

UNIVERSITEIT ANTWERPEN

Academiejaar 2023-2024

Faculteit Toegepaste Ingenieurswetenschappen

## Datacommunicatie

## 3-Netwerkarchitectuur

**Guido Van Landeghem**

### Reprografie-cursusdienst

Rosalie Delforge

Universiteit Antwerpen  
Groenenborger campus

[Rosalie.Delforge@universitas.be](mailto:Rosalie.Delforge@universitas.be)

T +32(0)3 360 80 89

**Bachelor of Science in de  
industriële wetenschappen: elektronica-ICT**

**STUDIEGIDSNR 1507FTINWA**

# INHOUDSTABEL

<b>1 De fysieke laag.....</b>	<b>30</b>
<b>1.1 Signalen en theoretische concepten.....</b>	<b>30</b>
1.1.1 Tijd- en frekwentiedomein.....	30
1.1.2 Frekwentiespectrum en bandbreedte .....	32
1.1.3 Analoge en digitale transmissie.....	40
1.1.4 Transmissievervorming .....	43
1.1.5 Transmissiecapaciteit.....	47
<b>1.2 Transmissiimedia.....</b>	<b>50</b>
1.2.1 Twisted pair .....	52
1.2.2 elektronische beperkingen op kopermedia.....	55
1.2.3 COAX.....	62
1.2.4 Fiber.....	64
<b>1.3 Datatransmissie.....</b>	<b>73</b>
1.3.1 Parallelle transmissiemethodes .....	73
1.3.2 Seriële transmissiemethodes .....	74
1.3.3 Bitsynchronisatie bij synchrone transmissie .....	81
1.3.4 Karakter- en frame-synchronisatie bij <u>byte</u> -georiënteerde synchrone transmissie	
89	
1.3.5 Karakter- en frame-synchronisatie bij <u>bit</u> -georiënteerde synchrone transmissie.	90
<b>1.4 Interface standaards op de fysieke laag.....</b>	<b>92</b>
1.4.1 Standaard Ethernet 10 base.....	92
1.4.2 Fast ethernet.....	95
1.4.3 Gigabit ethernet.....	99
1.4.4 10 Gigabit ethernet, IEEE 802.3ae .....	103
1.4.5 40/100 Gigabit ethernet.....	105
1.4.6 RS 232 .....	107
1.4.7 RS422 A / V11 en RS485 .....	109
<b>1.5 Het telefoonstelsel.....</b>	<b>113</b>
1.5.1 Analoge telefonielijnen.....	116
1.5.2 Digitale telefonielijnen .....	118
1.5.3 xDSL / ADSL, (Asymmetric) Digital Subscriber Line .....	126
<b>2 De datalink-laag.....</b>	<b>131</b>
<b>2.1 Ontwerpvereisten van datalink-protocollen.....</b>	<b>134</b>
2.1.1 Methodes voor foutencontrole.....	134
2.1.2 Herstel van transmissiefouten 'error control'.....	140
2.1.3 Flow control.....	147
2.1.4 Transparantie .....	150
<b>2.2 Het HDLC protocol .....</b>	<b>154</b>
2.2.1 HDLC toepassingsomgevingen.....	155
2.2.2 HDLC structuur.....	159

<b>2.3 Praktische datalink-protocollen.....</b>	<b>168</b>
2.3.1 LAPB.....	168
2.3.2 LAPM.....	169
2.3.3 SLIP, Serial Line Interface Protocol.....	171
2.3.4 PPP, Point to Point Protocol.....	172
2.3.5 De datalink laag in LAN's ➔ LLC.....	176
<b>2.4 De medium access controle deellaag.....</b>	<b>181</b>
2.4.1 Fysieke topologieën.....	182
2.4.2 CSMA/CD Medium access contole voor een bus.....	183
<b>2.5 Ethernet - IEEE 802.3 .....</b>	<b>188</b>
2.5.1 Standaard ethernet 10baseXX .....	188
2.5.2 IEEE 802.3 frame formaat.....	192
2.5.3 Fast ethernet.....	195
2.5.4 Gigabit ethernet.....	197
2.5.5 Evolutie in Ethernet .....	198
2.5.6 Verschil tussen ethernet en IEEE 802.3 frames .....	201
<b>2.6 Token passing op een token ring IEEE 802.5.....</b>	<b>206</b>
<b>3 De netwerklaag.....</b>	<b>211</b>
<b>3.1 Internetten : architectuur en ontwerpaspecten.....</b>	<b>211</b>
3.1.1 Architectuur van internetten .....	212
3.1.2 Ontwerpaspecten van de netwerklaag.....	217
3.1.3 Het TCP/IP model.....	223
<b>3.2 Het IP datagram .....</b>	<b>225</b>
<b>3.3 IP adressering .....</b>	<b>229</b>
3.3.1 IP adres-structuur – VOLLE KLASSE 'class-based'.....	229
3.3.2 Oefeningen op volle Klasse netwerken ( <b>ter info!!</b> ) .....	232
3.3.3 IP subnet – adressering .....	236
3.3.4 Oefeningen op subnetting van netwerken.....	244
3.3.5 IP VLSM subnet – adressering.....	256
3.3.6 CIDR Classless InterDomain Routing .....	259
3.3.7 IP intranetten en network address translation.....	263
3.3.8 Domain name system.....	265
<b>3.4 IP routing .....</b>	<b>268</b>
3.4.1 Routeringsprincipes .....	269
3.4.2 Routeringsprotocollen .....	271
3.4.3 Distance Vector Routing.....	275
3.4.4 Link State Routing.....	281
3.4.5 BGP, Border Gateway Protocol .....	291
<b>3.5 IP verpakking .....</b>	<b>293</b>
3.5.1 Address Resolution Protocol ARP en RARP .....	293
3.5.2 DHCP - Dynamic Host Configuration Protocol .....	297
3.5.3 Ethernet verpakking van IP pakketten.....	298
3.5.4 Fragmentatie .....	300
<b>4 De transportlaag.....</b>	<b>303</b>

<b>4.1 L4 Adressering : PORTS .....</b>	<b>306</b>
<b>4.2 UDP User Datagram Protocol .....</b>	<b>310</b>
<b>4.3 TCP Transmission Control Protocol .....</b>	<b>311</b>
4.3.1 TCP header .....	315
4.3.2 TCP protocol werking.....	318
4.3.3 TCP protocol werking : UITGEBREID.....	324
4.3.4 TCP congestion controle .....	332
4.3.5 TCP aanpassingen .....	337
4.3.6 TCP op een onbetrouwbare netwerkservice.....	338
4.3.7 TCP implementatie policy's .....	342
<b>5 Appendices .....</b>	<b>344</b>
<b>5.1 OSI protocols.....</b>	<b>344</b>
OSI model .....	344
OSI begrippen : SAP 's.....	344
OSI begrippen : service primitieven.....	347
OSI: protocolfuncties. ....	351
<b>5.2 overzicht TCP/IP over EtherNet.....</b>	<b>353</b>
5.2.1 TCP / IP over EN. : <b>Zenden</b> van een boodschap .....	354
5.2.2 TCP / IP over EN. : <b>Ontvangen</b> van een boodschap .....	355

# FIGURENLIJST

Figuur 1 Communicatie entiteiten .....	20
Figuur 2 'point to point' computerverbindingen a) rechtstreeks, b) via een modem .....	21
Figuur 3 LAN verbindingen van computers. ....	22
Figuur 4 LAN topologien : bus, ster en ring. ....	22
Figuur 5 Een bedrijfsnetwerk van automatiseringscomputers.....	23
Figuur 6 Een privaat netwerk. ....	23
Figuur 7 Publieke datanetwerken : a) PSDN .....	24
Figuur 8 Publieke datanetwerken : b) ISDN.....	24
Figuur 9 Een wereldwijd internettwerk.....	25
Figuur 10 Een breedband ISDN netwerk. ....	25
Figuur 11 Standaards en hun instellingen. ....	26
Figuur 12 De evolutie van een internet standaard. ....	27
Figuur 13 Structuur van het ISO model .....	28
Figuur 14 Indeling van het ISO model .....	28
Figuur 15 Functieoverzicht van de ISO layers .....	29
Figuur 16 Tijd- en frekwentiedomein voorstelling van een sinusgolf. ....	30
Figuur 17 Tijd- en frekwentiedomein voorstelling van drie sinus golven. ....	31
Figuur 18 Fourier analyse van een blokgolf. ....	33
Figuur 19 Fourier analyse van een 2e blokgolf. ....	34
Figuur 20 Fourier analyse van een zaagtand.....	35
Figuur 21 Fourier analyse van een niet periodiek signaal. ....	36
Figuur 22 Bandbreedte van een periodiek signaal. ....	36
Figuur 23 Tijd- en frekwentiedomein voorstellingen van een digitaal signaal. ....	37
Figuur 24 Transmissie van een digitaal signaal. ....	37
Figuur 25 signaalvervorming door de beperkte bandbreedte van een kanaal. ....	37
Figuur 26 'Aangepaste' of analoge transmissie van een digitaal signaal. ....	38
Figuur 27 beperkte bandbreedte van een UTP CAT5 kabel.....	38
Figuur 28 Effect van de kanaalsbandbreedte op een digitaal signaal .....	39
Figuur 29 Analoge transmissie van data.....	40
Figuur 30 AM, FM en PM modulatie van een digitale data. ....	40
Figuur 31 Digitale transmissie van data. ....	41
Figuur 32 Effecten van verwakking en vervormingen op een digitaal signaal. ....	43
Figuur 33 verwakking van een signaal. ....	44
Figuur 34 dB calculatie. ....	44
Figuur 35 vervorming van een signaal. ....	45
Figuur 36 Ruis. ....	46
Figuur 37 2B1Q baudreducerende code .....	48
Figuur 38 transmissiemedia, guided vs unguided. ....	50
Figuur 39 Verzwakking van transmissiemedia i.f.v. de frekwentie. ....	50
Figuur 40 Een 2-draadsverbinding .....	51
Figuur 41 Twisted pair kabel .....	52
Figuur 42 Twisted pair kabel 'twist length' .....	52
Figuur 43 Twisted pair kabel (UTP) .....	52
Figuur 44 Twisted pair kabel (STP).....	52
Figuur 45 Twisted pair kabel (FTP) .....	52
Figuur 46 Twisted pair kabel (SFTP) .....	52
Figuur 47 Vergelijking tussen UTP CATx bekabeling. ....	53
Figuur 48 Patch panel / horizontale bekabeling.....	54
Figuur 49 LAN bekabelingsstructuur.....	54

Figuur 50 kabeleigenschappen van UTP.....	55
Figuur 51 Attenuation to crosstalk ratio .....	56
Figuur 52 Evolutie in UTP kabel .....	56
Figuur 53 signaalbanen bij ethernet 10/100BASE.....	57
Figuur 54 Near End Cross Talk power spectrum .....	57
Figuur 55 NEXT cancellers en golfvormen .....	58
Figuur 56 Near End en Far end Cross Talk.....	59
Figuur 57 Powersum NEXT en FEXT.....	59
Figuur 58 Powersum NEXT en FEXT.....	60
Figuur 59 PSAELFEXT .....	60
Figuur 60 Karakteristieke impedantie van twisted pair lijnen .....	61
Figuur 61 echo/return loss en verzwakking/attenuation op UTP. ....	61
Figuur 62 Opbouw van een coax kabel. ....	62
Figuur 63 HFC coax bandbreedte .....	62
Figuur 64 Praktische LAN-thick ethernet-bekabeling .....	63
Figuur 65 Praktische LAN-thin ethernet-bekabeling .....	63
Figuur 66 lichtrefractie in een fiber .....	64
Figuur 67 Fiber doorsnede. ....	64
Figuur 68 Opbouw van een fiber .....	64
Figuur 69 Omzetting van elektrische signalen naar licht.....	65
Figuur 70 Transmissiemodes in een fiber .....	66
Figuur 71 Dispersie van pulsen in een multimode fiber .....	66
Figuur 72 spectrum van de gebruikte EM golven door glasvezels .....	68
Figuur 73 Dempingsverloop van een glasvezel.....	68
Figuur 74 Principe van Wave Division Multiplexing. ....	69
Figuur 75 Verzwakking in een fiber ifv. $\lambda$ . ....	69
Figuur 76 DWDM vs CWDM spacing.....	69
Figuur 77 Bochten en onzuiverheden in een fiber.....	70
Figuur 78 dispersie in een fiber.....	70
Figuur 79 Spirale propagatie van licht in een multimodefiber.....	70
Figuur 80 vergelijkende tabel voor 10G ethernet. ....	71
Figuur 81 Fiber SC en MT-RJ connectoren .....	72
Figuur 82 Parallelle transmissie. ....	73
Figuur 83 Seriele transmissie. ....	74
Figuur 84 Asynchrone seriele transmissie.....	75
Figuur 85 Asynchrone transmissie.....	75
Figuur 86 frame synchronisatie bij asynchrone transmissie. ....	76
Figuur 87 framesynchronisatie bij synchrone P2P transmissie.....	77
Figuur 88 framesynchronisatie bij synchrone MAC transmissie.....	77
Figuur 89 een unipolaire NRZ lijncode. ....	79
Figuur 90 een polaire NRZ lijncode.....	79
Figuur 91 foute synchronisatie bij digitale transmissie. ....	80
Figuur 92 Synchrone transmissie, klokcodering. ....	81
Figuur 93 Een bipolaire RZ coderingsschema. ....	81
Figuur 94 Manchester en differential manchester coderingsschema's.....	82
Figuur 95 Klokcoderingen bij synchrone transmissie in LAN's.....	83
Figuur 96 manchester code, een praktisch signaal. ....	83
Figuur 97 Coderingsschema's gebruikt in WAN omgevingen. ....	84
Figuur 98 AMI en ASI Coderingsschema's .....	85
Figuur 99 B8ZS Coderingsschema.....	85
Figuur 100 2B1Q Coderingsschema. ....	86
Figuur 101 Powerspectrum van AMI en manchester codes .....	87

Figuur 102 DPLL kloksynchronisatie .....	88
Figuur 103 DPLL werking .....	88
Figuur 104 Karakter en frame-synchronisatie bij byte-georiënteerde synchrone transmissie.....	89
Figuur 105 Karakter en frame-synchronisatie bij bit-georiënteerde synchrone transmissie .....	91
Figuur 106 ethernet evolutie .....	92
Figuur 107 ethernet bustopologie .....	92
Figuur 108 ethernet 10base-T : stertopologie .....	93
Figuur 109 Sterbekabeling bij ethernet 10baseT .....	93
Figuur 110 Frame vorm bij IEEE 802.3 / ethernet .....	94
Figuur 111 100 base T4 aansluiting en codering. ....	96
Figuur 112 100 base TX en FX aansluiting en codering. ....	97
Figuur 113 100 base TX MLT3 lijncodering .....	98
Figuur 114 1000base fysieke laag .....	99
Figuur 115 1000baseX fysieke laag .....	100
Figuur 116 1000baseX fysieke laag .....	100
Figuur 117 1000baseT op CAT5e .....	101
Figuur 118 4D (dimensional) 5 level basisband PAM code .....	101
Figuur 119 1000baseT fysieke laag .....	102
Figuur 120 10 Gigabit ethernet evolutie.....	103
Figuur 121 10 Gigabit ethernet opties. ....	104
Figuur 122 40- en 100 Gbase SR4. ....	105
Figuur 123 40- en 100 Gbase lijncodering 10Gbps per lane.....	105
Figuur 124 40- en 100 Gbase lane concept. ....	105
Figuur 125 RS232, V28 "unbalanced" signalen.....	107
Figuur 126 RS232 bitpatroon .....	107
Figuur 127 20mA current loop signalen.....	108
Figuur 128 RS 422 / V11 "balanced" signalen.....	109
Figuur 129 maximum afstand/bitrate voor RS422.....	109
Figuur 130 Spanningen op een RS422/485 lijn. ....	110
Figuur 131 RS422 – four-wire Network.....	110
Figuur 132 RS485 – two-wire multidrop Network .....	111
Figuur 133 RS485 – four-wire multidrop Network .....	112
Figuur 134 Telefonie aansluiting local loop. ....	113
Figuur 135 Frequentieband voor xDSL. ....	114
Figuur 136 het kabel TV netwerk, HFC Hybrid Fiber Coax bekabeling.....	114
Figuur 137 Bandbreedte gebruik bij HFC. ....	114
Figuur 138 toekomstbeeld voor de 'last mile'.....	115
Figuur 139 Modulatievormen : b)AM c) FM en d) PM.....	116
Figuur 140 16 en 256- QAM constellatie. ....	116
Figuur 141 Een 8-QAM golfvorm .....	117
Figuur 142 Digitalisatie van geluid. ....	118
Figuur 143 Companding van geluidssignalen. ....	118
Figuur 144 A/D conversie van een spraakkanaal. ....	119
Figuur 145 A law en $\mu$ law companding. ....	119
Figuur 146 ISDN basic access aansluiting. ....	119
Figuur 147 TDM mux van spraakkanaalen. ....	120
Figuur 148 T1 en E1 frames. ....	120
Figuur 149 Een SDH STM-1 module.....	122
Figuur 150 Een SONET STS-1/OC-1 frame .....	122
Figuur 151 Een SDH controle systeem.....	123
Figuur 152 SONET / SDH multiplex hierarchie. ....	123
Figuur 153 Inplanting van een TUG-2 in een STM-1 frame. ....	124

