**Introductie netwerken – afstandslessen**

**Transportlaag – TCP**

Voegt functionaliteit toe:

* Segmentatie
* Connection setup
* Error control
* Opnieuw zenden van verloren segmenten
* Flow control = Congestion control

Segmentatie:

Een bestand dat verstuurd moet worden, wordt opgedeeld in kleinere stukjes of segmenten. Ze moeten ook allemaal naar hetzelfde adres verstuurd worden: transportlaag – poorten, netwerklaag – ip-adressen, datalink laag – frames.

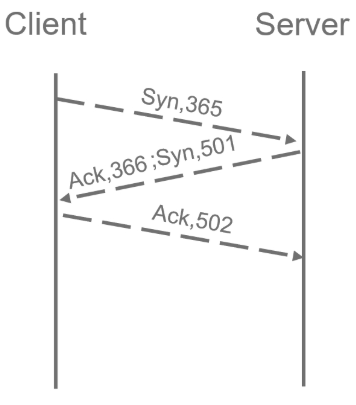
Door segmentatie kan de netwerkkaart niet gemonopoliseerd worden door één enkele applicatie. Elke applicatie krijgt even tijd om elk een segmentje door te sturen.

Sequence numbers:

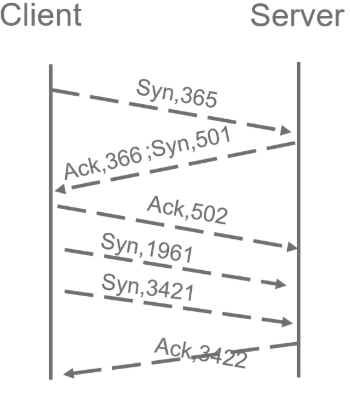
Veel van de diensten TCP levert, steunen op het gebruik van sequence numbers. Eigenlijk wordt elke byte van 1 nummer voorzien. Deze sequence numbers kunnen ook gebruikt worden om segmenten die in de verkeerde volgorde arriveren weer in de juiste volgorde te steken.

Connection setup – TCP 3-way handshake

De te gebruiken sequence numbers worden tussen de zender en ontvanger gecommuniceerd voordat de data verstuurd wordt. Dit gebeurt in 3 stappen.

1. De client start altijd de sessie! VB: de client start de sessie en kiest een nummer om de eerste byte aan te duiden, bvb “**Syn, 365”**.
2. De server bevestigt de ontvangst van byte 365 door te vragen naar byte 366. “**Ack, 366**”. Direct stuurt de server ook de nummer door waarmee hij de eerste byte zal aanduiden. Bvb 505 (=”Piggybacking”). “**Syn, 501**”.
3. De client bevestigt de ontvangst van byte 501 door te vragen naar byte 502. “**Ack, 502**”

Versturen van data

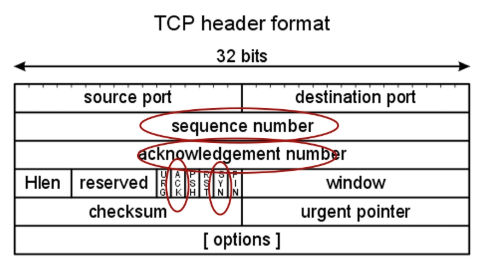
Tot nu toe is er nog geen enkele byte verstuurd die tot de data behoort. Nu zal de client de data, segment per segment, doorsturen.

Als we een segmentatie van 1460 bytes veronderstellen, zal de laatste byte van het eerste segment nummer 1825 hebben. “**Syn, 1825**”.

Nu zal hij het volgende segment sturen, de laatste byte van het tweede segment heeft dan 3421 als nummer. “**Syn, 3421**”.

De server bevestigt dat hij de gezonden segmenten heeft ontvangen, dit doet hij door steeds de laatste te bevestigen (Met Ack). “**Ack, 3422**”.

TCP header – TCP 3-way handshake

Het veld sequence number wordt gebruikt om de sequence numbers mee te verzenden.

Het veld acknowledgement number wordt gebruikt om de laatst ontvangen byte van een segment te bevestigen.

In de eerste stap van de 3-way handschake staat de syn-bit op 1

In de tweede stap staat de syn-bit en de ack-bit op 1

In de derde stap staat de ack-bit op 1 **(zie TCP 3-way handshake beetje terug)**

TCP is reliable

Segementen die veloren of corrupt zijn worden opnieuw verstuurt.

