecodeerde message digest te bekomen. Ipsec maakt gebruik van een standaard sleutelbeheerprotocol (Internet Key Exchange (IKE)) en een Internet Security Association and Key Managment protocol (ISKMP) waarin de procedures voor het tot stand brengen en verbreken van logische verbindingen gedefinieerd worden.

# Hoofdstuk 8

# Netwerkbeheer

8.1 ***Bespreek FCAPS en geef voorbeelden.***

*FCAPS (fault,configuration,accounting,performance,security)* is een netwerkbeheermodel dat 5 netwerkbeheergebieden definieert:

**foutenbeheer**

Het loggen, opsporen en reageren van/op foutcondities in het netwerk. Voorbeelden zijn fouten in een link, stroomonderbreking, ...

**configuratiebeheer**

Het bepalen welke apparaten in het netwerk aanwezig zijn, wat de hardware- en software-configuratie instellingen zijn, .... Voorbeelden zijn het achterhalen van routers, hubs, switches, routertabellen, ...

**accountbeheer**

Het specifiëren, loggen en controleren van toegangsrechten voor gebruikers voor netwerkbronnen. Voorbeelden: gebruikersquota, dataverkeerafhankelijke facturering, ...

**performantiebeheer**

Het kwantificeren, meten, rapporteren, analyseren en regelen van de performantie van verschillende netwerkcomponenten. Voorbeelden van dergelijke componenten zijn routers, hosts, links, maar ook end-to-end abstracties zoals een pad door het netwerk. Het verschil met foutenbeheer is performantiebeheer gericht is op langetermijnaspecten.

**beveiligingsbeheer**

Het regelen van de toegang tot netwerkbronnen, waarbij key distribution centers en certificate authorities bij betrokken zijn. Een andere vitale component zijn firewalls.

8.2 ***Wat is een MIB? Leg uit.***

De *Management Information Base (MIB)* is een soort virtueel informatiepakhuis waarin de beheerde objecten in een netwerk zijn opgeslagen. De waarden van de verschillende objecten samen vormen een getrouwe weergave van de huidige staat van het netwerk. De waarden kunnen door een beheersentiteit worden opgeslaan (met behulp van OBJECT-TYPE SMI-construct) door SNMP-berichten naar de agent te versturen die op de

beheerde node worden uitgevoerd namens de beheersentiteit. De objecten die logisch samen horen worden gegroepeerd in modules met behulp van de MODULE-IDENTITY SMI-construct. Om alle mogelijke objecten een naam te kunnen geven, wordt gebruik gemaakt van een hierarchische ISO Object Identifier Tree, waarbij iedere knoop een naam en een nummer heeft.

8.3 ***Bespreek het SNMP protocol (wat zijn de mogelijkheden).***

Het *Simple Network Managment Protocol (SNMP)* wordt gebruikt om MIB-informatie uit te wisselen tussen beheersentiteiten en agents die namens de beheersentiteiten worden uitgevoerd. Er zijn twee mogelijkheden om MIB info te bekomen:

**verzoek-antwoordmodus**


Het SNMP protocol wordt meestal in deze modus gebruikt. De beheersentiteit verstuurt een verzoek naar een SNMP-agent. Die ontvangt het verzoek, voert een bepaalde handeling uit, en verzendt het anwoord op het verzoek. Meestal is het een verzoek om de waarden in een MIB-object voor een beheerd apparaat terug te zenden of te wijzigen.

**trapmodus**


In deze modus stuurt een agent om een onderschepbericht naar een beheersentiteit. Die worden gebruikt om een beheersentiteit op de hoogte te brengen van een ongebruikelijke situatie waardoor waarden in een MIB-object zijn gewijzigd. Dit is bijvoorbeeld wenselijk wanneer een interface defect raakt, wanneer de belasting van een link een drempelwaarde dreigt te overschrijven, ...

SNMP definieert verschillende soorten Protocol Data Units (PDU's) die voor verschillende zaken gebruikt worden:

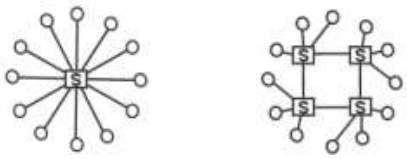
- *GetRequest, GetNextRequest en GetBulkRequest*: wordt gebruikt door beheerder om data op te halen van de agent (instantie, volgende in de lijst, blok)
- *InformRequest*: wordt gebruikt door een beheerder om MIB-waarden naar een andere beheerder te sturen
- *SetRequest*: wordt gebruikt door een beheerder om bij een andere beheerder MIB-waarden in te stellen
- *Response*: wordt gebruikt door een agent om antwoord te geven op een request van een beheerder
- *Trap*: wordt gebruikt door een agent om een beheerder in te lichten over een ongebruikelijke situatie

# Hoofdstuk 9

# Telefonie

9.1 ***Leg uit hoe (en waarom) men aan de typische topologie (ster-maas) komt van een telefoonnetwerk.***

Een ster topologie komt overeen met het verbinden van de klanten van een telefonie-systeem met een centraal punt. Er zal dan slechts 1 kabel zijn naar elke gebruiker, en de flexibiliteit om verbindingen op te zetten zal voorzien worden in het centrum van de ster (de exchange of central office). De gebruiker levert de informatie om de verbinding te maken met een andere gebruiker. De exchanges leveren de switching functionaliteit.
Een ster topologie is beter dan alle gebrukers onderling met elkaar verbinden, wat onmogelijk zou zijn, gezien het enorme aantal kabels, en de grote overhead dat dat zou voortbrengen (veel van de kabels zouden nooit gebruikt worden.
Het is duidelijk dat een enkele ster niet voldoende is om honderduizeden gebruikers met elkaar te verbinden. Daarom zal het netwerk in kleinere stukken gesplitst worden, die elk een eigen ster hebben. Opdat de gebruikers zouden kunnen verbinden met de verschillde sterren, moeten de sterren onderling met elkaar verbonden worden.
In onderstaande figuren staan een systeem met een enkele exchange, en een systeem met 4 exchanges.

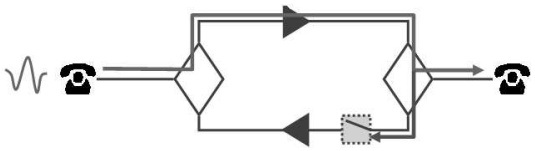



9.2 ***Hoe kan men over twee draden een bidirectioneel telefoongesprek voeren?***

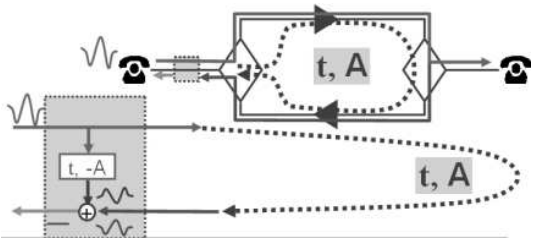
Een telefoonsysteem met 2 draden kan gebruikt worden voor verbindingen tussen lokale exchanges die dicht bij elkaar liggen. Op langere afstand zal men echter binnen het systeem moeten gebruik maken van 4 draden, waardoor amplificatie kan gebeuren op een eenvoudige manier. Langs beide kanten (inkomend en uitgaand) zal er zich dan een terminaal station verbinden dat het binnenkomend of uitgaand signaal versterkt, en dat

het signaal op twee binnenkomende draden omvormt ot een signaal op 1 binnenkomende draad en het signaal op een uitgaande draad omzet naar een signaal op twee uitgaande draden.
Er zijn nu 2 manieren om een bidirectioneel gesprek te voeren op een dergelijk systeem, waarbij echo's vermeden worden:

- **Half duplex operatie**: in dit geval zal men het stemsignaal enkel in één richting tegelijk doorsturen (de andere kant wordt geblokkeerd). Op deze manier wordt natuurlijk ook het echo signaal geblokkeerd. Omdat men in beide richtingen wil kunnen communiceren, zal er een stemactiviteitscircuit gebruikt worden, dat detecteert wie aan het spreken is, en dat de juiste transmissierichting zal activeren. Deze verbinding is kan men vergelijken met het walkie-talkie systeem: slechts één persoon kan 'tegelijk' praten.