Figuur 154 Een SDH frame formaat van STM-n.....	125
Figuur 155 Een SDH trunk lijn. ....	125
Figuur 156 Gehuurde lijnen : kostprijs.....	125
Figuur 157 Klassieke ADSL aansluiting. ....	126
Figuur 158 ADSL frekwentiebereikindeling. ....	126
Figuur 159 DMT bits per kanaal allocatie.....	127
Figuur 160 ADSL2+ frekentieband. ....	127
Figuur 161 VDSL2 frekentieband. ....	128
Figuur 162 ADSL2+ tov. VDSL2 afstanden. ....	128
Figuur 163 Een ADSL loop architectuur. ....	128
Figuur 164 Bit swapping op een ADSL lijn. ....	129
Figuur 165 . xDSL in een triple play architectuur.....	130
Figuur 166 De datalink : a) virtuele ; b) feitelijke communicatie. ....	131
Figuur 167 Communicatie L2 – L3 over een router. ....	132
Figuur 168 Situering van het datalink-protocol .....	133
Figuur 169 Rij – en kolom-pariteit. ....	136
Figuur 170 Een CRC berekening. ....	137
Figuur 171 Een hardware CRC generator. ....	138
Figuur 172 Een hardware CRC ontvanger. ....	139
Figuur 173 Idle RQ, implicit (+ explicit) retransmission .....	141
Figuur 174 Continuous request foutvrije communicatie.....	142
Figuur 175 Go back N : een fout I frame.....	143
Figuur 176 Go back N : een fout ACK frame .....	144
Figuur 177 selective repeat – implicit retransmission : a) fout I-frame, b) fout ACK frame. ....	145
Figuur 178 selective repeat – explicit retransmission , fout I-frame.....	146
Figuur 179 Sliding window flow control.....	147
Figuur 180 Maximum te verzenden frames in Go-back-N .....	148
Figuur 181 Windowing techniek voor een venstergrootte K = 7, modulo M=8.....	149
Figuur 182 Transparantie voor een asynchroon byte-protocol d.m.v. bytestuffing.....	150
Figuur 183 Transparantie voor een synchroon byte-protocol d.m.v. bytestuffing.....	151
Figuur 184 Transparantie voor een bit-georiënteerd point to point protocol d.m.v. bitstuffing.....	152
Figuur 185 Transparantie voor een bit-georiënteerd LAN protocol d.m.v. bits tellen. ....	153
Figuur 186 Transparantie voor een bit-georiënteerd LAN protocol d.m.v. bit-encoding violations .....	153
Figuur 187 Data link protocol point to point .....	155
Figuur 188 Fysische lay-out van a) multipoint en b) multidrop netwerken. ....	156
Figuur 189 Logische lay-out van multipoint / multidrop netwerken.....	156
Figuur 190 Poll-Select'werking bij multipoint/multidrop verbindingen. ....	157
Figuur 191 Een BSC – select sequentie. ....	157
Figuur 192 Data link protocol bij WAN's .....	158
Figuur 193 Data link protocol bij LAN's .....	158
Figuur 194 HDLC – netwerkconfiguraties .....	159
Figuur 195 HDLC frame formaat. ....	160
Figuur 196 HDLC frame types in standaard en extended vorm.....	161
Figuur 197 Link management in a) NRM, b) ABM. ....	163
Figuur 198 Dataverkeer op HDLC in NRM mode, Go-back-N, implicit ACK .....	164
Figuur 199 Dataverkeer op HDLC in NRM mode, Go-back-N, explicit ACK .....	165
Figuur 200 Dataverkeer op HDLC in ABM mode, foutloos, met 'piggy backing'	166
Figuur 201 LAPB werking : link, service primitieven en state diagram. ....	168
Figuur 202 Een modemverbinding.....	169
Figuur 203 LAPM protocol tussen 2 communicerende modems.....	169
Figuur 204 Een P2P internet verbinding .....	171
Figuur 205 Slip verpakking van een IP datagram .....	171

Figuur 206 Het PPP frame formaat .....	172
Figuur 207 Gebruik van PPP en SLIP. ....	173
Figuur 208 Situering van PPP in de TCP/IP suite .....	174
Figuur 209 Het PPP fase diagram. ....	174
Figuur 210 Een PPP – LCP pakket voor configure request. ....	175
Figuur 211 Opsplitsing datalink laag in 2 sublagen voor LAN's .....	176
Figuur 212 Datalink structuur bij LAN's.....	176
Figuur 213 LLC werking : 1) CL, unack en 3) ack, 2) CO .....	178
Figuur 214 LLC frame format (a) en controle veld (b).....	179
Figuur 215 LLC werking : overzicht. ....	180
Figuur 216. Indeling van de MAC principes.....	181
Figuur 217 Bus en Ring topologie van LAN's.....	182
Figuur 218 Fysische hub/tree topologie van LAN's.....	182
Figuur 219 Tijdsverloop (theoretisch) van een collision.....	183
Figuur 220 Praktisch verloop van een collision. ....	184
Figuur 221 CSMA/CD kent 3 toestanden : transmissie, contentie en rust. ....	184
Figuur 222 CSMA/CD collisions. ....	185
Figuur 223 a) Round trip delay op ethernet b) channel efficiency op ethernet. ....	186
Figuur 224. 10BASE5 dikke coax .....	188
Figuur 225. 10BASE2 dunne coax. ....	189
Figuur 226 Het schema van een 10base5 transceiver .....	189
Figuur 227. 10Baset Twisted Pair. ....	190
Figuur 228 Hub configuratie topologie en inwendig schema.....	190
Figuur 229. 10BASEF Fiber .....	191
Figuur 230 IEEE 802.3 frame formaat. ....	192
Figuur 231 De zendprocedure voor IEEE802.3 .....	194
Figuur 232 De ontvangstprocedure voor IEEE802.3.....	194
Figuur 233 Hub/tree afstand tussen DTE's. ....	195
Figuur 234 Fast ethernet frame. ....	196
Figuur 235 Een voorbeeld van een Gigabit ethernet configuratie. ....	197
Figuur 236 Een burst frame in een Gigabit ethernet configuratie.....	197
Figuur 237 evolutie van coax naar UTP in ethernet.....	198
Figuur 238. 1 collision domain. ....	198
Figuur 239 evolutie naar L2 scheiding in ethernet. ....	198
Figuur 240. Opsplitsing van het collision domain.....	199
Figuur 241. Full duplex ethernet aansluiting. ....	199
Figuur 242 Full duplex in ethernet.....	199
Figuur 243 Flow control in ethernet.....	200
Figuur 244 Ethernet vs IEEE stack .....	201
Figuur 245 Datalink frame formaat a) ethernet b) IEEE 802.3 .....	201
Figuur 246 IEEE's STP 802.1d encapsulatie door 802.3 – 802.2 .....	203
Figuur 247 Ethernet_SNAP frame.....	203
Figuur 248 Overzicht ethernet – IEEE 802.3 protocollen .....	204
Figuur 249 Principe van een token ring netwerk. ....	206
Figuur 250 Fysische layout van een token ring netwerk. ....	207
Figuur 251 De werking van een TCU voor token ring. ....	207
Figuur 252 IEEE 802.5 frame formaat : a) token, b) normaal frame. ....	208
Figuur 253 IEEE 802.5 veldopbouw. ....	208
Figuur 254 Foutdetectie op een ring-netwerk .....	210
Figuur 255 Het internetconcept . ....	212
Figuur 256 LAN – WAN verbindingen.....	212
Figuur 257 Verbinding tussen (sub)netwerken.....	213

Figuur 258 Een Tokenring netwerk.....	213
Figuur 259 Een Wireless uitbreiding op een ethernet netwerk.....	213
Figuur 260 Totaalbeeld van een internetverbinding.....	214
Figuur 261. Routering over ISPs .....	215
Figuur 262. Router forwarding.....	215
Figuur 263 Intermediate Systems : a) router, b) protocol converter.....	216
Figuur 264 Internettwerk services.....	217
Figuur 265 Verband tussen NSAP en NPA.....	219
Figuur 266. L2 switching op basis van MAC adressen.....	219
Figuur 267 IPv6 netwerkadressen. ....	220
Figuur 268 De TCP/IP protocol suite.....	223
Figuur 269 Het IP datagram formaat. ....	225
Figuur 270 TOS bits in een IP header. ....	226
Figuur 271. Een 16-bit checksum berekening, bv. van de IP header.....	227
Figuur 272. IP adressen notatie en indeling. ....	228
Figuur 273 IP adres klassen .....	229
Figuur 274 IPv4 adres space .....	230
Figuur 275 hosts op een netwerk.....	231
Figuur 276 Toekenning van IP adressen. ....	233
Figuur 277 Principe van subnetting .....	236
Figuur 278 Subnetting van een Klasse B adres .....	236
Figuur 279. Subnetmasker lengte .....	237
Figuur 280 Subnetting van een Klasse B site op de bytegrens .....	237
Figuur 281 Een klasse B netwerk met subnets op de byte-grens .....	238
Figuur 282 subnet 1 van klasse B netwerk met SNM = 255.255.255.0 .....	240
Figuur 283 Toekenning van IP adressen. ....	245
Figuur 284 Toekenning van IP adressen. ....	248
Figuur 285. Een bedrijfsnetwerk, klasse C.....	249
Figuur 286. Een bedrijfsnetwerk, te subnetten. ....	250
Figuur 287 Oefening op IP-subnetting .....	256
Figuur 288 IP variabel subnetmasker toekenning. ....	257
Figuur 289 Extended-network-prefix-length voorstelling. ....	258
Figuur 290. CIDR toewijzing van een /20 aan een ISP .....	259
Figuur 291. CIDR router tabel .....	260
Figuur 292. Route tabel van R1 in detail. ....	261
Figuur 293 IP network address translation .....	263
Figuur 294 De Domain Name Hierarchie .....	265
Figuur 295 Name to address resolution protocol .....	266
Figuur 296 Een nieuw domein toevoegen.....	267
Figuur 297 Forwarding door routers. ....	268
Figuur 298. Graph voorstelling van een netwerk. ....	268
Figuur 299. Hoe vindt een router zijn weg. ....	270
Figuur 300. Een praktische routeringstabell. ....	270
Figuur 301 De fysische topologie van het Internet. ....	272
Figuur 302 Logische topologie van internetten.....	272
Figuur 303 Intradomain- and Interdomain Routering.....	273
Figuur 304. Least cost trees van routers .....	274
Figuur 305 Distance Vector algorithme. ....	276
Figuur 306 Routing loop. ....	277
Figuur 307 RIP message .....	280
Figuur 308 Link state routing algoritme .....	283
Figuur 309 Oefening op link state routing. ....	284

Figuur 310 Een autonomoos systeem en zijn graph voorstelling.....	286
Figuur 311 De SPF tree voor router 6 .....	287
Figuur 312 Hierarchie in OSPF netwerken. ....	288
Figuur 313 OSPF Packet Header Format.....	289
Figuur 314 BGP message formaten. ....	292
Figuur 315 Ethernet verpakking van een ARP pakket.....	294
Figuur 316 ARP en RARP pakket. ....	294
Figuur 317 Een voorbeeld van een proxy ARP.....	296
Figuur 318. DHCP toekennen van IP adressen. ....	297
Figuur 319. DHCP uitwisseling. ....	297
Figuur 320 Datalink frame formaat : ethernet II en IEEE 802.3 - 802.2.....	298
Figuur 321 Vergelijking tussen ethernet en IP adressen.....	299
Figuur 322 Fragmentatie a) intranet b) internet....	300
Figuur 323 De werkbare lagen in een netwerk .....	303
Figuur 324. Domein van de transportlaag. ....	303
Figuur 325 De TCP/IP protocol suite.....	304
Figuur 326 TCP en UDP stack met de interlayer address-selectors.....	305
Figuur 327. Transportlaag interne routering via poortnummers. ....	306
Figuur 328. IANA poort nummer indeling .....	307
Figuur 329. een socket address .....	307
Figuur 330. Applicatie multiplexing in de transportlaag.....	308
Figuur 331 TCP en UDP sockets bij source en destination.....	308
Figuur 332 UDP datagram formaat.....	310
Figuur 333 pseudoheader bij de berekening van een UDP (of TCP) checksum.....	310
Figuur 334. TCP, een stroom service. ....	311
Figuur 335 TCP segment header en codebits .....	315
Figuur 336. Explicit Congestion Notificatin vlaggen in TCP.....	316
Figuur 337 Het opzetten van een TCP-connectie. ....	318
Figuur 338. TCP data transfer.....	320
Figuur 339. TCP 3-way close. ....	321
Figuur 340. TCP, een half-open connectie. ....	322
Figuur 341 Een abrupt TCP einde.....	323
Figuur 342. TCP receive buffer en window.....	325
Figuur 343. TCP send buffer en window. ....	325
Figuur 344. TCP window werking. ....	326
Figuur 345 TCP data transfer.....	328
Figuur 346 Een praktische TCP connectie opgezet. ....	329
Figuur 347 Het credit-based flow controle schema van TCP.....	330
Figuur 365 Slow start van TCP.....	333
Figuur 366 TCP slow start en congestion avoidance.....	334
Figuur 367 TCP fast retransmit. ....	335
Figuur 368 TCP congestion window aanpassing bij duplicate ACK's → Fast recovery.....	336
Figuur 369 TCP congestion window aanpassing bij RTO's → slow start.....	336
Figuur 370 TCP window 'scale option'. ....	337
Figuur 364 Het probleem van 'verloren' datasegmenten. ....	340

# AFKORTINGEN

## Hoofdstuk 1: DE FYSIEKE LAAG

10Base2	10Mbit/sec, Basisband, max 200m. (Thin EN)
10Base5	10Mbit/sec, Basisband, max 500m. (Thick EN)
2B1Q	2 Bits op 1 Quaternair niveau
4B3T	4 Bits op 3 Ternaire niveaus
4B5B	4 Bits op 5 Binaire niveaus
ACR	Attenuation to Crosstalk Ratio
ADSL	Asymmetric Digital Subscriber Line
AM	Amplitude Modulation
AMI	Alternate Mark Inversion
ANSI	American National Standards Institute
AP	Access Point
ASI	Alternate Space Inversion
ATM	Asynchronous Transfer Mode
ATU-C	Adsl Transceiver Unit – Central office
ATU-R	Adsl Transceiver Unit – Remote
AU	Administrative Unit
AUG	Administrative Unit Group
AUI	Attachment Unit Interface
B82S	Bipolar with 8 Zero Compression
C	Container
CEPT	Conference European of Post and Telecommunications
CSMA/CA	Carrier Sense Multiple Access / Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access / Collision Detection
CTS	Clear To Send
DCE	Data Circuit-terminating Equipment
DFTP	Dual Foiled Twisted Pair
DG	Data Grade (CAT5)
DMT	Discrete MultiTone
DPLL	Digital Phase Locked Loop
DSE	Data Switching Exchange
DSLAM	Digital Subscriber Line Access Multiplexer
DSSS	Direct Sequence Spread Spectrum
DTE	Data Terminal Equipment
DWDM	Dense Wave Division Multiplexing
ECMA	European Computers Manufacturers Association
EIA	Electrical Industries Association
EMI	Electro-Magnetic Interference
EN	Ethernet
EOF	End of Frame
ETX	End of Text
FDM	Frequency Division Multiplexing
FHSS	Frequency Hopping Spread Spectrum
FM	Frequency Modulation
FTP	Foiled Twisted Pair
FTTD	Fiber To The Desk
FTTH	Fiber To The Home