Om dit te kunnen doen worden verzonden segmenten door de zender bewaard in een buffer tot de goede ontvangst is bevestigd (Acknowledged)

Error control

De TCP-header voorziet in een veld “checksum”. De checksum is het resultaat van een berekening met als input o.a de verzonden data en de TCP-header. Zodra het segment is aangekomen, doet de zender de ontvanger dezelfde berekening op basis van dezelfde velden als de zender.

Als de cheksum’s niet hetzelfde zijn, is het segment corrupt gegaan, zo kan de zender weten wanneer hem terug te sturen.

Negative Ack

De ontvanger gaat verzoeken het segment opnieuw te sturen (“**NAck**”)

Opnieuw zenden van verloren segmenten

Als een segment verloren is, kan de ontvanger de ontvangst niet bevestigen. Als de zender na een bepaalde tijd nog steeds geen bevestiging heeft gekregen, zal hij het segment gewoon opnieuw sturen.

Sliding Windows

Zowel aan de kant van de zender, maar ook aan die van de ontvanger zijn buffers nodig om tijdelijk de segmenten te bewaren.

Buffers hebben een beperkte opslagcapaciteit, deze kunnen dus vol geraken (“buffer overflow”). Regelmatig moeten segmenten hier dus uit verwijderd worden om overflow te vermijden.

Wanneer data uit buffers verwijderen?

* + Zender: wanneer ze goed ontvangen zijn -> ACK
  + Ontvanger: regelmatig buffers leegmaken door segmenten aan de applicatielaag door te geven.

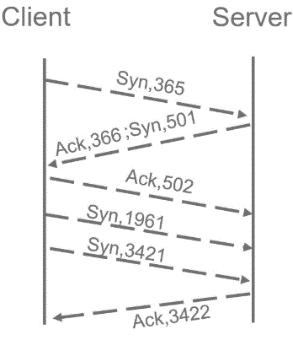
Congestion control

Congestion = teveel zenders / teveel verzonden data / te snel verzonden data waardoor het netwerk de trafiek niet aankan.

Symptomen:

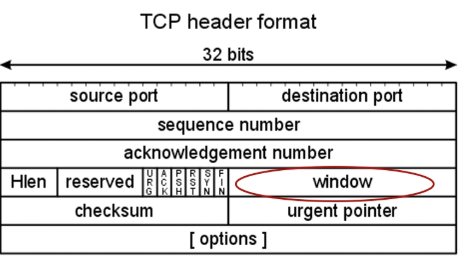
* + Verloren segmenten (packets dropped) door buffer overflow bij routers
  + Grotere vertragingen (queueing in router buffers)

Werking congestoin control

De ontvanger zal de ontvangst van elk segment niet appart bevestigen. De ontvanger zal dit op voorhand communiceren (TCP Recieve Window).

De zender zal op basis hiervan bepalen hoeveel segmenten hij kan versturen alvorens hij op een bevestiging moet wachten.

TCP Recieve Window

Mocht de ontvanger tijdens de sessie nog andere data ontvangen van andere zenders, dan zal er een nieuwe lagere waarde gecommuniceerd moeten worden.

De zender kan dan het aantal segmenten dat “geburst” wordt hierop aanpassen.

**De vorm van een ipv4-adres:**

Een ip-adres is een binair getal, bestaande uit 32 bits. Omdat het voor ons moeilijk is om met deze lange binaire adressen te werken, gaan we deze herschrijven, we zetten dit om naar een decimaal getal.

1. Deel de 32 bits op in groepjes van 8 bits, om deze groepjes te scheiden gebruiken we een punt (.). Zo 1 groepje van acht bits noemen we een octet.
2. Zet elke octet om naar een decimaal getal.

Voorbeeld: 11000000101010000000000000000001

1. 11000000.10101000.00000000.00000001
2. 192.168.0.1

**Betekenis van het netmasker:**

Decimale notatie

Als je data wilt sturen naar een computer, gebeurt dit naar een computer in een bepaald netwerk. Om data te versturen moeten we dus weten: in welk netwerk de computer staat, welke computer het is in dat bepaalde netwerk. -> Een ip adres moet dus een deel bevatten om het netwerk aan te duiden.

Vb: 11000000101010000000000000000001

Welke bits zijn dat juist het netwerkdeel, en welke het hostdeel? -> Wordt aangeduid door de netmask.