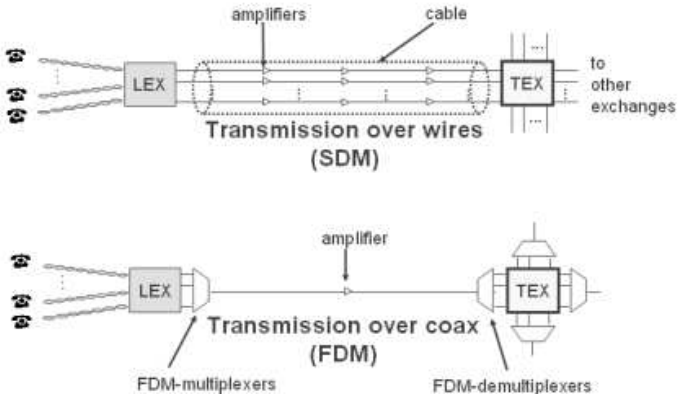


- **Echo cancellation**: in dit geval zal men een echo verwijderingsblok gebruiken om de echo te onderdrukken. Dit gebeurt door het "opnemen" van het originele signaal, en het op een later tijdstip (T) opnieuw af te spelen met de gepaste verzwakking (A). Door het inverteren van dit signaal, en het af te trekken van de ontvangen echo, zal men de echo verwijderen.

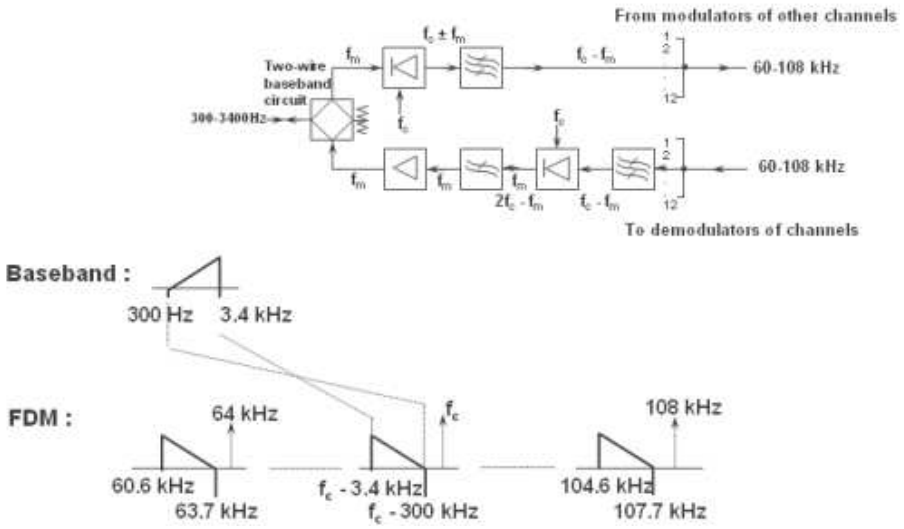



9.3 ***Leg de werking uit van FDM uit.***

We vermelden eerst een andere manier van multiplexen: Space Division Multiplexing (SDM) waarbij de verschillende kabels die nodig zijn om de circuits te verbinden met de exchange, elk gebruik maken van een eigen kabel, en dus ook van hun eigen amplifiers, en waarbij alle kabels samengevoegd worden in één grote kabel.
Bij *Frequency Division Muliplexing (FDM)* zal men gebruik maken van coaxiale kabel, om op die manier minder gebruik te maken van amplifiers dan bij SDM, omdat, door de lagere verzwakking van het gebruikte transmissiemedium (coaxiale kabel), minder amplifiers voldoende zijn over dezelfde afstand. Bovendien zullen alle telefoonsignalen verstuurd worden over een enkele coaxiale kabel.

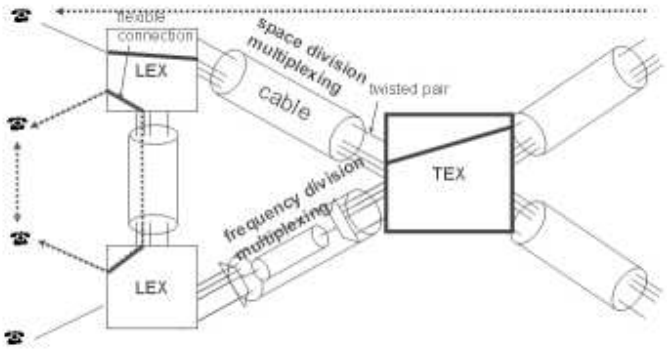



Een telefoonkanaal gebruikt enkel een frequentieband tussen 300 en 3400 Hz (baseband signaal). We kunnen dergelijke telefoonsystemen samen multiplexen tot een nieuw signaal met een hogere bandbreedte, die een multichannel carrier systeem genoemd wordt. We kunnen dit bereiken door ieder kanaal te moduleren op een verschillende carrier frequentie.
We geven een voorbeeld met het multiplexen van 12 kanalen. Aan de



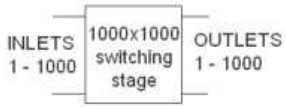

verzenderkant, wordt op ieder inkomend signaal ($f_m$) een gebalanceerde modulator toegepast, die de gepaste carrier ($f_c$) gebruikt. De output van die modulator is een dubbelzijdige band onderdrukt-carrier signaal ($f_c \pm f_m$). Dat signaal wordt doorgegeven aan een bandpass filter die de bovenste band ($f_c + f_m$) onderdrukt, en de onderste band ($f_c - f_m$) verstuurt (daardoor maximaliseren we het aantal kanalen dat we kunnen versturen in de beschikbare bandbreedte). De verschillende output's van die signalen (12 in totaal) worden gecombineerd om een FDM signaal te bekomen waarbij ieder telefoonkanaal vertaald wordt naar een ander stuk van het frequentie spectrum. We merken op dat de plaats tussen twee verschillende carriers 4 kHz is, om overlapping van kanalen te vermijden.

9.4 ***Waarom praat men over "circuit switching" in telefonie?***

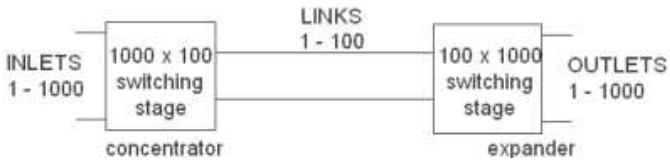



De verschillende switches (LEX (local exchange) en TEX (transit exchange)) spelen een centrale rol in de werking van het netwerk, ze leveren de flexibele interconnectiviteit tussen de verschillende telefoontoestellen, die enkel geleverd wordt voor de duur van de verbinding. Daarna kunnen ze gebruikt worden om andere oproepen te ondersteunen. Verbindingen worden niet onderbroken, en eventuele nieuwe oproepen zullen geen verbinding krijgen als een geen middelen beschikbaar zijn in het netwerk. Het is duidelijk dat een telefoonnetwerk vaste bandbreedte (3,4kHz) verbindingen (circuits) aanbiedt, waardoor ze circuit switched netwerken genoemd worden. Daardoor zijn de end-to-end vertragingen ook heel laag, wat belangrijk is voor telefonie.