HDB3	High Density Bipolar 3
HF	High Frequency
HFC	Hybrid Fiber Coax
I <sup>2</sup> C	Inter IC Communication
IEEE	Institution of Electrical & Electronics Engineers
ILD	Injection Laser Diode
IR	Infra Red
ISDN	Integrated Services Digital Network
ISI	Inter Symbol Interference
ISMN	Industrial Scientific & Medical Network
ISO	International Standardization Organization
ISP	Internet Service Provider
ITU-T	International Telecommunications Union – Telecommunications sector
LAN	Local Area Network
LD	Laser Diode
LF	Low Frequency
LTE	Line Termination Equipment
MAC	Medium Access Control
MAN	Metropolitan Area Network
NAV	Network Allocation Vector
NEXT	Near End Cross Talk
NID	Network Interface Device
NMS	Network Managing System
NRZ	Non Return to Zero
NTE	Network Termination Equipment
OSI	Open System Interconnection
P2P	Point to Point
PABX	Private Automatic Branch Exchange
PAM	Phase Amplitude Modulation
PBX	Private ( telephone ) Branch Exchange
PCM	Pulse Code Modulation
PDH	Plesiochronous Digital Hierarchy
PISO	Parallel In Serial Out
PLL	Phase Locked Loop
PM	Phase Modulation
POTS	Plain Old Telephone System
PRI	Primary Rate Interface
PSDN	Public Switched Data Network
PTE	Path Termination Equipment
QAM	Quadrature Amplitude Modulation
RTS	Request To Send
RZ	Return to Zero
SDH	Synchronous Digital Hierarchy
SFTP	Shielded and Foiled Twisted Pair
SIPO	Serial In Parallel Out
SNR	Signal to Noise ratio
SOF	Start of Frame
SOHO	? p44 ?
SONET	Synchronous Optical NETwork
SPE	Synchronous Payload Envelope
STE	Section Termination Equipment

STM	Synchronous Transport Module / Mode
STP	Shielded Twisted Pair
STX	Start of Text
TC	Terminal Controller
TDM	Time Division Multiplexing
TU	Tributary Unit
TUG	Tributary Unit Group
UART	Universal Asynchronous Receiver – Transmitter
USB	Universal Serial Bus
UTP	Unshielded Twisted Pair
VC	Virtual Container
VDSL	Very-high-data-rate Digital Subscriber Line
VG	Voice Grade (CAT3)
VLF	Very Low Frequency
VLSI	Very Large Scale Integration
VSAT	Very Small Aperture Terminal
WAN	Wide Area Network
WDM	Wave Division Multiplexing
WLAN	Wireless Local Area Network
WNIC	Wireless Network Interface Card

## Hoofdstuk 2: DE DATALINK-LAAG

ABM	Asynchronous Balanced Mode
AC	Access Control
ACK	Acknowledge
Ack-CL	Acknowledged Connection Less
Ack-CO	Acknowledged Connection Oriented
ADCCP	Advanced Data Communication Control Procedure
AMP	Active Monitor Present
AP	Access Point
ARM	Asynchronous Response Mode
ARQ	Automatic Repeat Request
BCC	Blocksum Check Character
BEC	Backward Error Control
BER	Bit Error Rate
BP	Branching Point
BSC	Bisync / Binary Synchronous Control
BSS	Basic Service Set
CCITT	Consultative Committee for International Telegraph and Telephone
CL	Connection Less
CO	Connection Oriented
CRC	Cyclic Redundancy Check
CS	Carrier Sense
CSLIP	Compressed Serial Line Interface Protocol
CSMA/CA	Carrier Sense Multiple Access / Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access / Collision Detection
CT	Claim Token
DA	Destination Address
DAT	Duplicate Address test
DCE	Data Circuit-termination Equipment
DCF	Distributed Coordination Frame

DFWMAC	Distributed Foundation Wireless MAC
DIFS	DCF IFS
DLE	Data Link Escape
DLP	Data Link Protocol
DM	Disconnect Mode
DS	Distribution Service / System
DSAP	Destination Service Access Point
DSSS	Direct Sequence Spread Spectrum
DTE	Data Termination Equipment
ECP	Error Correction Part
ED	End Delimiter
EOF	End Of Frame
EOT	End Of Transmission
ESD	End of Stream Delimiter
ESS	Extended Service Set
ETX	End of Text
EXCH	Exchange
FC	Frame Control
FCS	Frame Check Sequence
FDDI	Fiber Distributed Data Interchange
FEC	Forward Error Control
FHSS	Frequency Hopping Spread Spectrum
FIFO	First In First Out
FILO	First In Last Out
FRMR	FRaMe Reject
FS	Frame Status
HDLC	High-Level Data Link Control
IFS	InterFrame Space
ISM	Industrial, Scientific and Medical (frekwentieband)
LADB	Link Access Procedure Balanced (X25)
LAPD	Link Access Procedure D-channel (ISDN)
LAPM	Link Access Procedure Modem
LCP	Link Control Protocol
LLC	Logical Link Control
LWE	Lower Window Edge
MAC	Multiple Access Control
MTU	Maximum Transfer Unit
NAK	Not Acknowledge
NCP	Network Control Protocol
NIC	Network Interface Card
NLPID	Next Layer Protocol IDentifier
NRM	Normal Response Mode
NTE	Network Termination Equipment
OUI	Organization Unique Identifier
PCF	Point Coordination Function
PIFS	PCF IFS
PISO	Parallel In Serial Out
PPP	Point to Point Protocol
PS	Packet Switching
PSE	Packet Switching Exchange
REJ	REject

RNR	Receiver Not Ready
RR	Receiver Ready
RSET	ReSET
SA	Source Address
SABM	Set Asynchronous Balanced Mode
SABME	Set Asynchronous Balanced Mode Extended
SAP	Service Access Point
SARM	Set Asynchronous Response Mode
SARME	Set Asynchronous response Mode Extended
SD	Start Delimiter
SDLC	Synchronous Data Link Control
SDU	Service Data Unit
SFD	Start of Frame Delimiter
SIFS	Short InterFrame Space
SIPO	Serial In Parallel Out
SLIP	Serial Line Interface Protocol
SNA	Simple Network Architecture
SNAP	SubNetwork Attachment Point
SNRM	Set Normal Response Mode
SNRME	Set Normal Response Mode Extended
SOF	Start Of Frame
SREJ	Selective REject
SSAP	Source Service Access Point
SSD	Start of Stream Delimiter
STX	Start of Text
TCU	Trunk Coupling Unit
UIP	User Interface Part
Unack-CL	Unacknowledged Connection Less
UP	Unnumbered Poll
UWE	Upper Window Edge
VC	Virtual Circuit
XID	eXchange IDentification
802.2	Logical Link Control
802.3	Ethernet
802.4	Token Bus
802.5	Token Ring
802.11	Wireless

### Hoofdstuk 3: DE NETWERK-LAAG

ABR	Area Border Router
AP	Application Progress
ARP	Address Resolution Protocol
ARPA	Advanced Research Projects Agency
AS	Autonomous System
ASBR	Autonomous System Border Router
BER	Bit Error Rate
BGP	Border Gateway Protocol
CIDR	Classless Inter Domain Routing (supernetting)
CL	Connection Less
CLNP	Connection Less Network Protocol
CLNS	Connection Less Network Service
CO	Connection Oriented
CONS	Connection Oriented Network Service
DARPA	Defense Advanced Research Projects Agency
DDN	Defense Data Network
DIB	Directory Information Base
DNS	Domain Name System / Server
DoD	Department of Defense
DV	Distance Vector
EGP	Exterior Gateway Protocol
EGW	External GateWay
ES	End System
FTP	File Transfer Protocol
GW	GateWay
ICMP	Internet Control Message Protocol
IGMP	Internet Group Management Protocol
IGP	Interior Gateway Protocol
IGW	Internal GateWay
InterNIC	Internet Network Information Center
IPnG	IP next Generation
IS	Intermediate System (gateway)
IS-IS	Intermediate System to Intermediate System
ISP	Internet Service Provider
LLC	Logical Link Control
LSA	Link State Advertisement
LSP	Link State Package
MTU	Maximum Transfer Unit
NAT	Network Address Translation
NFS	Network File System
NIC	Network Information Center
NPA	Network Point of Attachment
NS	Network Service
NSAP	Network Service / System Access Point
NSF	National Science Foundation
NSFNET	National Science Foundation NETwork
OSPF	Open Shortest Path First
PAT	Port Address Translation
PMTU	Path Maximum Transfer Unit

QoS	Quality of Service
RARP	Reverse Address Resolution Protocol
RFC	Request For Comment
RIP	Routing Information Protocol
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SPF	Shortest Path First
TL	Transport Layer
TOS	Type Of Services
TTL	Time To Live
UDP	User Datagram Protocol
vBNS	very high speed Backbone Network Service
VC	Virtual Circuit
VLSM	Variable Length Subnet Mask

### **Hoofdstuk 4: DE TRANSPORT-LAAG**

ASAP	As Soon As Possible
AWND	Allowed Window
CWND	Congestion Window
EBCDIC	Extended Binary Coded Decimal Interchange Code
IANA	Internet Assigned Addresses Authority
ISN	Initial Sequence Number
MSS	Maximum Segment Size
QoS	Quality of Service
RTT	Round Trip Time
SSTRESH	Slow Start Threshold size
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

### **Appendix A: Het OSI-Model**

NSAP	Network Service / System Access Point
PCI	Protocol Control Information
PDU	Protocol Data Unit
PSAP	Presentation Service Action Point
SAP	Service Action Point
SDU	Service Data Unit
SSAP	Session Service Action Point
TSAP	Transport Service Action Point

# DATACOMMUNICATIE

## 3 NETWERKARCHITECTUUR

### VOORWOORD

Deze cursus heeft als belangrijkste doelstelling een beeld te geven van de moderne elektronische communicatietechnieken, gefocussieerd rond computernetwerken, maar met een blik op de toekomst waar data, spraak en video tezamen over multiservice ('triple play') netwerken zullen getransporteerd worden.

We behouden voor de eerste 4 hoofdstukken de leidraad van het ISO-datacommunicatie-model. Door het explosief karakter van deze technologieën kan men dit werk niet zien als een volledig naslagwerk, maar als leidraad door de grote hoeveelheden informatie die beschikbaar worden gesteld, zowel in boekvorm, papers als soft-tutorials. Het is dus altijd mogelijk om documenten met meer diepgang te bekomen op aanvraag. We proberen een uitgebreide bibliografie samen te stellen, maar het is zeer moeilijk dit werk up to date te houden aangezien evoluties in deze wereld soms nauwelijks maanden in beslag nemen.

Als belangrijkste naslagwerken gebruiken we vooral (in volgorde van belangrijkheid) de handboeken :

1. Data Communications and Networking 5E, door Behrouz Forouzan. Uitg. McGraw-Hill. ISBN 978-0-07-131586-9. Een heel uitgebreid, modern, praktisch werk dat zowat alle aspecten van datacommunicatie voor een ingenieur behandelt.
2. Data Communications, Computer Networks and Open Systems (3E/4E) / Computer Networking And the Internet (5E), door Fred Halsall, uitg. Addison Wesley, 3<sup>e</sup> en 4<sup>e</sup> / 5<sup>e</sup> editie, ISBN 0 321 26358-8. Een volledig en zeer grondig werk dat van versie 3 en 4 met een academisch inslag evolueerde naar een zeer goede basis met aandacht ook voor de praktische realisaties in versie 5. Het is weliswaar te uitgebreid om in het beperkte tijdsbestek volledig te behandelen.
3. Data & computer communications, door William Stallings, uitg. Prentice Hall, 10<sup>e</sup> editie, ISBN 9781292014388. De 10e editie van dit boek alsook zijn voorgangers vanaf ed. 6 werden geraadpleegd. Het is een zeer praktische en volledige benadering vanuit een toch vrij elektronisch standpunt in verstaanbare taal met duidelijke figuren.
4. Computernetwerken, door Andrew S. Tanenbaum, 5<sup>e</sup> herziene uitgave, ISBN 978-90-430-2120-3. Dit werk wordt in de vele hogescholen en universiteiten als hét standaardwerk beschouwd en is een van de meest uitgebreide in zijn soort. Het heeft echter minder dan de voorgaande een elektronische inslag en de didaktische opbouw is door de verscheidenheid van de behandelde onderwerpen soms ver zoek. Dit maakt het boek zeer goed geschikt als naslagwerk, maar minder als 'tutorial'.

Deze werken zijn allen terug te vinden de mediatheek.

Hiernaast wordt er geput uit verscheidene andere boeken, papers, bedrijfscursussen postacademische- en internet-HTML cursussen om het beeldje volledig te maken. Interessante links zijn (op het moment van schrijven, google ...) o.a. :

- 'The Ethernet Evolution From 10 Meg to 10 Gig How it all Works' van UNH-IOL , univ. Van New Hampshire.  
[https://www.iol.unh.edu/sites/default/files/knowledgebase/ethernet/ethernet\\_evolution.pdf](https://www.iol.unh.edu/sites/default/files/knowledgebase/ethernet/ethernet_evolution.pdf)
- The TCP/IP guide van Charles M. Kozierok. Een vrije online gids van ong. 2000 blzn.  
<http://www.tcpipguide.com/free/>
- TCP/IP Tutorial en technical overview. Een IBM redbook.  
<https://www.redbooks.ibm.com/redbooks/pdfs/gg243376.pdf>

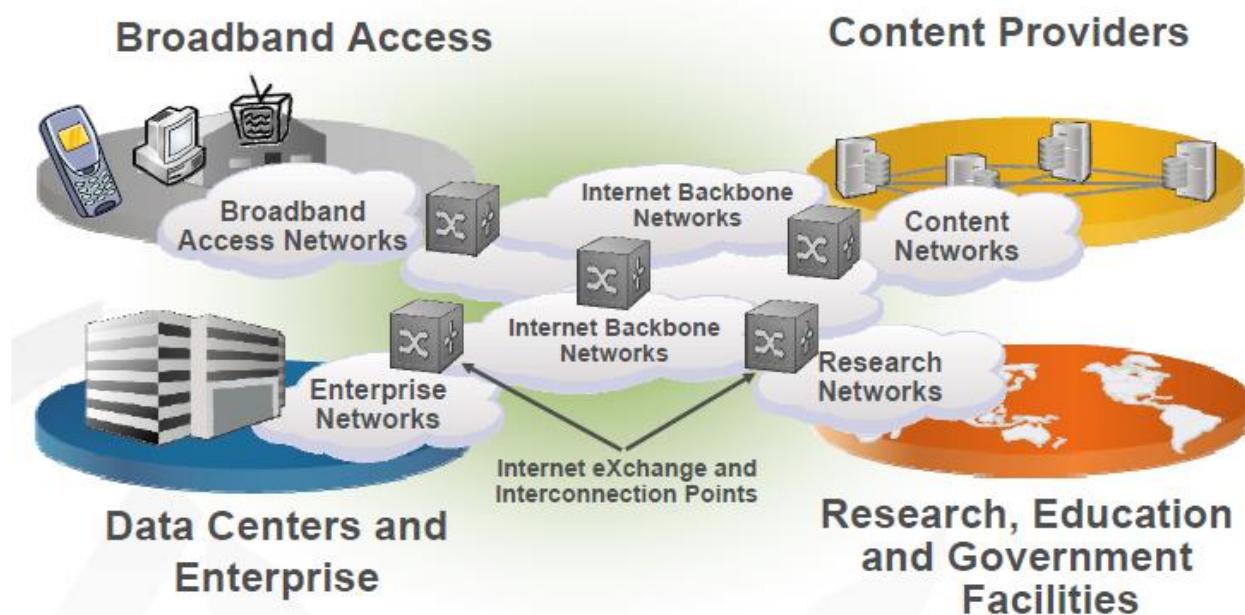
...