11000000101010000000000000000001 -> ip-adres -> 192.168.0.1

11111111111111111111111100000000 -> netmask -> 255.255.255.0

Hier zien we dat de eerste 24 bits (1) uitdrukken wat het netwerk aanduid. De overige 8 (0) duiden aan welke computer in dit netwerk het gaat.

192.169.0.0 = network-id , 1 = host-id

Alternatieve notatie van een netmask

We kunnen ook gewoon achter het ip-adres een “slash” (/) en daarachter het aantal bits voor het netwerkdeel.

In het geval van het voorbeeld zou het dan: “**192.168.0.1/24**” zijn.

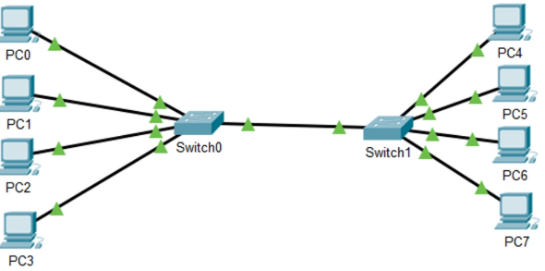
**Logische netwerken:**

Broadcast domain

Broadcast = bericht dat naar iedereen gestuurd wordt. Broadcast krijg je als je een bericht stuurt naar het broadcast adres.

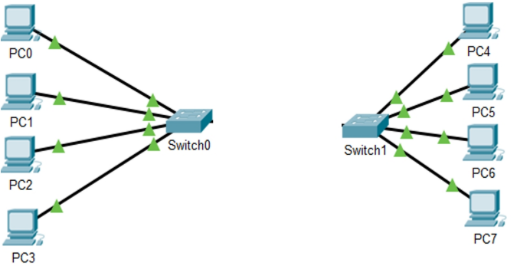
Alle **intermediary devices**, behalve de routers, sturen de data bestemd voor **het broadcast adres** verder over alle verbonden poorten.

**Broadcast domain:** al de nodes die kunnen bereikt worden door data te zenden naar het broadcast adres.

Hoeveel broadcast domain tel je in dit voorbeeld? – 1, als er geen router is, is het 1 groot broadcast domain.

Hoeveel hosts zullen betrokken zijn als PC0 een broadcast bericht verstuurt? – 8, hijzelf en de andere 7 in het broadcast domain

Segmenteren van broadcast domains

Om de impact van broadcasts te beperken gaan we het netwerk **segmenteren**.

Nu zullen er dus 2 broadcast domains zijn met beide 4 nodes.

Als we nu een broadcast willen verzenden naar een ander segment, kunnen we dit doen door de segmenten te verbinden met een router.

Connecteren van broadcast domains:

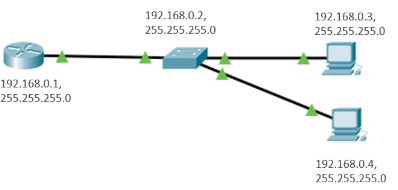
Toch kunnen alle hosts nog niet met elkaar communiceren.

* **Routers verwachten dat de broadcast domains een ID hebben.** Zo kunnen ze deze van elkaar onderscheiden.

**Routeren** = pakketjes doorsturen tussen verschillende broadcast domains. Routers kunnen enkel routeren als zij de broadcast domains van elkaar kunnen onderscheiden.

Elke broadcast domain moet daarom een identiteit (= **networkID** ) krijgen.

Elke host kan nu aangeduid worden met een **hostID**, bevindend in een bepaald netwerk met een **netwerkID**.

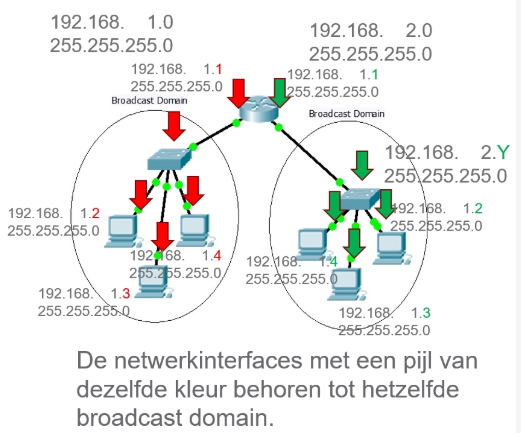
Voorbeeld van een correct geadresseerd netwerk.