9.5 ***Waarom gebruikt men expansie, concentratie en distributie in een telefoonschakelaar?***

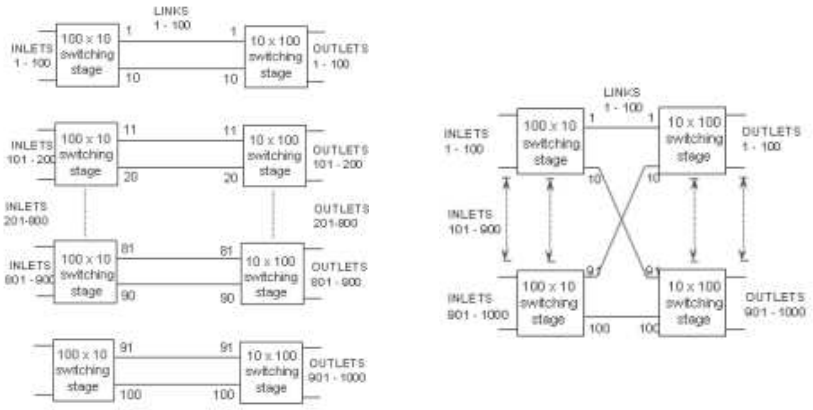



De verschillende cross-points (basis switching elementen) kunnen onderling verbonden worden door een simpele matrix, maar dit vereist 1 miljoen basis switching elementen, wat onpraktisch en zeer duur is. Het is bij deze manier van werken mogelijk om iedere ingang met de nodige uitgang te verbinden (fully non-locking), hoeveel verbindingen er ook zijn op dat moment. Maar als we een maximum van 1000 gelijktijdige verbindingen op een bepaald tijdstip beschouwen, blijkt dit zzer inefficiënt.

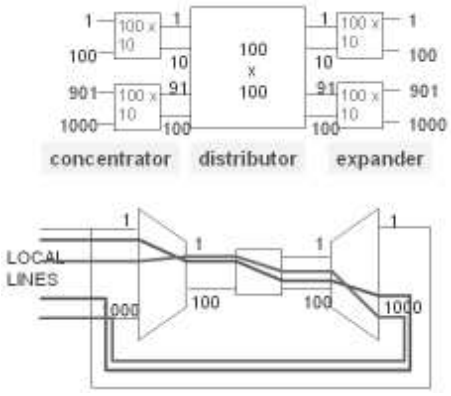



Een eerste stap om een betere efficiëntie te bereiken is het gebruik van 2 matrices, die elk 1000x100 contacten hebben. Daardoor reduceren we het

aantal cross-points tot 200.000, maar ook het aantal mogelijke verbindingen van 1000 tot 100. Omdat het onwaarschijnlijk is dat meer dan 100 gebruikers een verbinding maken op hetzelfde moment, is dit een aanvaardbare oplossing. De eerste matrix die gebruikt wordt, noemt men de concentrator (concentratie van 1000 naar 100 contacten), de tweede wordt expander (expansie van 100 naar 1000 lijnen) genoemd.



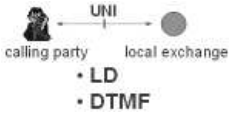

Om het aantal switches nog te reduceren, kan men gebruik maken van grouping. We kunnen daarbij het aantal cross-points reduceren tot 20.000, maar zelfs als we de groepen onderling verbinden, bestaat er nog steeds een belangrijke beperking in dit systeem: twee of meer lijnen van een groep kunnen niet verbonden zijn met dezelfde output groep. In de eerste figuur is er nog een groter probleem, waardoor deze manier van werken onaanvaardbaar is: de ingangen van de eerste groep kunnen niet verbonden worden met andere groepen.




Door het aantal switches licht te verhogen tot 30.000 en een extra component, een distributor, te gebruiken, kunnen we het systeem verbeteren. De distributor zorgt ervoor dat de 100 uitgaande lijnen van de verschillende groepen verbonden kunnen worden met de 100 inkomende lijnen. De beperking is nu zo dat slechts 10 lijnen van iedere groep onderling met elkaar kunnen verbonden worden. De reductie in complexiteit (van

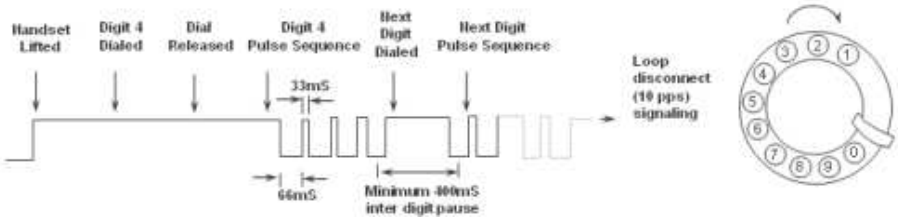
200.000 naar 30.000 switches) is vooral belangrijk voor grote switching netwerken.
In deze figuur zijn de uitgang lijnen van de expander verbonden met de ingang lijnen van de concentrator, omdat we een bidirectioneel systeem willen bekomen.

9.6 ***Bespreek LD en DMTF.***



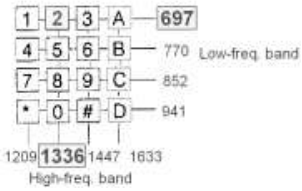

*LD* en *DMTF* zijn 2 systemen die toelaten om de LEX te laten weten welke verbinding men wenst te maken. Deze systemen worden gebruikt op de User Network Interface (UNI), die de gebruikersterminal verbindt met de LEX switch controller.

**Loop Disconnect (LD)**




Bij Loop Disconnect wordt een lus voltooid wanneer de hoorn van de telefoon wordt opgenomen. Diegene die een oproep wil maken, kan dan het nummer vormen m.b.v. een draaischijf (meestal), die een aantal pulsen genereert, door de weerstand van de batterij los te koppelen. Deze manier van werken is zeer traag, omdat het typisch 6 tot 15 seconden duurt om een nummer te vormen.

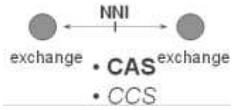
**Dual Tone Multi Frequency (DMTF)**




Bij Dual Tone Multi Frequency maakt men gebruik van drukknop telefoon. Wanneer een nummer ingedrukt wordt, worden twee frequenties gegenereerd die naar de LEX gestuurd worden, en daar gedecodeerd worden. Men maakt gebruik van 2 frequenties om het systeem betrouwbaarder te maken (geen interferentie met andere signalen). Dit systeem is ook veel sneller dan LD, en biedt de mogelijkheid voor extra knoppen (* en #) die gebruikt kunnen worden voor extra functionaliteit (bvb. phone banking).

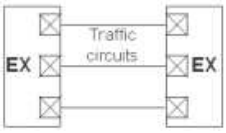
Hoewel DMTF het meest gebruikt wordt, bestaan er nog steeds lokale exchanges die een LD signaal vereisen, waardoor in moderne telefoons nog steeds de mogelijkheid bestaat om pulsen te gebruiken.

9.7 ***Bespreek CAS en CCS.***




*CAS* en *CSS* zijn classificaties van NNI (inter exchange, tussen de swtich controllers) signaling systemen, die gebaseerd zin op de manier dat signaling informatie gelinkd wordt aan het corresponderende stemkanaal. In analoge telefonie gebruikt men altijd CAS (buiten voor sommige lange afstandsverbindingen).