# I. DATACOMMUNICATIE

## INLEIDING

Waar u als lezer waarschijnlijk niet meer van overtuigd dient te worden is dat, net zoals mensen, ook computers de noodzaak kennen om met elkaar te communiceren. Als we dan de definitie van computers in de toekomst extrapoleren naar TV, radio, camera, auto, ...tot en met lichtschakelaars, zien we welk een belangrijke rol datacommunicatie zal innemen.

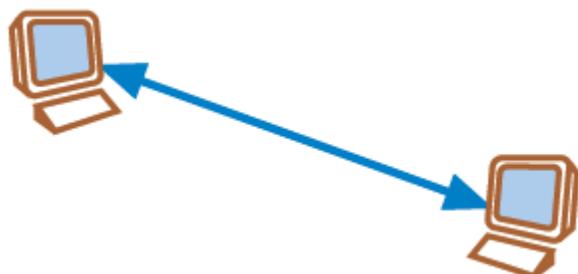
Anno 2013 ziet dit er in grote lijnen uit zoals in Figuur 1. We merken hier meteen dat dit zich niet zal beperken tot computercommunicatie maar dat ook telefonie en video zich hierin een plaats hebben verworven.



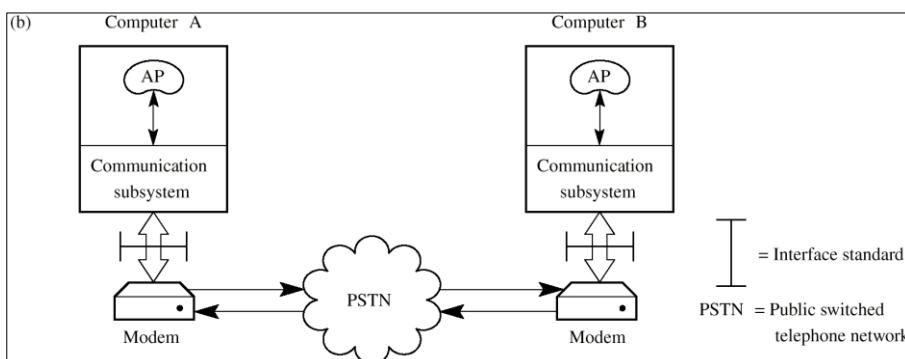
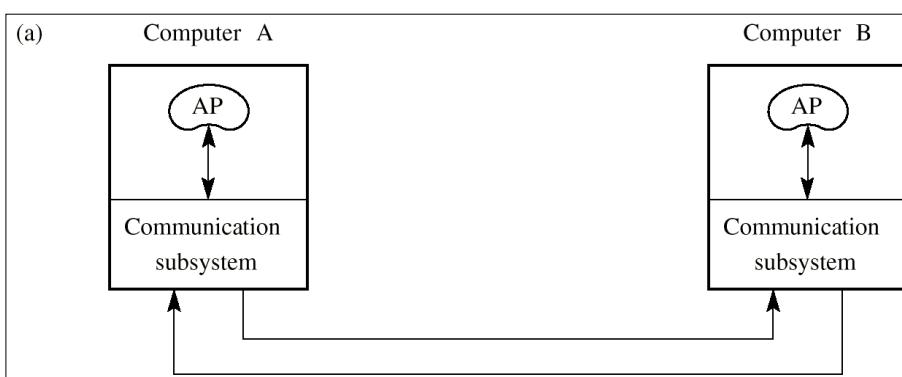
Figuur 1 Communicatie entiteiten

In een dergelijk complex netwerk wordt het probleem opgesplitst in hanteerbare stukken, lagen genoemd, en wordt het volledige communicatiemodel beschouwd als een opeenstapeling van de lagen, zie ook verder.

## COMMUNICATIE NETWERKEN.

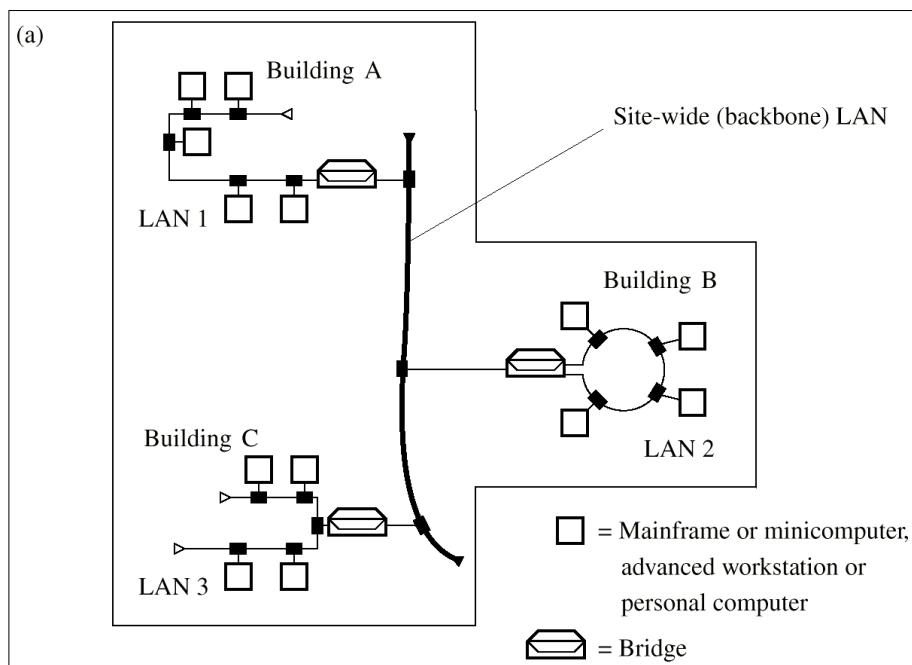


2 computers kunnen rechtstreeks met elkaar worden verbonden als ze in dezelfde kamer staan. Dit noemt men een eenvoudige 'point to point link', P2P.



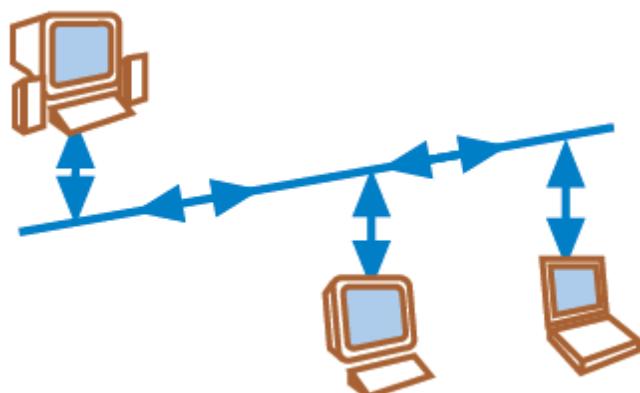
Zijn ze geografisch op een afstand gelegen dan moet deze 'point to point link' uitgebreid worden met een **modem** om de data te transporteren op het telefonienet.

Figuur 2 'point to point' computerverbindingen a) rechtstreeks, b) via een modem



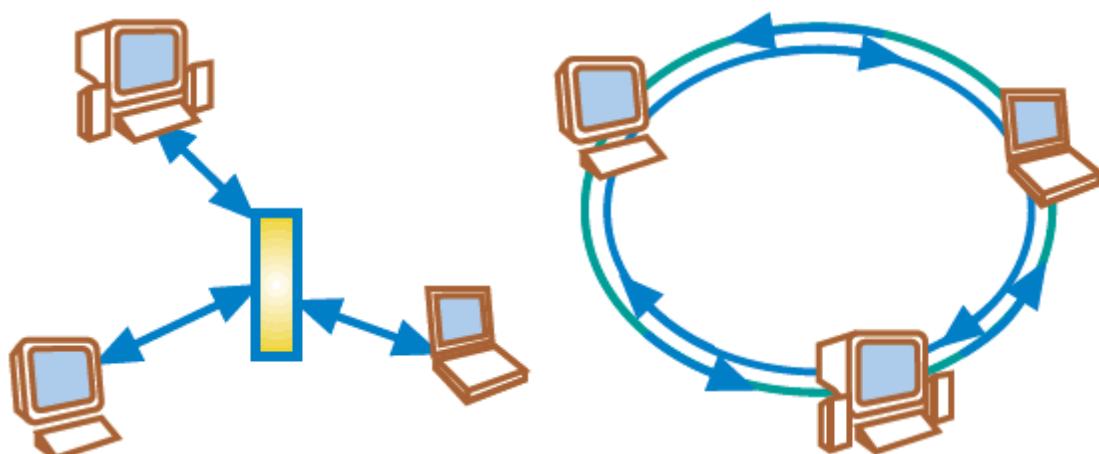
Als er echter meer dan 2 computers aan te pas komen, dan moet het communicatienetwerk een ‘schakelfunctie’ bevatten om toe te laten dat de computers met elk ander station kunnen communiceren.

Figuur 3 LAN verbindingen van computers.

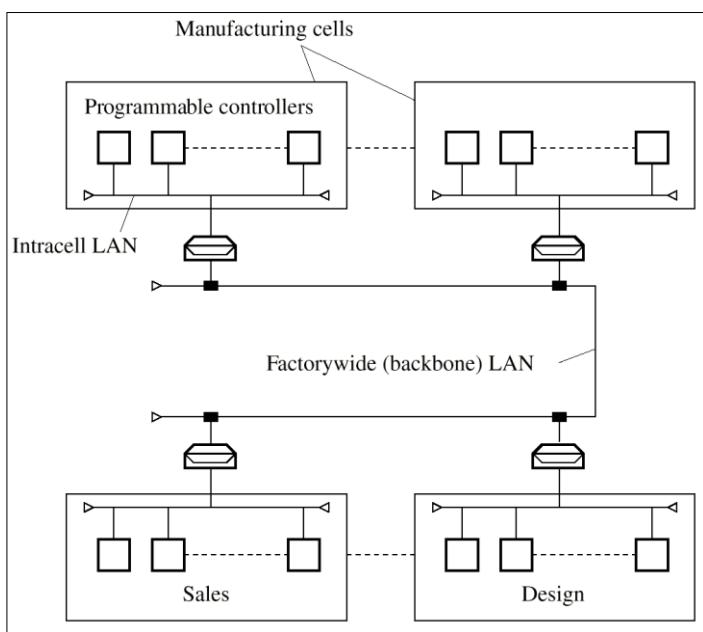


Dit schakelnetwerk kan verschillende topologieën aannemen zoals een **bus** (bv. shared ethernet, CSMA/CD), ...

een **ster** (bv. switched ethernet FD) of een **ring** (bv. Token ring).



Figuur 4 LAN topologien : bus, ster en ring.

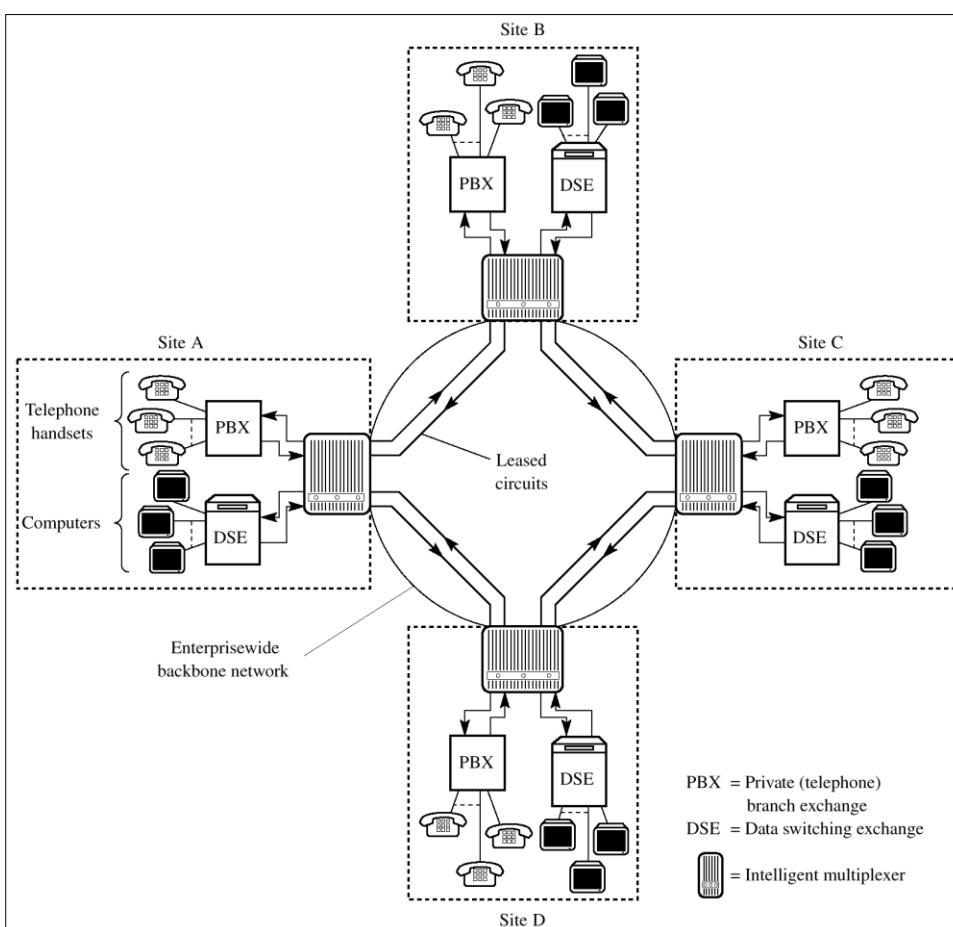


In automatiseringsomgevingen wordt het zelfs meer en meer gebruikelijk ettelijke hiërarchische niveaus te creëren gaande van **sensor/actuator-bussen** die de transducers koppelen met een meetcontroller/PLC, die op zijn beurt communiceert met de procescomputer over de **veldbus**, die vervolgens verbonden is met het **bedrijfsnetwerk**, bv. een ethernet LAN.

Figuur 5 Een bedrijfsnetwerk van automatiseringscomputers.

Voor de volledigheid vermelden we hieronder nog een aantal oudere oplossingen die mogelijk sporadisch nog in functie zijn maar die tegen snel tempo worden uitgefaseerd. Ze dienen in dit werk vooral om de evolutie van netwerken aan te tonen en hierdoor de huidige systemen beter te begrijpen.

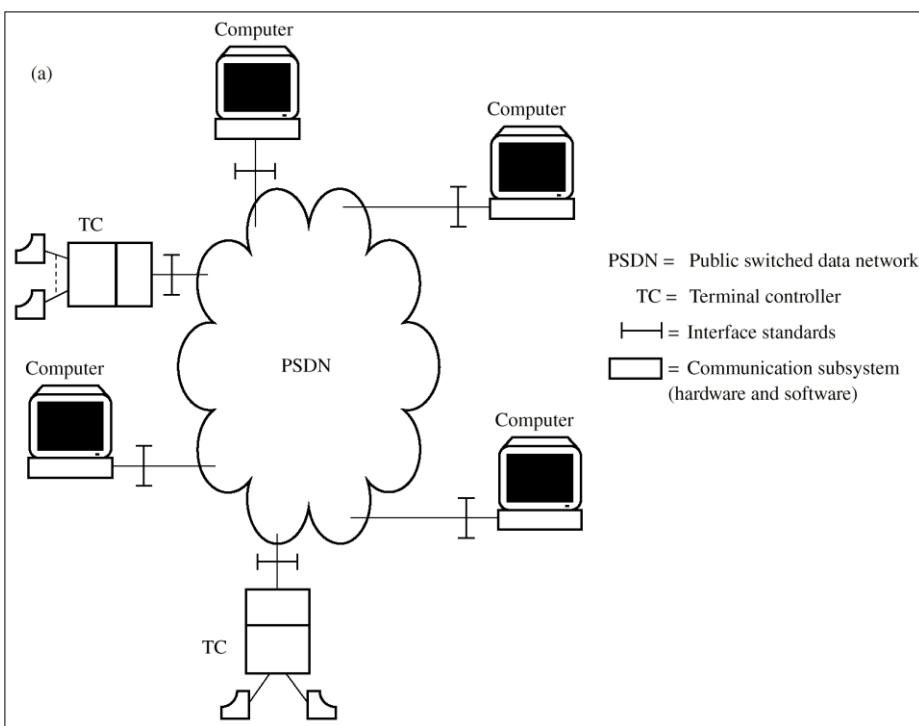
Staan de computers te ver van elkaar opgesteld dan wordt (werd) gebruik gemaakt van publieke dragers.