Al de hosts die verbonden zijn met dezelfde poort van de router **en** deze routerpoort zelf, bevinden zich dan in hetzelfde **broadcast domain.**

Deze hosts moeten voor hun ip-adres hetzelfde **networkID** krijgen, hier is dat **192.168.0**

Het netwerkID bestaat uit 24 bits, dus het **netmask** is dan 255.255.255.0

De hosts in dit broadcast domain moeten voor het **hostID** van hun ip-adres een unieke waarde krijgen zodat ze te onderscheiden zijn in dit netwerk.

Toewijzen van ip-adressen

Alle netwerkinterfaces die zich in hetzelfde broadcast domain bevinden, moeten zich logisch in hetzelfde netwerk bevinden. Dat betekent dat het netwerkdeel van deze ip-adressen hetzelfde moet zijn.

Het hostdeel van deze ip-adressen moet voor elke computer in dit netwerk uniek zijn.

\*192.168.2.Y moet 192.168.2.1 zijn en het netwerkdeel moet voor dit broadcast domain 192.168.**2** zijn

**De betekenis van een default gateway:**

Broadcast – Routers

Bij gebruik van het IPv4-protocol start communicatie (meestal) met een broadcast ARP.

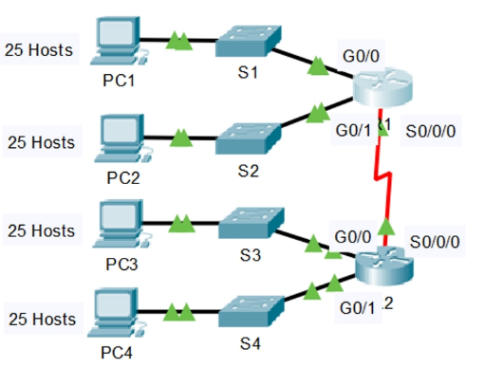
Nog andere applicaties in de IPv4-stack zijn broadcast based. Broadcasts hebben een negatieve impact op netwerk. Routers spelen een rol in het minimaliseren van die impact. -> Zie les ‘Logische netwerken’

**Het concept van subnetten:**

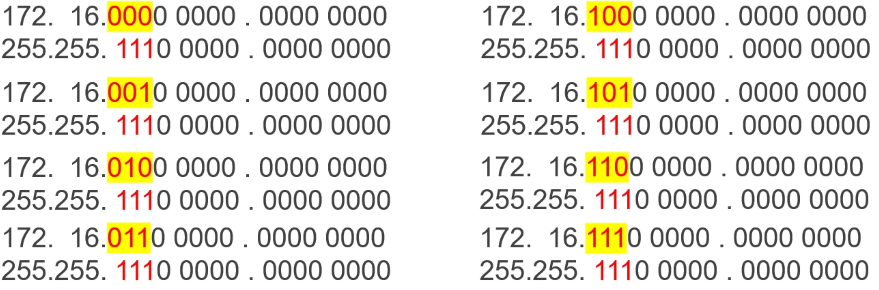
Doel van subnetten

1 classful netwerkadres opdelen in meerdere netwerkadressen in functie van de fysische topologie van het netwerk.

Scenario



* netwerk met 5 broadcast domains.
* 1 classful netwerkadres
  + 172.16.0.0
  + 255.255.0.0
* 16 bits over om 2^16 bitcombinaties te maken die allen de octetten 172.16.x.x gemeenschappelijk hebben
* Die bitcombinaties gaan we gebruiken om
  + Minstens 5 verschillende combinaties te gebruiken als netwerkadres (5 broadcast domains)
  + De overige bitcombinaties te gebruiken als ip-adressen in deze netwerken
* Hoeveel van die 16 bits zullen we gebruiken
  + Voor de netwerk-adressen?
  + Voor de ip-adressen in die netwerken?
* Bepalen van het aantal nodige netwerkadressen
  + 172.16.0000 0000 . 0000 0000
  + 255.255.0000 0000 . 0000 0000
* 5 broadcast domains, dus 5 netwerkadressen nodig.
  + 3 bits -> 2 is te weinig, 2^2 is namelijk maar 4, wat niet genoeg is
  + Verlengen van de netmask met het nodige aantal bits
    - 255.255. 1110 0000. 0000 0000
* Bepalen van welke bitcombinaties je kan maken met de nieuwe netwekbits



* + -> Omzetten naar decimale waarden
* De netwerkadressen toekennen aan de fysische broadcast domains