**CAS**

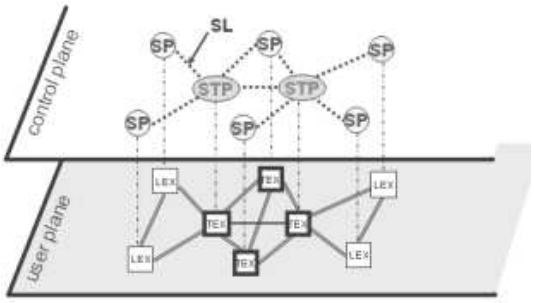



Bij Channel Associated Signaling (CAS) is er een signaling kanaal geassocieerd met ieder stemkanaal, waardoor een signaling voorzieningen moeten zijn op ieder circuit.

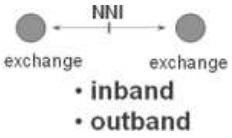
**CSS**


Bij Common Channel Signaling (CSS), dat gebruikt wordt op de nieuwere systemen, er er slechts één signaling kanaal dat gebruikt wordt voor de verschillende stemkanalen. In digitale systemen gebruikt men zowel CSS als CAS.

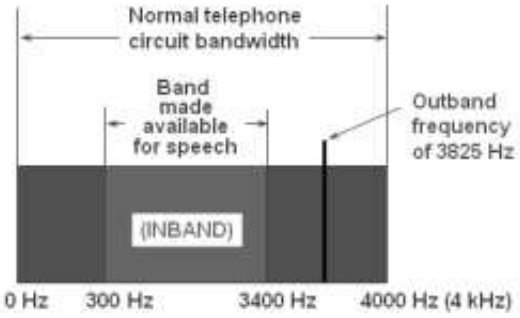
Dit systeem wordt gebruikt in het Integrated Digital Network (IDN), dat een CSS netwerk (SS7, Signaling Systems nr. 7) heeft. Dit netwerk (het controlenetwerk) bestaat uit *Signaling Points* (SP's, waar signaling boodschappen kunnen onstaan of eindigen), *Signaling Transfer Points* (STP's, waar signaling booschappen kunnen gerouteerd worden) en *Signaling Links* (SL, waarover de signaling boodschappen verstuurd worden). Het netwerk is bericht (pakket) gebaseerd, en het systeem van de signaling links komt niet overeen met de fysische links. Normaal gezien zal iedere switch een SP hebben, maar er zullen minder STP's zijn. Een STP hoeft niet overeen te komen met een switching node, het kan ook een intelligente server zijn in een Intelligent Network (IN). Er is ook een CCS signaling netwerk voor analoge telefonie (SS6), waar de digitale boodschappen verstuurd worden via modems.




9.8 ***Bespreek inband en outband signalering.***




Het onderscheid tussen inband en outband systemen is een tweede classificatie van FDM systemen (na het verschil tussen CAS en CSS).

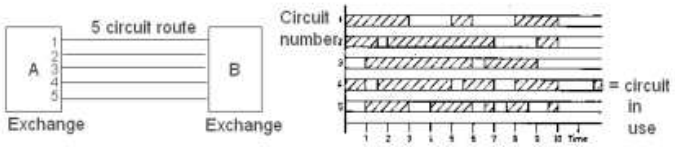



Voor *inband signaling* zal men dezelfde frequentie gebruiken als de normale (stem)frequentie (tussen 300 en 3400 Hz). Voor *outband signaling* gebruikt men een frequentie (of twee frequenties) die buiten die band liggen (maar nog altijd binnen de 4 kHz band die gebruikt wordt bij FDM).
In analoge telefonie wordt inband signaling gebruikt voor *inter register signaling*, dat de informatie draagt zoals "gekozen nummer", tussen exchange registers. Inter-register signaling maakt gebruik van een enkel of twee frequentie systemen.
Outband signaling wordt gebruikt voor *line signaling*, dat de lijn en de gebruikelijke voorzieningen controleert. Dit onderdeel maakt gebruik van een multifrequency code (MFC) om de informatie te versturen. Er worden altijd 2 frequenties verstuurd om het systeem betrouwbaarder te maken. Forward transmission gebruikt 6 frequenties (1380, 1500,1620,1740 en 1860), net als backward transmission (540, 660, 780, 900, 1020 en 1140). Door twee tonen te kiezen uit 6 beschikbare tonen kunnen we code nummers gebruiken tussen 1 en 15 ($6 \times 5 = 30$ keuzes, $30/2 = 15$ mogelijkheden, want altijd 2 frequenties).

Voor inband signalering zal het normale stemcircuit onderbroken worden, wat geen probleem vormt, want er gebeurt geen signalering tijdens het gesprek.

9.9 ***Leg uit: Erlang en GoS .***

Met **Erlang** bedoelt men de verkeersintensiteit, die gelijk is aan het gemiddelde aantal oproepen die tegelijk in behandeling zijn gedurende een bepaalde periode. Die wordt gemten in 'erlangs'. We bekomen de verkeersintensiteit door de totale bezettijd van alle circuits in de route op te tellen, en die som te delen door de tijdsperiode waarin de meting gedaan werd.
In onderstaand voorbeeld is de berekende waarde 3,5 erlang $((6 + 7,5 + 7,5+8+6)/10 = 3,5)$, wat betekent dat er gemiddeld 3,5 circuits gebruikt werden tijdens de beschouwde periode.



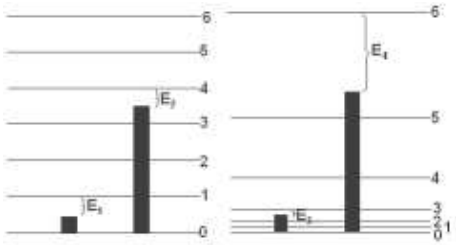

Men zal een wiskundig model gebruiken om een verkeersmodel op te stellen, zodat verkeersproblemen opgelost kunnen worden. Daarbij maakt men gebruik van *pure-chance trafiek* (Poissonistische trafiek), waarbij oproepaanvragen en oproepeinde's onafhankelijke gebeurtenissen zijn (dit is zo als een groot aantal oproepen beschouwd wordt) en een *statistisch equilibrium*, waarbij met er van uit gaat dat de statistieken niet gewijzigd worden in de beschouwde periode. Daarmee komen we tot formules voor het aantal oproepen in een gegeven periode (Poisson distributie), de intervallen tussen oproepaanvragen (negatieve exponentiële distributie) en oproepduur (negatieve exponentiële distributie).
Men zal ook formules gebruiken voor de kans dat $n$ circuits gebruikt worden wanneer de aangeboden trafiek $A$ Erlang bedraagt (waarbij men er van uitgaat dat er een oneindig aantal circuits beschikbaar is, zodat er geen oproepen verloren gaan), en formule om de **servicegraad (grade of service (GoS))** te bepalen in een lost call cleared systeem (dit is een systeem waarbij er $N$ uitgaande circuits zijn, en $A$ Erlang trafiek, en waarbij de volgende veronderstellingen gedaan worden: pure-chance trafiek, statistisch equilibrium, volledige beschikbaarheid (iedere oproep die toekomt kan verbonden worden met een vrij circuit) en oproepen die congestie (ze komen toe wanneer alle circuits bezet zijn) ondervinden, gaan verloren.
We merken nog op dat, indien met de GoS wil verhogen, men dan meer circuits zal moeten voorzien om dezelfde trafiek te kunnen verwerken. Voor $GoS = 0,01$ kunnen we volgende benaderende formules gebruiken, waarbij $N$ het aantal circuits voorstelt, en $A$ de trafiek intensiteit:

$$N = 6 + \frac{A}{0,85} \quad voor\ A < 75$$

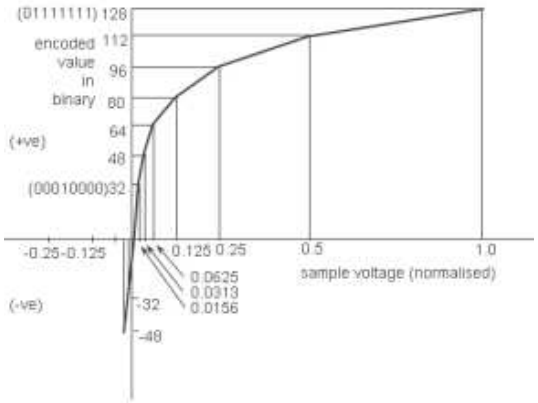
$$N = 14 + \frac{A}{0,97} \quad voor\ 75 < A < 400$$

$$N = 29 + A \quad voor\ 400 < A$$

9.10 ***Waarom gebruikt men geen lineaire A/D conversie bij digitale telefonie?***