Men spreekt dan van een WAN. Deze kan een aantal vormen aannemen. Als alle computers behoren tot hetzelfde bedrijf, en er is voldoende trafiek dan worden er lijnen gehuurd met aan beide zijden de gepaste schakel-elementen die zowel **data** als **spraak** overbrengen.

Dit wordt ook wel eens een privaat netwerk genoemd.

Figuur 6 Een privaat netwerk.

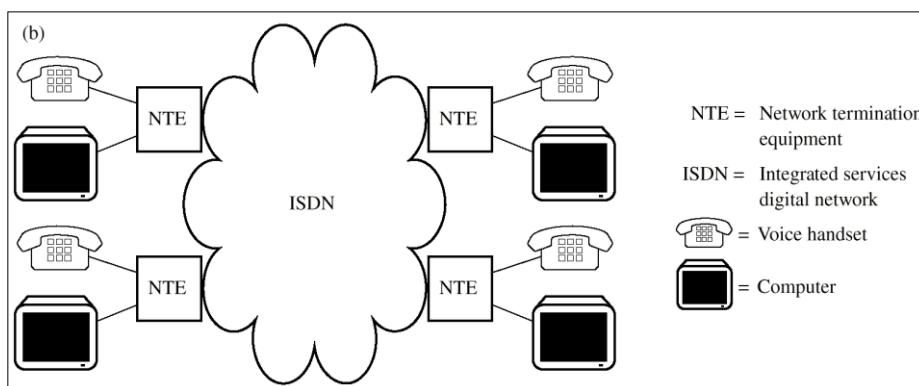


Landelijke dataverbindingen zoals bankterminals worden dan weer rechtstreeks aangesloten op publieke **datanetwerken**, PSDN, ‘public switched datanetworks’.

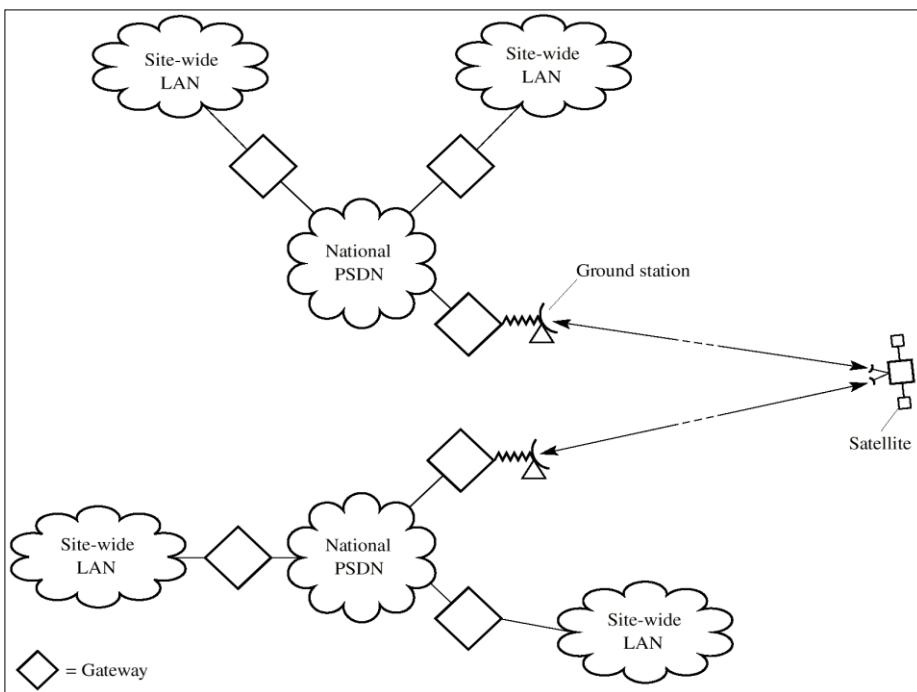
Deze netwerken zijn ontworpen voor **data**-transport en bestaan **naast** het **telefonie**-netwerk.

Figuur 7 Publieke datanetwerken : a) PSDN

Deze duale toestand wordt dan weer opgelost door de telefonie-netwerken toegankelijker te maken voor data → **ISDN** (Integrated Services Digital Networks).

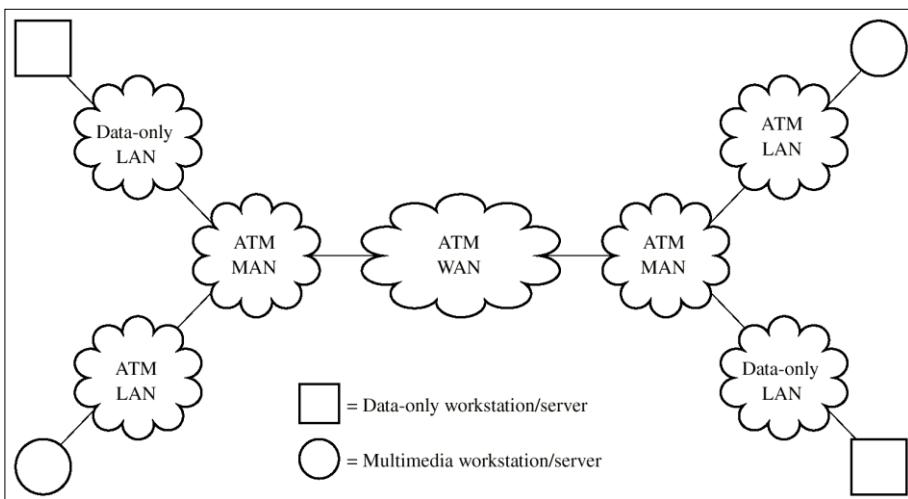


Figuur 8 Publieke datanetwerken : b) ISDN.



In de voorgaande configuraties zijn we er van uit gegaan dat alle computers behoren tot hetzelfde netwerk. Maar **wereldwijd** worden nu ook computers uit verschillende LAN's met elkaar verbonden door bvb. een PSDN. Men spreekt van 'internetworking' of **internet**.

Figuur 9 Een wereldwijd internetwerk.



Computers evolueerden tot werkstations die **verschillende informatie-types** ondersteunen zoals videotelefonie, videoconferenties of algemener multimedia.

Figuur 10 Een breedband ISDN netwerk.

De transmissie-eisen voor dergelijke verbindingen zijn zo hoog dat ze '**breedband ISDN**' (multiservice) netwerken worden genoemd. Aangezien de verschillende informatietypes, verschillende karakteristieke eisen aan het transport opleggen is er een nieuwe benadering ontworpen om hieraan te voldoen : **ATM**, Asynchrone Transfer Mode.

(MAN : Metropolitan Area Network, meestal een dubbele glasvezelring rondom een grote stad.)

# STANDAARDEN

In vroegere tijden beschermden computerfabrikanten zichzelf door hun eigen (= **proprietary**) standaarden te definiëren voor communicatie tussen hun systemen → ‘closed systems’.

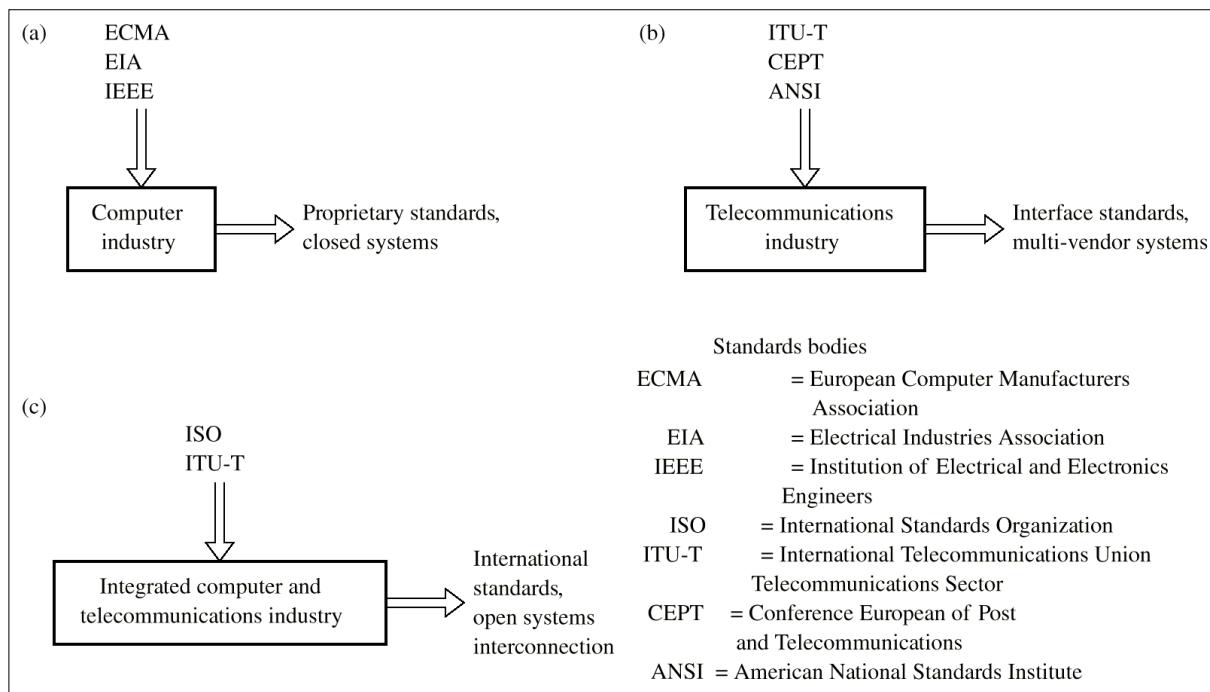
Als reactie daarop zijn verschillende internationale instanties standaarden gaan definiëren die nu algemeen aanvaard worden. Men spreekt bv. in de telefoniewereld van **V-, X- of I-series recommendations**.

- **V-series** : als standaard tussen DTE's en modems (→ PSTN, publiek **telefonienetwerk**):  
vb. V24 (ITU-T) = RS232D (EIA).
- **X-series** : tussen DTE's en PSDN's (public switched **data networks**) :  
bv. X25 (of tussen DTE en DCE : X21=RS232)
- **I-series** : tussen een DTE en ISDN (**integrated services digital network**, = **telefonie + data**) bv. I-430, I-440, I-450,... Zie ISDN.

Door standaardisatie is de gebruiker niet meer gebonden aan 1 fabrikant. Een en ander evolueerde tot ‘**open systems**’ met hieronder in Figuur 11 een diagram van de werkterreinen van de verschillende standaardiseringinstituten. In grote lijnen: deel a) zijn standaarden die voortspruiten uit de computer industrie, deel b) uit de telefonie wereld.

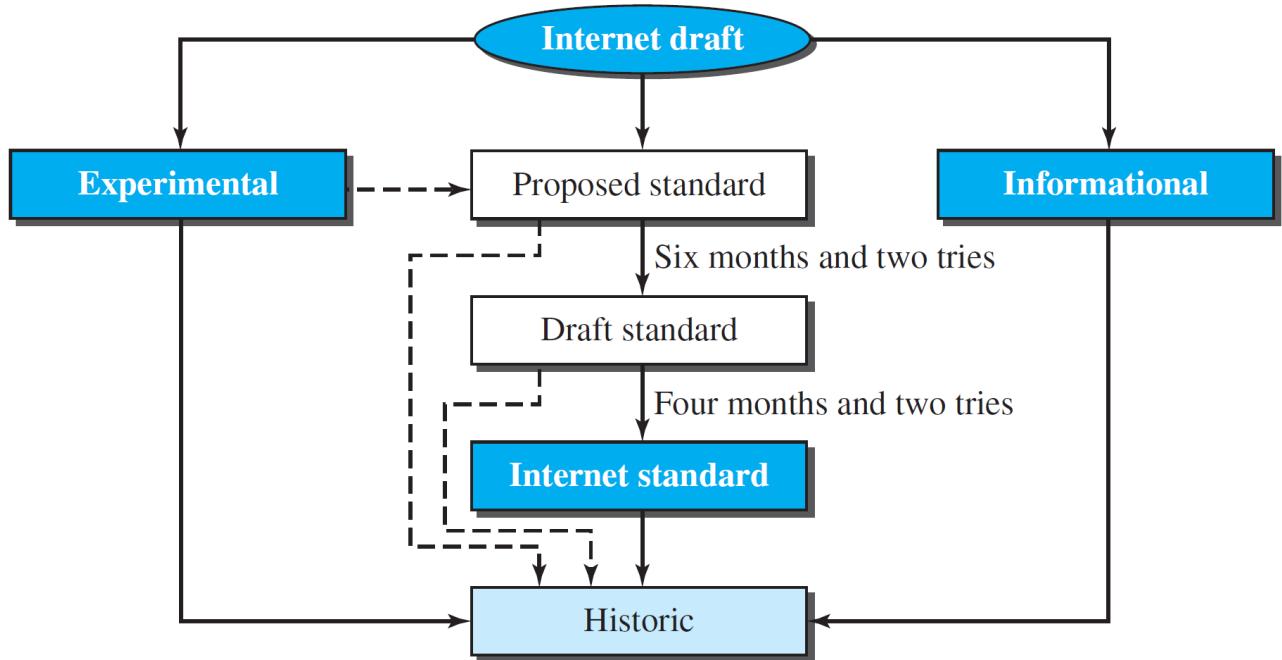
Deel c) hiervan duidt op het ontstaan, halfweg de jaren '70, van het **ISO** (International Standards Organisation) referentiemodel voor ‘**Open Systems Interconnection**’, **OSI**.

Het beschrijft de structuur van het volledige communicatiesubsysteem in elke computer.



**Figuur 11 Standaards en hun instellingen.**

Wat hier nu ontbreekt is het instituut dat alle internet standaarden regelt: IETF.



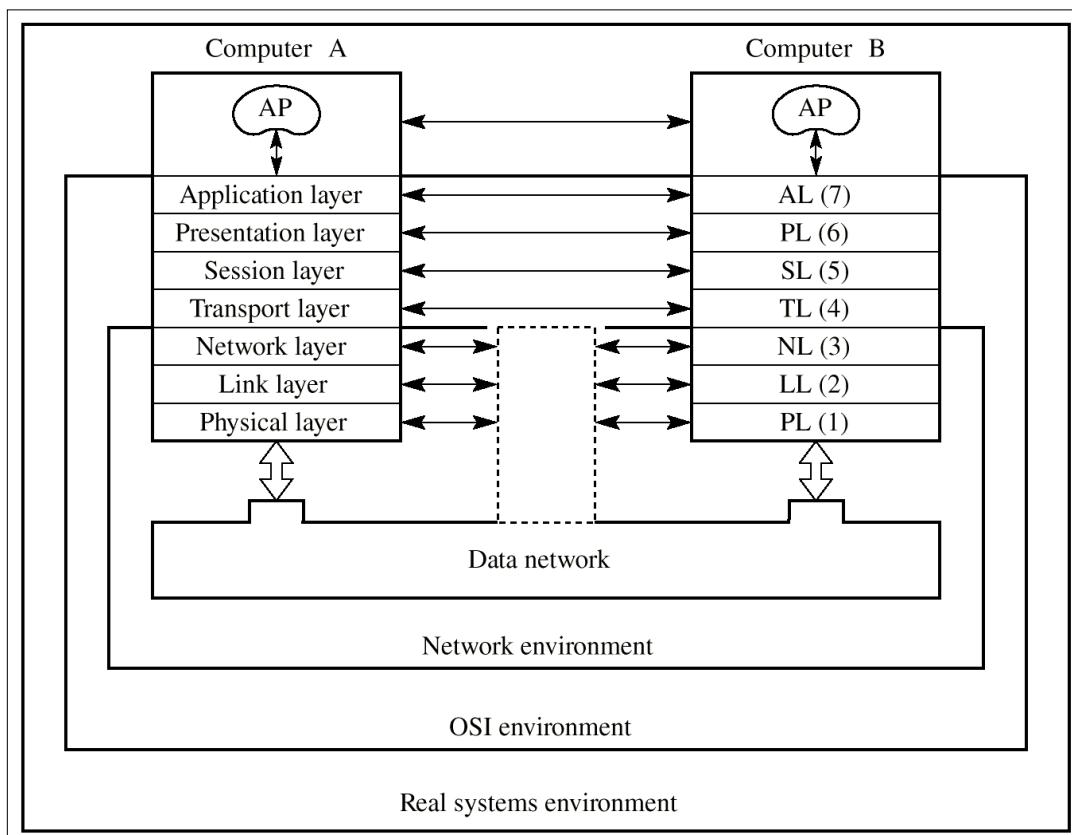
Figuur 12 De evolutie van een internet standaard.