**De ip-adressen in een subnet:**

Ip-adressen

1. Plaatsing van routers deelt het netwerk op in kleinere broadcast domains.
2. Aan de fysische broadcast domains, worden logische netwerk-adressen toegekend.
3. De fysische broadcast domains worden gebruikt om computers met elkaar te verbinden.
4. Elke node die we met het netwerk verbinden, moet een geldig adres (ip-adres) krijgen in dit netwerk

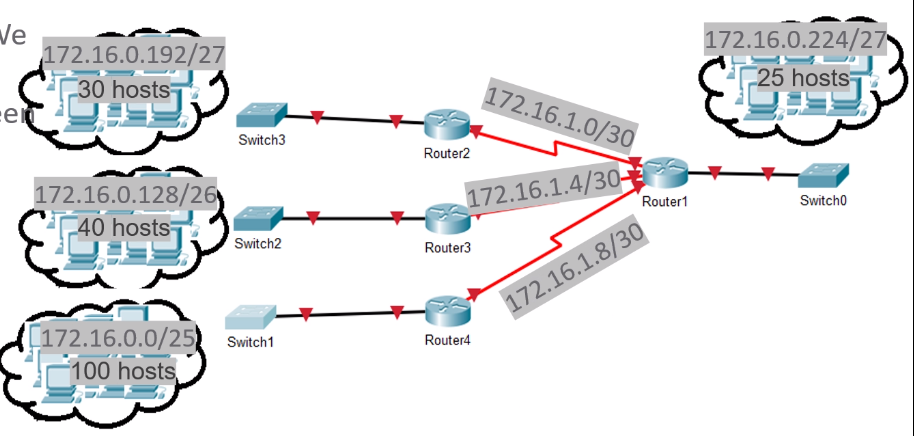
Concept – geldige ip-adressen in een subnet

* Voorbeeld subnet 172.16.64.0, 255.255.224.0
* 172.16.0100 0000. 0000 0000 -> hostdeel
* Wat zijn de mogelijke bit-combinaties?
  + Kleinste geldig adres: 172.16.0100 0000 . 0000 0001
  + Grootste geldig adres: 172.16.0101 1111. 1111 1110
  + (172.16.0101 1111 . 1111 1111 is een broadcast domain)
* Decimaal:
  + Kleinste: 172.16.64.1
  + Grootste: 172.16.95.254

**VLSM**

Waarom VLSM?

Als we steeds dezelfde netmask gebruiken, zullen veel ip adressen ongebruikt blijven.



**Schrijfwijze van een Ipv6-adres**

Bestaat uit 128 bits, we zetten deze samen in groepjes van 4 bits.

Elke groepje van 4 bits wordt omgezet naar een hexadecimaal karakter, deze worden nog eens in groepjes van 4 gezet, met een colon ( : ) ertussen

0010000000000001000011011011100010101010101010100001000100010001000000000000000000000000000000000000000000000000000 0000100000000

->

0010 0000 0000 0001 0000 1101 1011 1000 1010 1010 1010 1010 0001 0001 0001 0001 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001 0000 0000

->

20010DB8AAAA11110000000000000100

->

2001:0DB8:AAAA:1111:0000:0000:0000:0100

Worden:

* Voorgesteld als acht 16-bit segmenten = hextetten
* Deze hextetten gaan van 0000 naar FFFF
* De hextetten worden gescheiden door een colon ( : )

Waarom IPv6:

Er is zoveel vraag naar ip-adressen dat op basis van 32-bits lange adressen er onvoldoende ip-adressen zouden zijn.

IPv6 is vanaf nul ontwikkeld en houdt meer rekening met de hedendaagse noden dan IPv4

Verkorte schrijfwijze: Leading zero’s

Nullen aan het begin van het segment ( = leading zero’s ) moeten niet geschreven te worden

3ffe : 0404 : 0001 : 1000 : 0000 : 0000 : 0ef0 : bc00

->

3ffe : 404 : 1 : 1000 : 0 : 0 : ef0 : bc00

Verkorte schrijfwijze: Double colon ( :: )

Een aaneensluitende reeks nullen mag vervangen worden met een dubbele colon ( :: )

ff02 : 0000 : 0000 : 0000 : 0000 : 0000 : 0000 : 0500 - > ff02 :: 500

Network prefixes

In IPv4 wordt de prefix – dat is het netwerkdeel van het ip-adres aangeduid met een netmask of een bitcount

255.255.255.0 of /24

Bij IPv6 gebruiken we alleen de bitcount ( prefix length)

Bv **/64** -> Het netwerkdeel is 64 bits lang