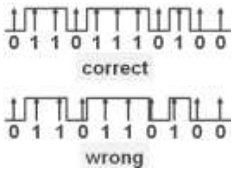



Moest men een lineaire A/D conversie gebruiken bij telefonie, dan zou de storing erger zijn voor kleine signalen dan voor sterke signalen (erger voor mensen die stil spreken, dan voor mensen die luid spreken). Die storing wordt door de ontvanger gehoord als ruis, en dus zal de graad van ruis erger zijn als het input signaal gereduceerd wordt. Dit probleem wordt geminimaliseerd door de quantisatie niveuas logaritmisch de spreiden, zodat grote signalen een grote fout toelaten, en kleine signalen enkel een kleine fout kunnen hebben.
Deze logaritmische quantisatie resulteert bovendien in een meer efficiïte quantisatie. We bekijken dit aan de hand van een voorbeeld. Stel dat we een dynamisch gebied nodig hebben tussen de -35 dBm en +3 dBm, met een signaal/quantisatiestoornis van 40 dB. Voor lineaire quantisatie zouden we meer dan 8000 (13 bit) niveau's nodig hebben, voor logaritmische quantisatie zijn er slechts 256 (8 bit) niveau's nodig. Dit is belangrijk om de nodige bandbreedte voor het versturen van informatie te verlagen (64 kB/s i.p.v. 104 kB/s). Er worden 2 verschillende companding (compression/expansion) wetten gebruikt: de A-wet en de Mu-wet. Bij de A-wet worden er 13 rechtlijnige segmenten gebruikt om een logaritmische curve voor te stellen, bij de Mu-wet wordne 15 segmenten gebruikt.
Op onderstaande figuur (A-law) zien we dat de bovenste helft van het domein $(0,5 \rightarrow 1,0)$ slechts 16 quantisatie-niveau's gebruikt, terwijl de laagste 1,6% van het positieve gebied als 32 niveau's gebruikt.

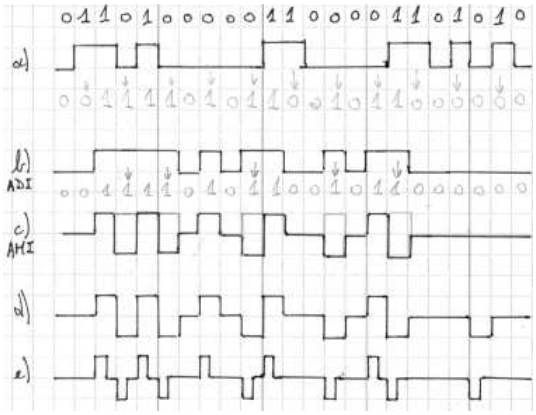


9.11 ***Waarom gebruikt men lijncodering?***

*Lijncodering* wordt gebruikt om het digitale signaal, dat bekomen werd door A/D conversie, voor te bereiden om het te versturen. Daarbij moeten enkele problemen problemen opgelost worden: voorkomen dat een DC level voorkomt in het signaal en de bepaling van de klok uit het signaal aan de ontvanger kant (dit is belangrijk omdat de verzender en ontvanger niet gesynchroniseerd zijn, zodat er bitfouten zullen optreden omdat een van beide niet op het gepaste moment zal samplen, zie onderstaande figuur). Een andere vereiste is het zo klein mogelijk houden van de nodige bandbreedte.




Enkele voorbeelden van codeerschema'szijn: Nonreturn to Zero-Level (NRZ-L), NonReturn to Zero Inverted (NRZI), Bipolar-AMI, Preudoternary, (Differential) Manchester, B8ZS, High Density Bipolar 3 (HDB3).
We bekijken nu HDB3 meer in detail: het is een dezelfde als Bipolar AMI, buiten het feit dat gelijk welke string van 4 nullen vervangen wordt door een code violation. We bekijken de stappen om een binaire sequentie om te zetten in een HDB3 signaal aan de hand van een voorbeeld. De eerste




stap is om het binaire signaal om te zetten in een uni-polair signaal (0 = laag, 1 = hoog). Dan doen we stapsgewijs enkele omzettingen die het uiteindelijke signaal zullen produceren.
Het gebruikte voorbeeld: 01101000001000011010 10

**Alternate Digit Inversion (ADI)**

Ieder andere bit (dus bits 1,3,5,...) wordt geinverteerd (enkel de grafiek), zodat lange strings van dezelfde waarde onderbroken worden. Er kunnen echter ook nieuwe dergelijke strings ontstaan.

**Alternate Mark Inversion (AMI)**

In deze stap wordt de polariteit van iedere andere 1-bit (mark) geinverteerd. Op die manier bekomen we een bipolair signaal met een gemiddeld DC niveau van 0 volt. Er kan nog altijd een string van nullen voorkomen.

**Zero string substitution**

Nu vervangen we elke string van 4 nullen door een string van 3 nullen, en een pulse met polariteit die dezelfde is als de laatst gebruikte polariteit (op die manier wordt de ontvangst van de data niet corrupt). Zo bekomen we een output signaal dat niet meer dan 3 opeenvolgende nul-bits bevat.

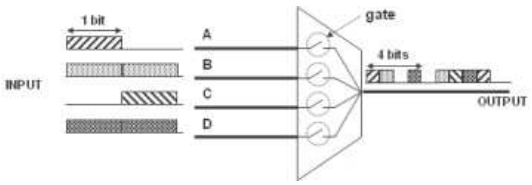
**Return to zero coding**

Na de eerste 3 stappen bekomen we een Non Return To Zero (NZR) signaal, omdat iedere mark (= 1-bit) in positieve of negatieve potentiaal blijft voor de volledige duur van het pulse interval (en keert dus niet terug naar nul). Door de pulse duur te halveren bekomen we een Return To Zero (RZ) signaal. Hierdoor wordt de energie inhoud van het signaal gereduceerd, waardoor we een cross-talk reductie en een reductie in inter-symbool interferentie (veroorzaakt door spreiden van de pulse) reduceren.

9.12 ***Leg de werking van TDM uit. Wat is het verschil tussen byte en bit interleaved TDM?***

*Time Division Multiplexing (TDM)* is gebaseerd op digitale signalen (64 kb/s stem) die gecompresseerd worden in de tijd, en samengevoegd worden tot een hoger bitrate signaal (vb. 30 digitale telefoonsignalen worden gemultiplexed in een enkele 2 Mbit/s signaal). Deze multiplexeringstechniek wordt gebruikt in digitale netwerken. Bijvoorbeeld, 4 inkomende (tributary) signalen gemultiplexed in een enkele digitale outputstroom (aggrerate signaal, 256 kb/s). Het is duidelijk dat de outputstream (minstens) 4 keer de bitstream moet zijn van de individuele inputs.
Er zijn twee verschillende vormen: *bit interleaved TDM* en byte interleaved TDM.