Een internet standaard ondergaat een evolutie van 'Internet draft', wat eigenlijk een werkdocument is dat 6 maand 'leeft', naar standaard. Een draft kan bij goedkeuring door de Internet autoriteiten een RFC (Request For comments) publicatie krijgen met een nummer. Het wordt dan nauwkeurig onder de loep genomen door alle belanghebbende partijen.

Een RFC kan dus 6 'maturiteits-levels' krijgen, in experimentele toestand zal het nooit effectief toegepast worden op het Internet. Informational is zoals het woord zegt informatief.

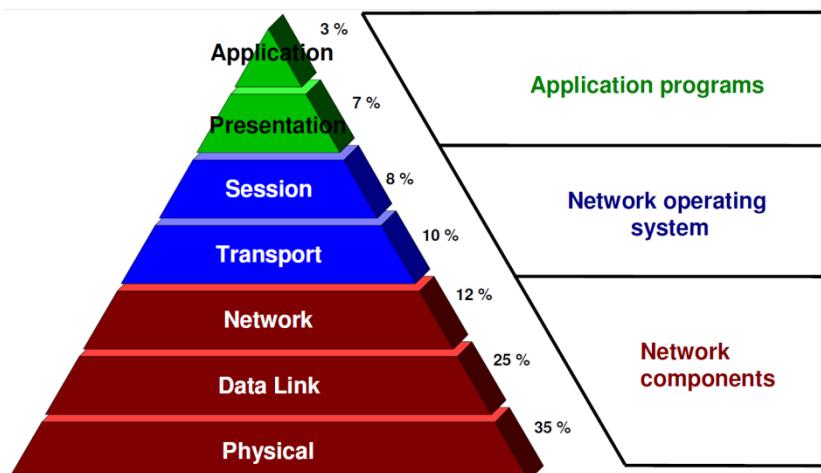
## HET OSI MODEL VAN ISO

Een communicatie-subsysteem is een complex geheel van hard- en software. Dit laatste werd in vroegere dagen meestal geschreven als 1 'klomp' assembler wat aanpassingen en foutlokalisatie er niet gemakkelijker op maakte. Daarom heeft ISO het gehele probleem trachten op te delen in 7 lagen, elk met hun eigen specifieke functies. De 7 lagen zijn bovendien in te delen in 2 groepen : netwerk- en application-georiënteerd (→ OSI environment), zie Figuur 13.



Figuur 13 Structuur van het ISO model

De 3 onderste lagen (1-3), de netwerk-lagen, verzorgen de **datacommunicatie** tussen 2 computers. De 2 tussenliggende lagen (4-5) zijn een onderdeel van het operating systeem dat zich concentreert op de netwerk-communicatie.



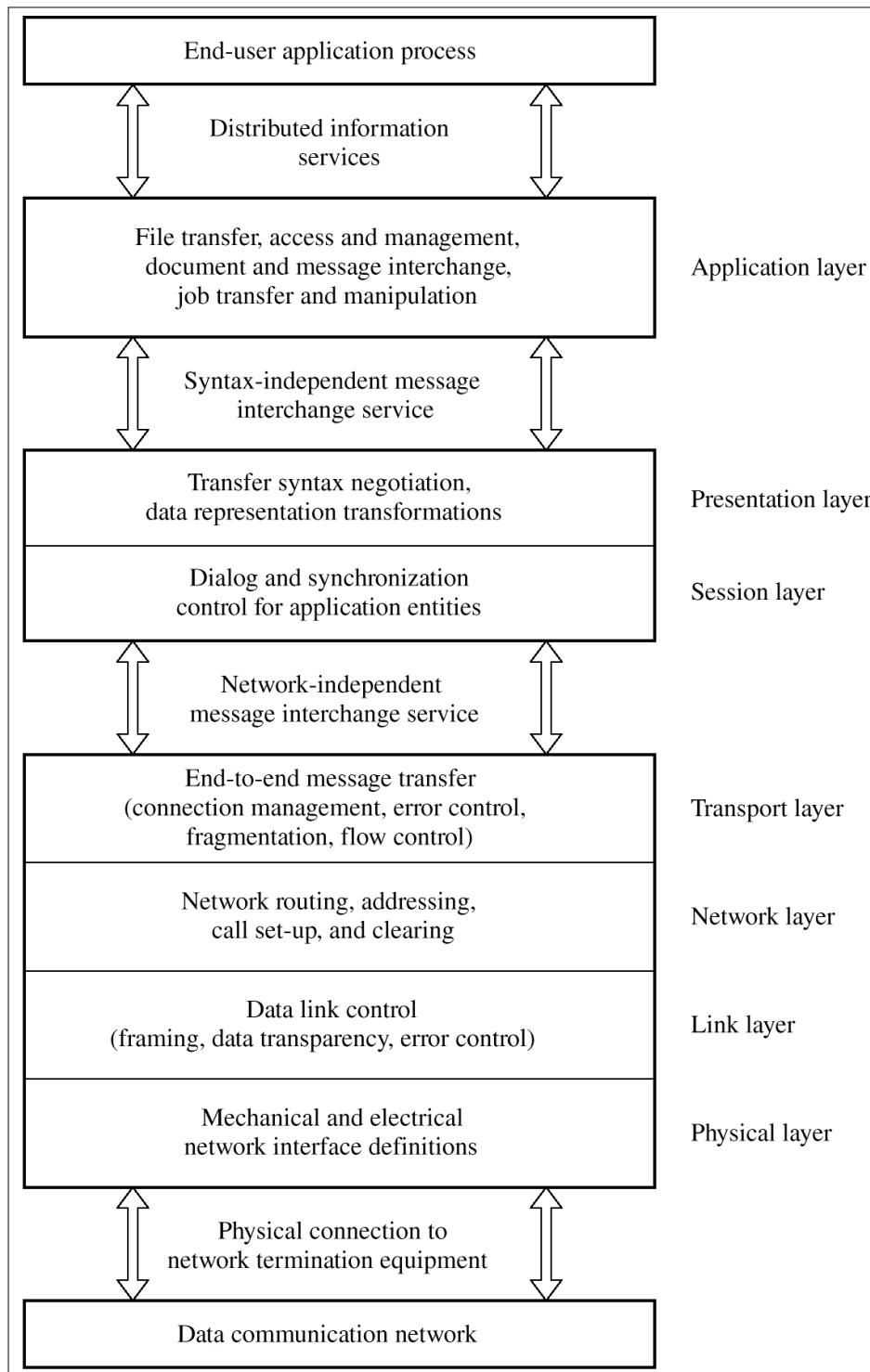
De 2 bovenste (6-7) zijn applicatiegericht en beheren de protocols om 2 eindgebruiker-toepassingen met elkaar te laten communiceren via een set van services aangeboden door het operating system.

Figuur 14 Indeling van het ISO model

De belangrijkste functies van de lagen worden uiteengezet in onderstaande figuur.

Elke laag werkt volgens een welbepaald protocol om boodschappen door te geven met een corresponderende laag in een ‘afgelegen’ systeem, **peer to peer** communicatie, en heeft een goed gedefinieerde **interface** naar de eigen onder- en bovenliggende lagen → services.

Later in dit werk komen we uitgebreider terug op deze functies en alternatieve modellen. Voorlopig dient dit alleen als situatieschets voor het verdere betoog van deze cursus.



Figuur 15 Functieoverzicht van de ISO layers

# 1 DE FYSIEKE LAAG

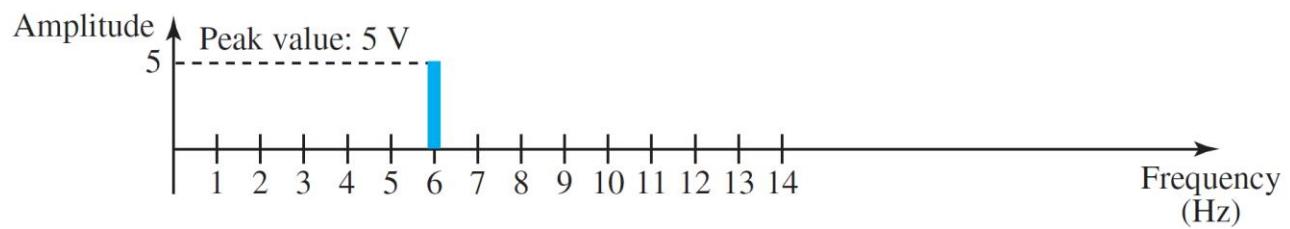
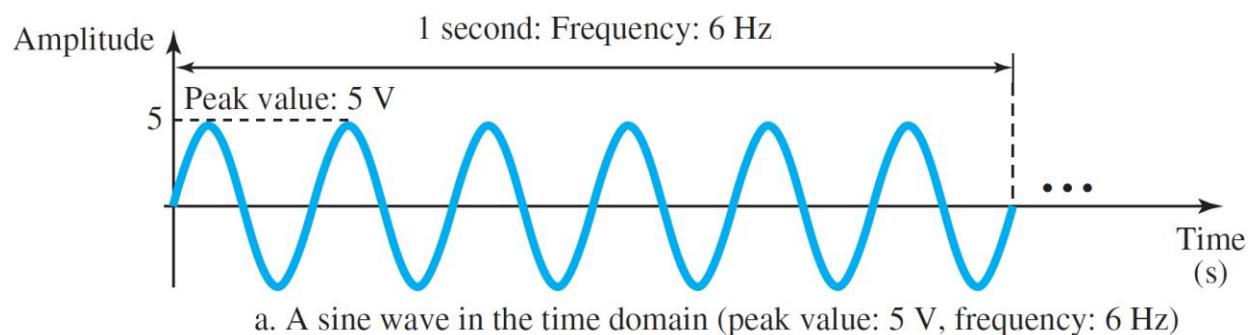
## INLEIDING

In dit hoofdstuk bespreken we de volledige problematiek zoals afgebakend in de **onderste laag** van het **OSI-model**. De theoretische basis voor dit hoofdstuk is/wordt gelegd in de cursus digitale techniek, vooral pt. 1.1. We geven hier alleen de belangrijkste stellingen om ons betoog op te kunnen bouwen.

### 1.1 SIGNALEN EN THEORETISCHE CONCEPTEN.

#### 1.1.1 TIJD- EN FREKWENTIEDOMEIN

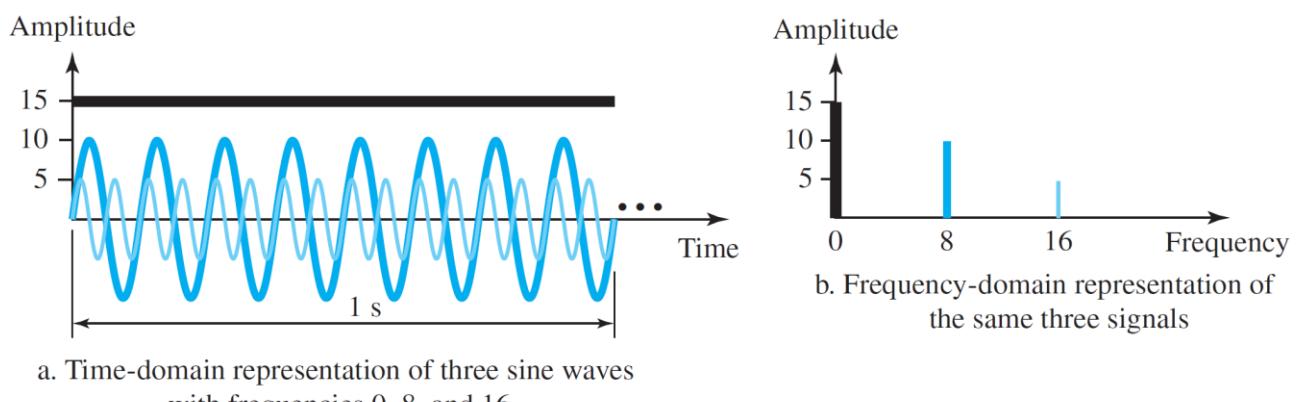
Een sinus is een typisch signal dat we (tot nu toe) in het tijdsdomein hebben voorgesteld:



b. The same sine wave in the frequency domain (peak value: 5 V, frequency: 6 Hz)

Figuur 16 Tijd- en frekwentiedomein voorstelling van een sinusgolf.

Een frekwentiedomein plot is typisch niet geïnteresseerd in de fase, enkel amplitude en frekventie. Dit is vooral interessanter als we werken met een (golf die bestaat uit een) combinatie van sinussen.



Figuur 17 Tijd- en frekwentiedomein voorstelling van drie sinusgolven.

Een DC signaal heeft enkel een amplitude, frekventie = 0.

In datacommunicatie kan je met een enkelvoudige sinus geen informatie overbrengen. De signalen bestaan steeds uit een compositie van sinussen. De Franse wiskundige Jean Baptiste Fourier bewees dat elk samengesteld (composiet) signaal bestaat uit een combinatie van enkelvoudige sinussen met verschillende frekventies, amplitudes en fasen.

Dit samengesteld signaal kan op zichzelf periodiek zijn (= er zit een steeds wederkerend patroon in) of niet-periodiek. Een dergelijk **periodiek** signaal zullen we kunnen voorstellen in het frekwentiedomein door een reeks **discrete** sinussen.

## 1.1.2 FREKWENTIESPECTRUM EN BANDBREEDTE

### FOURIER

Elk repetitief (= periodiek) signaal  $s(t)$  kan voorgesteld worden door de som van een aantal sinussen en cosinussen, met verschillende amplitudes. Deze sinussen worden gekenmerkt door hun frekwentie en amplitude waarbij we ze voorstellen in het frekwentiedomein. We bekomen dit na omzetting van het signaal in het tijdsdomein door fourier transformatie.

$$s(t) = A_0 + \sum_{n=1}^{\infty} A_n \sin(2\pi nft) + \sum_{n=1}^{\infty} B_n \cos(2\pi nft)$$

Met als coefficienten:

$s(t)$  is periodiek dus heeft een periode:  $T = 1/f$ .

$A_0$ : Als je het signaal integreert naar  $t$  gedurende een periode, en het geen

'DC'- component heeft, is dus  $A_0 = 0$ . ( $\rightarrow A_0$  = de DC- component).

$$A_0 = \frac{1}{T} \int_0^T s(t) dt \quad A_n = \frac{2}{T} \int_0^T s(t) \cos(2\pi nft) dt$$

$$B_n = \frac{2}{T} \int_0^T s(t) \sin(2\pi nft) dt$$

De andere componenten  $A_n$  en  $B_n$  vind je door het signaal te vermenigvuldigen met een (co)sinus met een frekwentie  $f$  gelijk aan die van oorspronkelijk signaal  $s(t)$ , bv.  $A_1$  voor  $n = 1$ , of met frekwenties die een geheel veelvoud zijn ( $n = 1, 2, 3, \dots, \infty$ ) van de 'grondfrekwentie'  $f$  ( $\rightarrow$  discrete sinussen).

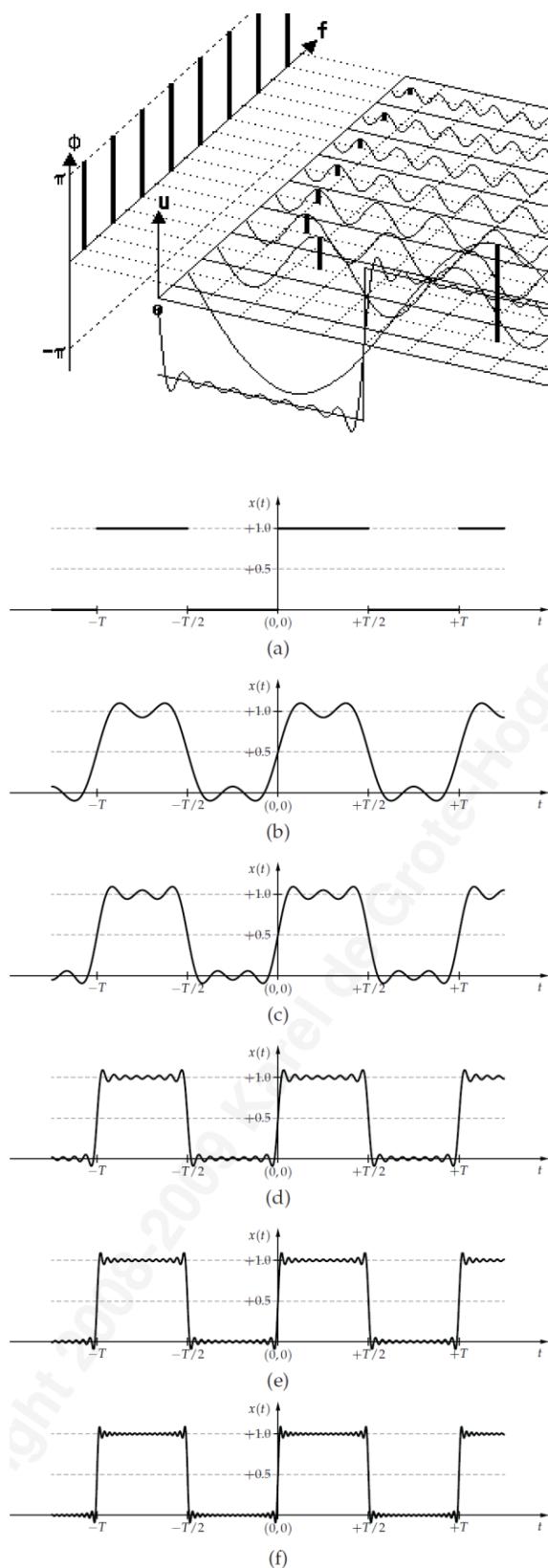
Naast het klassieke wikipedia vind je van fourier een heel (heerlijke/) leerrijke applet onder :

<http://www.falstad.com/fourier/>

Je kan verschillende golven (tijdsdomein) voorstellen door een aantal discrete sinussen en/of cosinussen die telkens frekwenties bevatten die een geheel veelvoud zijn van de grondfrekwentie van het oorspronkelijk signaal. We noemen dit de 'harmonischen'. Als je  $\infty$  veel harmonischen neemt wordt dit uiteindelijk de perfecte voorstelling van het oorspronkelijk signaal.

Door Fourier ANALYSE kan je leren hoe bv. een blokgolf (maar ook andere golven) kunnen worden benaderd / vervangen worden door een resp. eindig / oneindig aantal van zijn harmonischen. Tracht in bovenstaande applet steeds meer harmonischen te selecteren door de schuifbalk "Number of Terms".

We zien op identieke wijze in onderstaande figuur (uit de cursus DSP van W. Daems) dat naarmate men meer harmonischen gebruikt, men meer en meer de oorspronkelijke golfvorm zal benaderen.



Figuur 18 Fourier analyse van een blokgolf.

De blokgolf wordt door Fourier ontleed in:

- De DC component  $A_0 = 0,5 +$
- (ideaal) oneindig veel harmonischen volgens de formule :

$$x(t) = \frac{4}{\pi} \sum_{k=1,3,5,\dots}^{\infty} \frac{1}{k} \sin k \cdot \omega t$$

$$= \frac{4}{\pi} \left( \sin \omega t + \frac{1}{3} \sin 3\omega t + \frac{1}{5} \sin 5\omega t + \frac{1}{7} \sin 7\omega t + \dots \right)$$

Geen cosinussen:  $\forall A_n = 0$

We zien de oneven harmonischen terugkomen met steeds lagere coëfficiënten. Naaststaande figuur zal in beeld b)  $k=1$  en  $3$  voorstellen, beeld c)  $k \leq 5$ , d)  $k \leq 15$ , e)  $k \leq 25$ , f)  $k \leq 35$ ,  
Naarmate meer harmonischen worden opgenomen zullen we de blokgolf meer en meer benaderen.

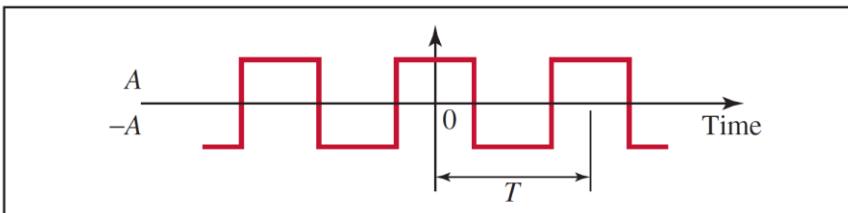
Wat we uit naaststaande figuren ook kunnen afleiden is dat de **steile** flanken van de blokgolf vooral gevormd worden door de hogere harmonischen, de platte (... DC) waarden door de lagere.

Deze steile flanken zullen na een transmissiemedium met beperkte bandbreedte, zie verder, niet meer zichtbaar zijn bij de ontvanger.

Als we bv. deze blokgolf anders plaatsen in het tijdsdomein zoals in onderstaande Figuur 19 Fourier analyse van een 2e blokgolf., dan hebben we :

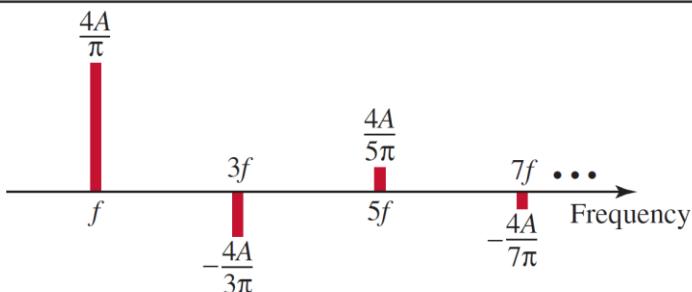
- **Geen DC component:  $A_0 = 0$**  het signal is verschoven tov de Y-as...+
  - Allemaal cosinussen, ipv. sinussen : het signal is verschoven tov de X-as...
- $\forall A_n = 0 !$ , enkel  $B_n$

Time domain



$$A_0 = 0 \quad A_n = \begin{cases} \frac{4A}{n\pi} & \text{for } n = 1, 5, 9, \dots \\ -\frac{4A}{n\pi} & \text{for } n = 3, 7, 11, \dots \end{cases} \quad B_n = 0$$

$$s(t) = \frac{4A}{\pi} \cos(2\pi ft) - \frac{4A}{3\pi} \cos(2\pi 3ft) + \frac{4A}{5\pi} \cos(2\pi 5ft) - \frac{4A}{7\pi} \cos(2\pi 7ft) + \dots$$

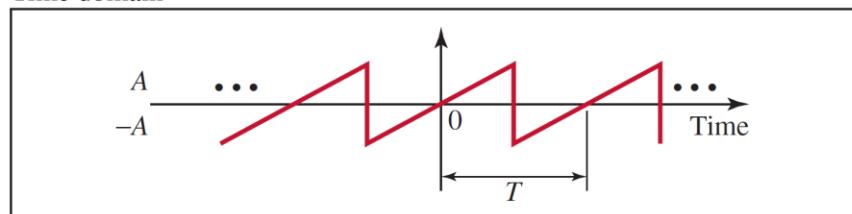


Frequency domain

Figuur 19 Fourier analyse van een 2e blokgolf.

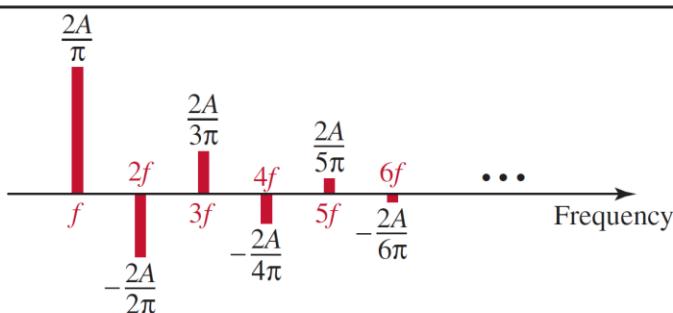
Figuur 20 Fourier analyse van een zaagtand.

Time domain



$$A_0 = 0 \quad A_n = 0 \quad B_n = \begin{cases} \frac{2A}{n\pi} & \text{for } n \text{ odd} \\ -\frac{2A}{n\pi} & \text{for } n \text{ even} \end{cases}$$

$$s(t) = \frac{2A}{\pi} \sin(2\pi ft) - \frac{2A}{2\pi} \sin(2\pi 2ft) + \frac{2A}{3\pi} \sin(2\pi 3ft) - \frac{2A}{4\pi} \sin(2\pi 4ft) + \dots$$



Frequency domain

Figuur 20 Fourier analyse van een zaagtand.

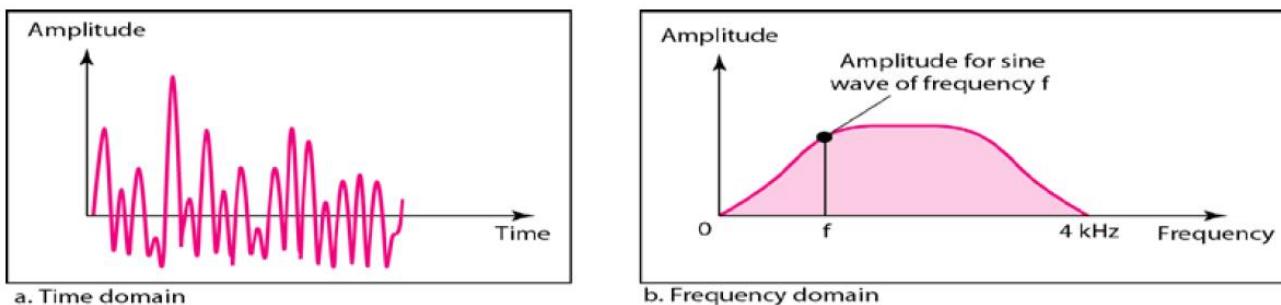
Een andere site die deze componenten makkelijk (visueel) laat berekenen is :

<http://www.fourier-series.com/f-transform/index.html>

Kies 'FOURIER SERIES', met daaronder 'flash program'. Creeer een functie met de slider en 'manually calculate' de coëfficiënten. Je ziet visueel de integraal in werking: oppervlakte berekening.

Algemeen kan men stellen dat **periodieke** signalen (tijdsdomein) kunnen worden samengesteld door een aantal **discrete** frekwenties in het frekventiedomein. Zie evt ook Figuur 22.

Niet-periodieke signalen kunnen dan weer voorgesteld worden door een **continue** groep van sinussen in het frekventiedomein. Hieronder zie je bv. een audiosignaal.



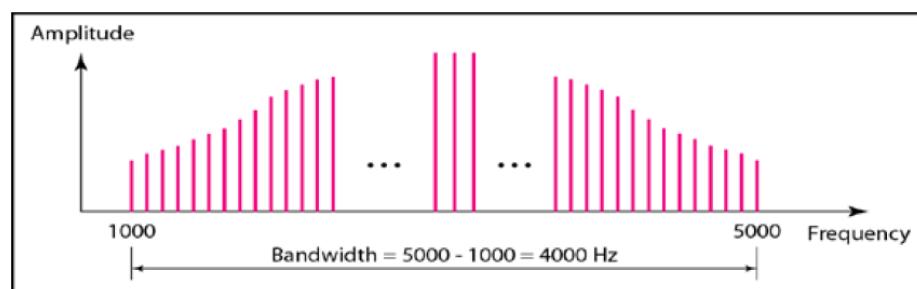
Figuur 21 Fourier analyse van een niet periodiek signaal.

## BANDBREEDTE

De **bandbreedte** van dit signaal wordt gegeven door :

$$B = f_{\max} - f_{\min}$$

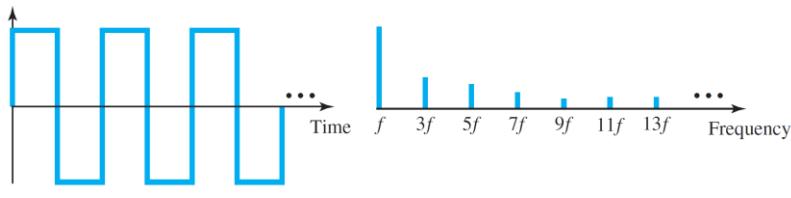
Hier is deze bandbreedte (voor bv. een **telefonie** gesprek) typisch 4KHz. Deze bandbreedte definitie wordt evenzeer gebruikt indien het gaat over periodieke signalen, dus met discrete frekventiecomponenten.



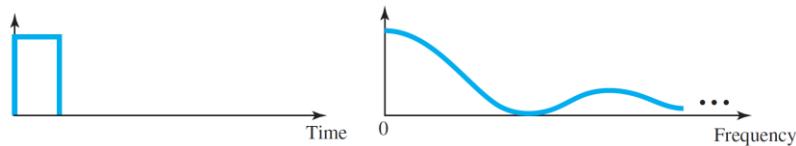
Figuur 22 Bandbreedte van een periodiek signaal.

We zullen verder ook zien dat we de bandbreedte voor transmissie moeten beperkt houden wegens de beperkte doorlaatkarakteristiek van een medium. Zo zal een digitaal signaal, doorgestuurd op een kanaal, steeds vervormd bij de ontvanger aankomen door de beperkte doorlaatkarakteristiek van het kanaal.

Bv. Digitale signalen met onbeperkte bandbreedte.



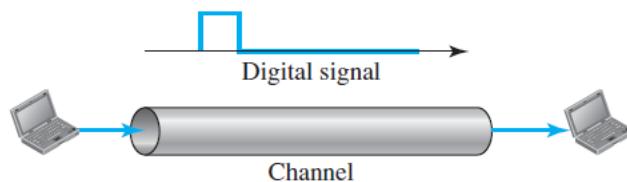
a. Time and frequency domains of periodic digital signal



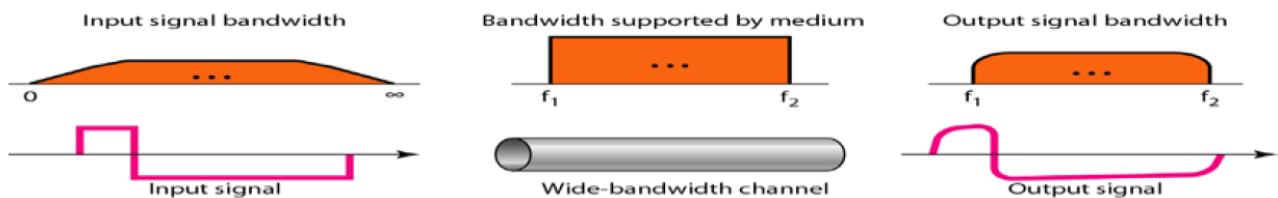
b. Time and frequency domains of nonperiodic digital signal

Figuur 23 Tijd- en frekventiedomein voorstellingen van een digitaal signaal.

Als we deze signalen nu door een kanaal sturen worden de frekventiediagrammen met mekaar 'vermenigvuldigd'.