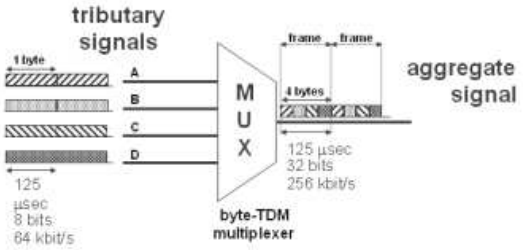
**bit interleaved multiplexing**




Bij deze vorm van TDM worden de gepaste poortpulsen toegepast op de verschillende inputs, en door het samentellen van de outputs of de gepoorte signalen, bekomt men het output signaal.

**byte interleaved multiplexing**

Bij deze vorm wordt kan men een frame zien dat herhaald wordt om de 125 $\mu$sec. Dit frame zal altijd dezelfde vorm hebben, die zal starten

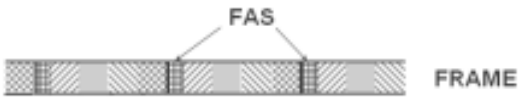



met een byte van het eerste tributary (inkomend signaal multiplexer), en zal eindigen met een byte van het laatste tributary. Dit frame zal continu herhaald worden. Men zal een frame alignment signal (FAS) toevoegen om het begin van een frame te kunnen herkennen.

We bekijken nog enkele belangrijke punten bij het versturen en (de)multiplexen van digitale signalen.

Een *klok* is vereist zowel aan de verzender als ontvanger kant, die bovendien hetzelfde moet zijn aan beide kant, omdat anders bits niet kunnen opgehaald worden (omdat men het exacte moment wanneer gesampled moet worden moet kennen). Klok extractie kan gebeuren gebaseerd op de start/stop bits, de synchronisatiebits of -informatie in het line coding proces.

Het moet mogelijk zijn om de bytes te bekomen uit de individuele bits. Dit wordt ondersteund door het gebruik van een frame structuur, die enkele bepaalde bits zal bevatten in een elk frame: het *Frame Alignment Signal (FAS)*, een gekend patroon van bits. Op deze manier kunnen die bits gedetecteerd worden, en zo de positie van een frame bepaald worden, zodat ook de individuele bytes kunnen bepaald worden.

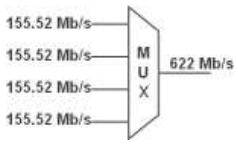



9.13 ***Wat is het verschil tussen een synchroon en een plesiochroon signaal?***

Wanneer verschillende signalen gemultiplexed worden tot een hoger bitrate signaal, dan is het belangrijk dat alle tributary signalen dezelfde bitrate hebben. Dit is niet altijd het geval, waardoor er twee multiplexing technieken zijn: *synchroon* en *plesiochroon*.

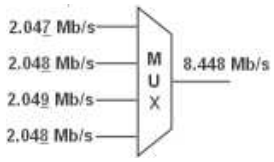
**synchroon**

Bij synchroon multiplexing hebben de tributary signalen exact dezelfde bitrate. Daardoor is het mogelijk om een aggregaat signaal te bekomen dat een exact veelvoud is van de tributary signalen. In de figuur zien we een Synchronous Digital Hierarchy (SDH) waar 4 tributaries van 155,52 Mbit/s (exact) gemultiplexed worden in een 622 Mbit/s signaal.




**plesiochroon**

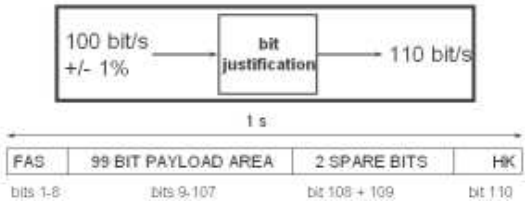
Bij plesiochroon multiplexing hebben de verschillende tributary signalen licht verschillende bitrates (in de orde van 10 tot 100 ppm verschillen). Met plesiochroon bedoelen we quasi-synchroon. In de figuur zien we een Plesiochronous Digital Hierarchy (PDH) waar 4 tributaries van 2,048 Mbit/s (gemiddeld) gemultiplexed worden in een 8,448 Mbit/s signaal.




Het aggeregate signaal is meer dan 4 keer een 2.048 signaal, omdat het in staat moet zijn 4 signalen met bitrates die verschillen van de nominale 2.048 Mbit/s te dragen. Die verschillen zullen opgelost worden door het gebruik van bit justification.

9.14 ***Leg het principe uit van bit-justification. Waarom wordt het gebruikt?***

Het *justification proces* wordt gebruikt om het multiplexen van asynchrone lagere orde tributary signalen in een aggerate signaal toe te laten. Het probleem hierbij is, hoe kan een vaste frame structuur gebruikt worden om verschillende tributary signalen te bevatten met licht andere bitrates, en nog steeds, in elk frame, een integraal aantal bits bevatten.
We leggen het proces uit m.b.v. een voorbeeld. De nominale bitrate in het voorbeeld is 100 bit/s (tussen 99 en 101 bits/s). We moeten een signaal bekomen dat gesynchroniseerd is op 110 bit/s.

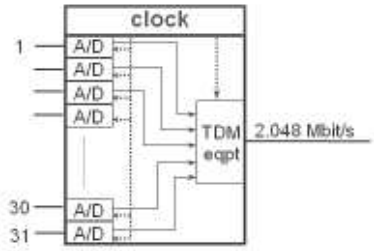



Het *payload* gebied is 99 bits lang, en er zijn twee reserver bits voor justificatie. Als de bitrate 99 bit/s is, dan worden de 99 bits opgeslaan in het payload gebied van het frame. Als de bitrate 100 bit/s is, dan zal het eerste frame 99 bits transporteren, en het volgende frame 101 bits transporteren (99 bits in de payload, en 2 in het reserver gebied). Als de bitrate 101 bits/s is, dan zal elk frame 101 bits bevatten. Als een frame 101 bits

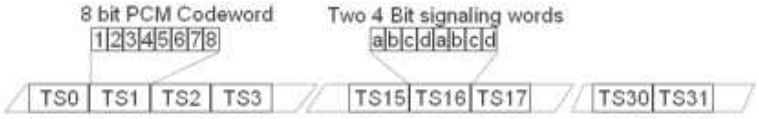
transporteert, wordt dit aangeduidt d.m.v. de *House Keeping (HK)* bit. Merk op dat nog andere bitrates ook mogelijk zijn. Bvb. een bitrate van 99,5 bit/s zal ervoor dat iedere 4de frame 2 extra bits zal bevatten. De house keeping bit zorgt er ook voor dat het reconstrueren van de originele bitrate van het tributary signaal mogelijk is.

9.15 ***Wat is een primaire multiplexer? Bespreek kort.***

Plesiochronous Digital Hierarchy (PDH) is een transmissie standaard die verschillende niveau's van multiplexen standaardiseerd. In die standaardisatie wordt op het eerste niveau synchroon multiplexen toegepast: van 64 kbit/s naar 2,048 Mbit/s. De multiplexer die hiervoor zorgt wordt een *primaire multiplexer genoemd*, en maakt gebruik van byte interleaved TDM. We bekijken de multiplexer m.b.v. onderstaande figuur. De multiplexer




combineert 32 signalen van 64 kbit/s in een aggregate signaal van 2048 kb/s. Hierbij worden 30 signalen analoge stemcircuits (1-15 rn 17-31) gebruikt en 1 signaling circuit. Het overige circuit wordt gebruikt voor voor framing en alarms. De verschillende analoge stemsignalen worden omgezet naar een digitaal signaal van 64 kb/s (waarbij A-law Pulse Code Modulation gebruikt wordt). De multiplexer maakt van byte interleaved multiplexing, en de line coding is het HDB3 systeem.
De frame structuur van de primaire multiplexer is weergegeven in onderstaande figuur.




De frame duur is 125 $\mu$sec en het frame bevat 32 tijdsloten van 8 bits, dus 256 bits in totaal. Een tijdslot van 8 bits wordt elke 125 $\mu$sec herhaald, wat resulteerd in een bitrate van 64 kb/s, en de aggregate bitrate is dus 2.048 Mb/s (met een toegelaten variatie van $\pm$ 50 ppm).
Het eerste tijdslot (TS0) wort gebruikt voor framing en alarm informatie, het 17de tijdslot (TS16) wordt gebruikt voor signaling. Het is zeer belangrijk voor het demultiplexen dat het signaal aan de start van het frame gedetecteerd kan worden. Dit gebeurt door het FAS in TS0 (een 7 bit sequentie 0011011, in de bits 2 tot 8 van TS0), en dit enkel voor even genummerde frames. De overige bit wordt gebruikt voor andere doeleinden.

9.16 ***Wat is PDH?***

*Plesiochronous Digital Hierarchy (PDH)* is een transmissie standaard die verschillende niveau's van multiplexen standaardiseerd. Op het eerste niveau wordt synchroon byte interleaved TDM gebruikt (primaire multiplexer). De hogere multiplexeringsstappen gebruiken plesiochroon bit interleaved TDM. De verschillende niveau's (voor Europa) staan in onderstaande tabel.

**Europe**

| Bitrate kbit/s | multiplex factor |
|---|---|
| 64 | |
| 2048 | 30/31 |
| 8448 | 4 |
| 34368 | 4 |
| 139264 | 4 |

We bekijken de tweede orde multiplexing frame structuur met een bitrate van 8.448 kb/s ± 30 ppm, die positieve justification en bit interleaved multiplexing gebruikt. Er zijn 4 tributary signalen van 2048 kb/s, de frame duur is 100,4 $\mu$sec, en de frame lengte is 848 bits. Elk tributary signaal kan 206 bits in een frame innemen (met de stuff bits inbegrepen), en een frame bevat 4 setsvan 212 bits. De maximale justification rate per tributary signaal is ongeveer 10kb/s.
De figuren tonen de frame structuur helemaal, en de eerste 20 bits van elke set van een frame meer in detail.

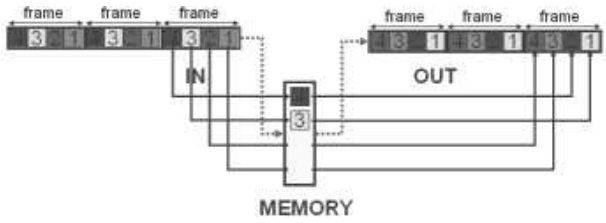
| Frame structure | Bit number |
|---|---|
| | SET 1 |
| Frame alignment signal (1111010000) | 1 to 10 |
| Alarm indication to remote multiplexer | 11 |
| Bit reserved for national use | 12 |
| Bits from tributaries | 13 to 212 |
| | SET 2 |
| **Justification control bits** $C_{i1}$ | **1 to 4** |
| Bits from tributaries | 5 to 212 |
| | SET 3 |
| **Justification control bits** $C_{i2}$ | **1 to 4** |
| Bits from tributaries | 5 to 212 |
| | SET 4 |
| **Justification control bits** $C_{i3}$ | **1 to 4** |
| Bits from tributaries available for **justification** | **5 to 8** |
| Bits from tributaries | 9 to 212 |

| SET | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 1 | JC1 | JC1 | JC1 |
| 2 | 1 | JC2 | JC2 | JC2 |
| 3 | 1 | JC3 | JC3 | JC3 |
| 4 | 1 | JC4 | JC4 | JC4 |
| 5 | 0 | T1 | T1 | JB1 |
| 6 | 1 | T2 | T2 | JB2 |
| 7 | 0 | T3 | T3 | JB3 |
| 8 | 0 | T4 | T4 | JB4 |
| 9 | 0 | T1 | T1 | T1 |
| 10 | 0 | T2 | T2 | T2 |
| 11 | AI | T3 | T3 | T3 |
| 12 | R | T4 | T4 | T4 |
| 13 | T1 | T1 | T1 | T1 |
| 14 | T2 | T2 | T2 | T2 |
| 15 | T3 | T3 | T3 | T3 |
| 16 | T4 | T4 | T4 | T4 |
| 17 | T1 | T1 | T1 | T1 |
| 18 | T2 | T2 | T2 | T2 |
| 19 | T3 | T3 | T3 | T3 |
| 20 | T4 | T4 | T4 | T4 |

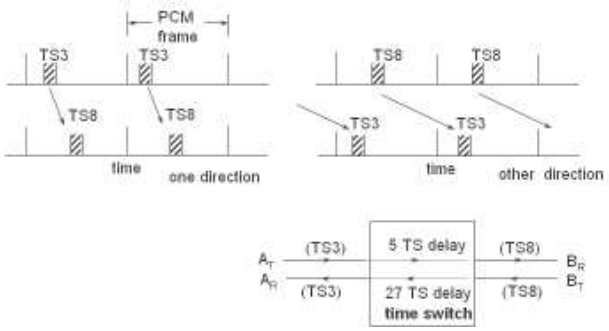
De eerste 10 bits van de eerste set vormen het FAS. De eerste 4 bits van de sets 2, 3 en 4 dienen als controlebits voor de justification bit van het overkomstige tributary signaal. But 11 van set 1 is een alarm bit, bit 12 is gereserveerd voor eventuele andere doeleinden. De bits 5-8 van de vierde set zijn de 4 justification bits voor de 4 tributary signalen. Alle overige bits in de sets zijn telkens bits die komen met de bits die resulteren uit de bit interleaved multiplexing (telkens in de volgorde 1,2,3,4).

9.17 ***Bespreek het principe van time switching.***

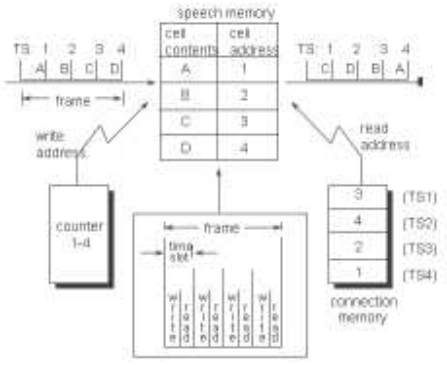
Het doel van een time switch (die ook Time Slot Interchange (TSI) genoemd wordt), is het overbrengen van de inhoud van een tijdslot naar een




ander, niet samenvallend, tijdslot. Daardoor zal de bedoelde vertraging optreden in een time switch.



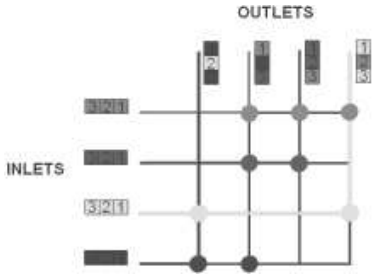

We bekijken een voorbeeld waarbij we de inhoud van TS3 naar TS8 willen overbrengen, die een vertraging van 5 tijdsloten vereist. Als we een frame beschouwen met 32 tijdsloten (de 2 Mb/s standaard), dan moeten we we 27 tijdsloten wachten om de inhoud van TS8 naar TS3 over te brengen (dit is nodig omdat we een bidirectionele verbinding hebben), waarbij we de frame grens moeten overschrijven.



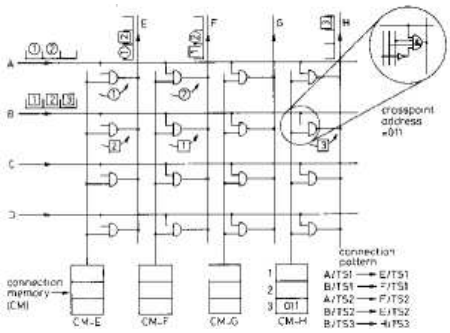

We bekijken een voorbeeld van een time switch. Typisch aan een time switch is dat die slechts 1 input en 1 output bus heeft. In het voorbeeld beschouwen we een frame van 4 tijdsloten. De inhoud van de tijdsloten worden opgeslaan in volgorde van aankomst (cyclic write) in het speech memory (SM) deel, en ze worden gelezen in een volgorde die bepaald wordt door het connection memory (acyclic read) deel. Het is ook mogelijk om een acyclic write / cycllic read implementatie te maken.

Time switchen laten ook, net als space switches, een super-multiplexing werking toe. Een aantal PCM (Pulse Code Manipulation) gemultiplexeerde kanalen zullen in parallel behandeld worden, waarvoor een hoge werkingssnelheid nodig is van de time switch.
Als men een time switch maakt met 16 PCM signalen (2 Mb/s) aan de input, en 16 PCM signalen aan de output, dan is het mogelijk om de 512 resulterende tijdsloten met elkaar te verbinden. Dit betekent dat een een niet-blokkerende werking mogelijk is in een time swtich (in tegenstelling tot een space switch). Merk op dat slechts één fysieke input en één fysieke output beschouwd worden (in tegenstelling tot een space switch). In de praktijk zal het gebruik van time switches alleen beperkt worden, en daarom zullen ze gecombineerd worden met space switches.

9.18 ***Bespreek het principe van space switching.***




Een digitale space switch bestaat uit een time division gemultiplexeerde matrix met ingangen en uitgangen die PCM (Pulse Code Modulation) signalen vervoeren. Om een bepaald tijdslot in een inkomend PCM signaal te verbinden met het corresponderende tijdslot (die dus hetzelfde tijdslotnummer (TS) heeft) van een uitgaand PCM signaal, moet het gepaste cross-point van de space-switch gebruikt worden voor de duur van dat tijdslot, en dit moet ieder frame herhaald worden (voor de duur van de gesprekken). Op andere momenten, kan de space switch gebruikt worden om de inhoud van andere gesprekken tussen tijdsloten te swtichen.
We bespreken een voorbeeld van een space switch. Het connection mem-

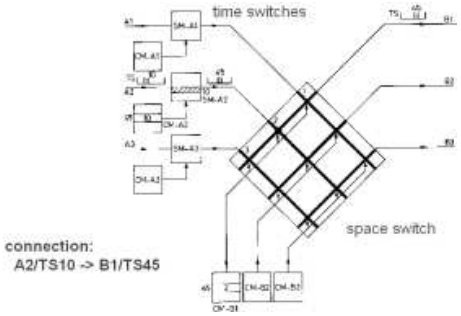



ory (CM) zal de cyclus bevatten van de cross-points die gesloten moeten worden. Die cyclus zal ieder frame herhaald moeten worden. In dit voorbeeld hebben we een frame met 3 tijdsloten en een 4x4 space switch.
We zien dat hier veel minder cross-points nodig zijn dan bij een analoge

space switch. Als we 40 PCM signalen beschouwen (2 Mb/s die correspondeert met 30 telefoonkanalen), dan hebben we een 40x40 space switch nodig, dus met 1600 cross-points. Als we 40x30 telefoonsignalen in een analoge exchange moet switchen, hebben we ongeveer 1,5 miljoen cross-points nodig ($40 \times 30 = 1200 \rightarrow 1200 \times 1200 = 1.440.000$).
Een van de grote problemen bij de implementatie van een space switch is de hoge graad van blokkering, omdat er een grote kans is dat twee of meerdere gesprekken hetzelfde tijdslot willen gebruiken. Daarom zal een combinatie met een time switch nodig zijn.

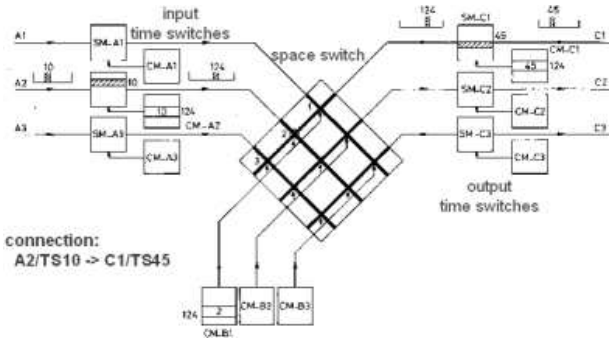
9.19 ***Waarom zal men time en space switching combineren?***

Een van de grote problemen bij de implementatie van een space switch is de hoge graad van blokkering, omdat er een grote kans is dat twee of meerdere gesprekken hetzelfde tijdslot willen gebruiken. Time switches hebben dat probleem niet, maar het gebruik van time switches alleen zal beperkt worden, en daarom zullen ze gecombineerd worden met space switches.




Eerst bekijken het principe van een *Time-Space (T-S) switch*, die een time switch (cyclic writing, acyclic reading) bevat aan elk input bussen van een enkele space switch. In het bovenstaande voorbeeld, wordt TS10 van input A overgebracht naar TS45 op output B1. Als we echter ook een tijdslot verschillend van TS10 van A2 naar TS45 op B2 of B3 willen overbrengen, zal deze geblokkeerd worden, omdat TS45 reeds gebruikt wordt. Een T-S switch zal wel een grotere capaciteit hebben dan een enkele T-switch, maar het lijdt wel nog steeds onder het inherent blokkeren van corresponderende tijdsloten op de output bus van de time switch.
Een *Space-Time (S-T) switch* is gelijkaardig aan een T-S switch, maar de space switch verbindt de input en output bussen eerst, en daarna zal de time switch de nodige time slot vertragingen verzorgen. Het blokkeringsprobleem blijft omdat slechts één van de inputs van de space switch een verbinding kan maken met een output bus gedurende een van de tijdsloten.

Bij een *Time-Space-Time (T-S-T) switch* verbindt de input time switch het input tijdslot aan eender welk vrij tijdslot op de bus naar de space switch input. De output time switch verbindt het gekozen tijdslot van de space switch naar het gepaste tijdslot aan de output. In het voorbeeld brengen we het tijdslot TS10 van input A2 over naar TS45 van output C1. Dit gebeurt via TS124 binnen in de space switch. Merk op dat de input




time switch cyclic-write/acyclic read is, en de output acyclic-write/cyclic-read.
Op gelijkaardige manier kunnen we een *Space-Time-Space (S-T-S) switch* implementeren. De input space switch verbindt de input bus naar een time switch gedurende het input tijdslot. De output space switch verbindt de time switch met de output bus gedurende het output tijdslot.
Zowel T-S-T als S-T-S switches zijn in staat om even hoge capaciteiten te leveren, met lage blokkeringsprobabiliteit.
Merk op dat een space switch groeit als het kwadraat van het aantal input en output bussen, terwijl een time switch lineair groeit met het aantal tijdsloten. Daarom worden space switches opgedeeld in verschillende delen (voor de realisatie van switches met grote capaciteit), om hun grootte te beperken, waardoor implementaties ontstaan als T-S-S-T of T-S-S-S-T. Het gebruik van space switches in verschillende delen reduceert de kost, maar tegelijktijd neemt de blokkeringsprobabiliteit toe.