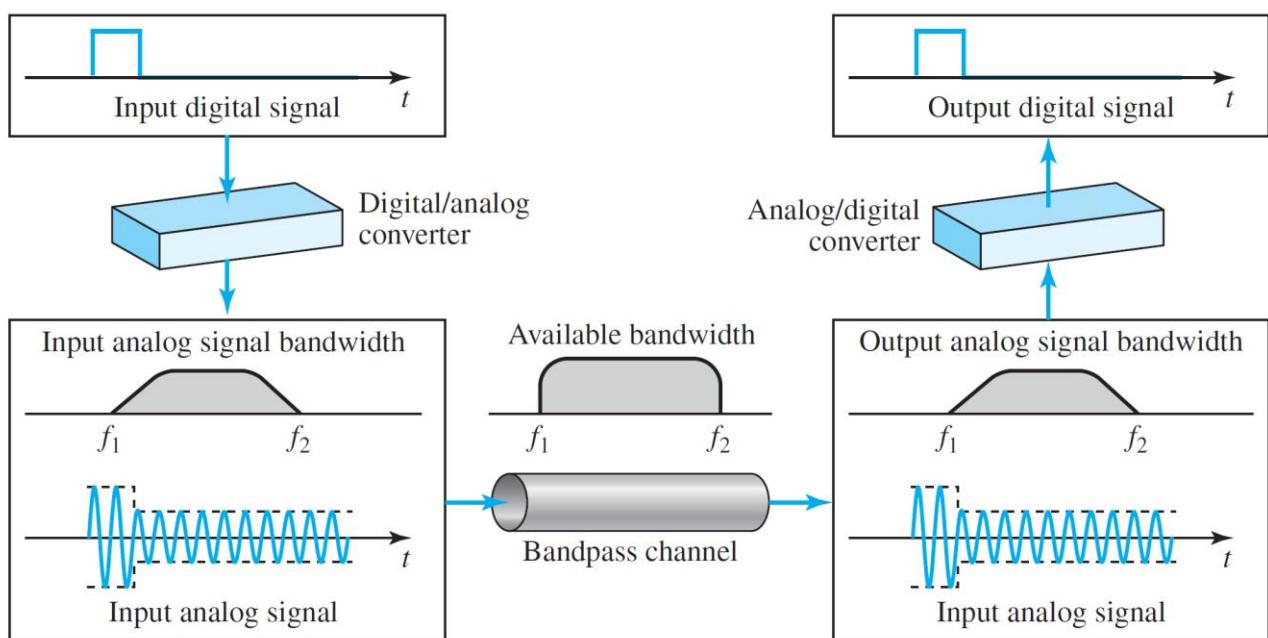
Figuur 24 Transmissie van een digitaal signaal.



Figuur 25 signaalvervorming door de beperkte bandbreedte van een kanaal.

Niet alle (hoge) frekventies zullen aan de uitgang van het kanaal verschijnen, met vervorming tot gevolg.

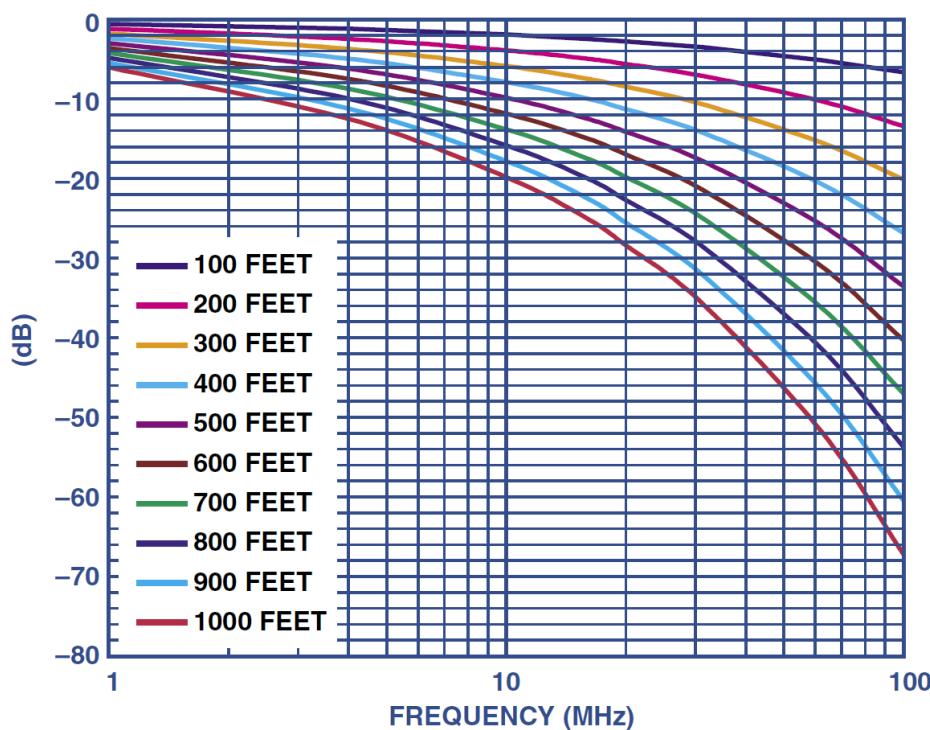
Als we dit signaal echter zouden moduleren, bv. door AM (Amplitude Modulatie), met een frekventie midden in de bandbreedte van het kanaal is het veel waarschijnlijker dat we een signaal ontvangen waar we wel duidelijker de digitale informatie kunnen uithalen... dus met minder kans op transmissiefouten.



Figuur 26 ‘Aangepaste’ of analoge transmissie van een digitaal signaal.

Praktisch voorbeeld van een kanaal-bandbreedte:

Een veelgebruikte bekabeling voor computernetwerken is UTP, CAT5e kabel.



Figuur 27 beperkte bandbreedte van een UTP CAT5 kabel.

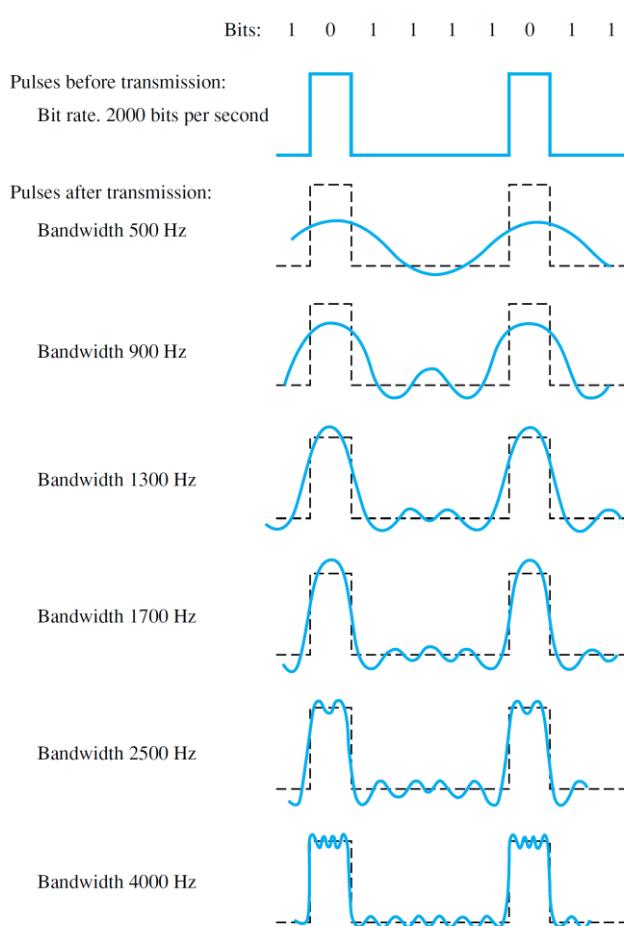
We zien dat voor een typische maximale lengte van 100m (=320ft) er voor hoge frekenties, 100MHz, een verzwakking optreedt van -20dB, wat overeenkomt met een ‘uitgangssignaal’ met een verzwakking A van:

$$A = 20 \log (V_2/V_1) \rightarrow A = -20\text{dB} \rightarrow V_2 = V_1/10 \text{ of}$$

$$A = 10 \log (P_2/P_1) \rightarrow A = -20\text{dB} \rightarrow P_2 = P_1/100 !!$$

= 1/100e van het vermogen!

We kunnen dus begrijpen dat als we een blokgolf sturen door een transmissiemedium met een eindige, beperkte bandbreedte (wat ze allemaal hebben, zelfs fibers), deze blokgolf "beschadigd" bij de ontvanger zal aankomen. Het is kunst om deze ontvangers ZO te ontwerpen dat ze zelfs bij gebrek aan hogere harmonischen de originele bitstroom toch nog kunnen recupereren.



Hoe hoger de bandbreedte van het medium, hoe beter het signaal dat bij de ontvanger aankomt, en hoe makkelijker het wordt om de bitstroom te reconstrueren zonder fouten.

Figuur 28 Effect van de kanaalsbandbreedte op een digitaal signaal

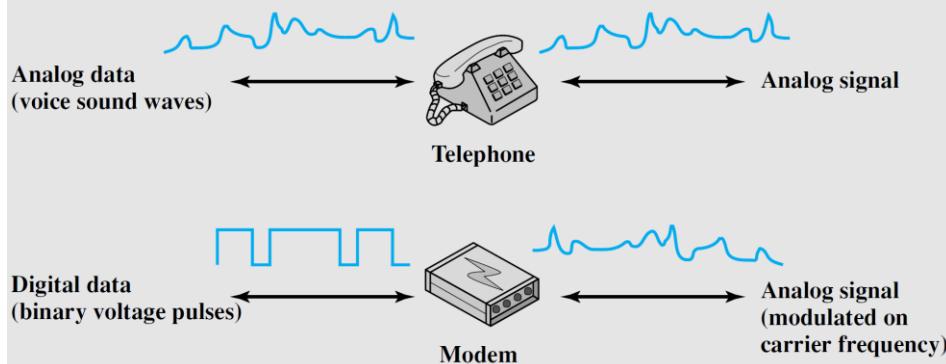
Omgekeerd : een gegeven medium-bandbreedte kan verschillende 'data rates' ondersteunen, afh. van de ontvanger zijn mogelijkheid om signalelementen te onderscheiden in het bijzijn van ruis en andere negatieve invloeden. Het spreekt voor zich dat een medium met een hogere bandbreedte een hogere kost met zich meebrengt en het dus van de inventiviteit van de signaalcodering zal afhangen om het signaalspectrum zo perfect mogelijk te 'mappen' in de bandbreedte van het medium.

### 1.1.3 ANALOGE EN DIGITALE TRANSMISSIE.

Als we data van een punt naar een ander punt wensen te verzenden moeten we ons bewust zijn van de aard van de **data** enerzijds en de fysieke propagatie van het overgezonden **signaal** anderzijds. Het begrip **transmissie** kan dan worden omschreven als de communicatie van **data** door de propagatie en processing van **signalen**. Onderstaande figuur geeft de mogelijkheden weer :

(Analoge) spraak heeft typisch een bandbreedte van 100Hz tot 7KHz, en wordt identiek verzonden. Het

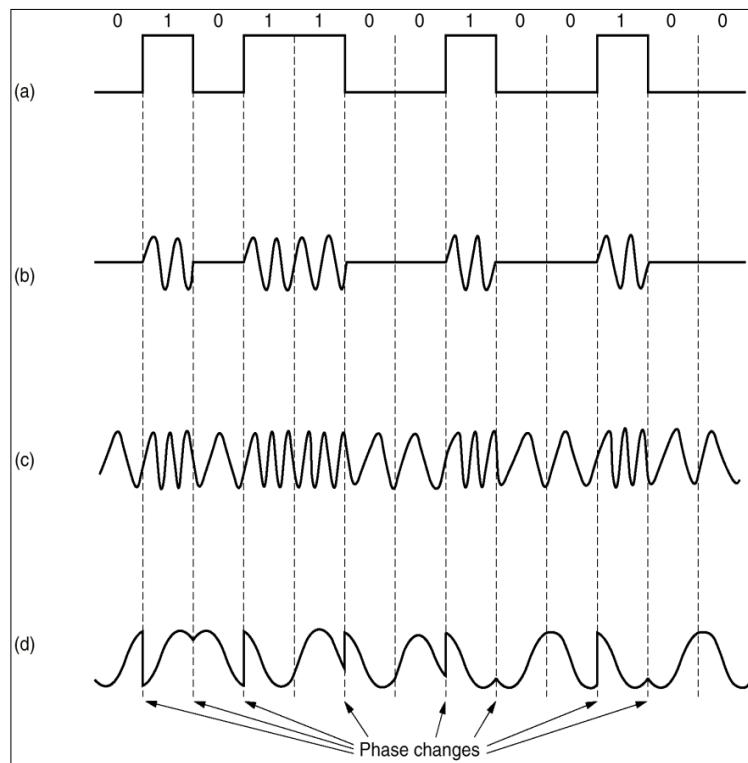
**Analog signals: Represent data with continuously varying electromagnetic wave**



telefoon-netwerk heeft een bandbreedte van 300Hz tot 3400Hz waardoor de spraak vervormd wordt maar toch nog perfect interpreteerbaar is voor de ontvanger. Ook ruis vervormt het signaal.

Figuur 29 Analoge transmissie van data.

Op hetzelfde medium kan **digitale data** worden verzonden, maar door een **analoog signaal**. Het digitale signaal (bvb. 28Kbps) heeft té hoge frekwentiecomponenten die niet door het medium worden doorgelaten. Daarom zal een **modem** de bits coderen door frekwenties die wél door het medium worden doorgelaten te moduleren in :



b) amplitude → AM, zie ook bv. Figuur 26 'Aangepaste' of analoge transmissie van een digitaal signaal.

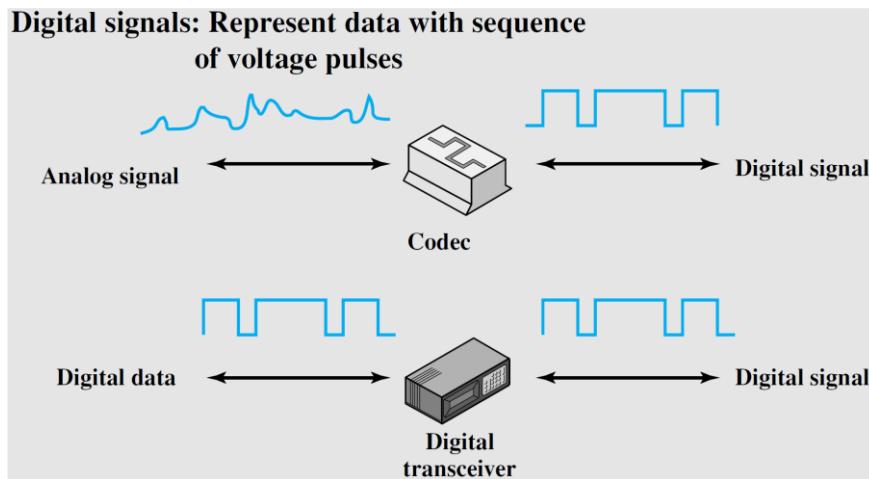
c) frekwentie → FM

d) fase → PM

of een combinatie ervan → QAM, quadratuur amplitude modulatie. (zie ook pt. 1.5.1 op p.116).

Figuur 30 AM, FM en PM modulatie van een digitale data.

Omgekeerd wordt in de moderne transmissiesystemen de signalen **digitaal** getransporteerd. Daarom moet spraakdata worden gedigitaliseerd door een codec, zie ook pt. 1.5.2 op p. 118. De resulterende bitstroom wordt verzonden en in de ontvanger wordt de **analoge data** gereconstrueerd.



Ook **digitale data** wordt over de digitale transmissie-systemen verzonden, maar meestal gecodeerd om de ontvanger toe te laten te synchroniseren en de storende invloeden teniet te doen, zie verder.

Figuur 31 Digitale transmissie van data.

**Analoge transmissie** moet voorzien in **versterkers** om het analoge signaal, dat verzwakt door de afstand, terug op te krikken. Spijtig genoeg versterkt men dan ook de ruiscomponenten en door meerdere versterkers in cascade te plaatsen om langere afstanden te overbruggen, wordt het signaal steeds meer en meer vervormd. Gelukkig kan zowel spraak- als video-data wel een en ander van vervorming verdragen. **Digitale data** zal hierdoor echter een serieuze foutenlast oplopen. Voor meer informatie ivm. de transmissie van analoge signalen wordt verwezen naar de cursus telecommunicatie.

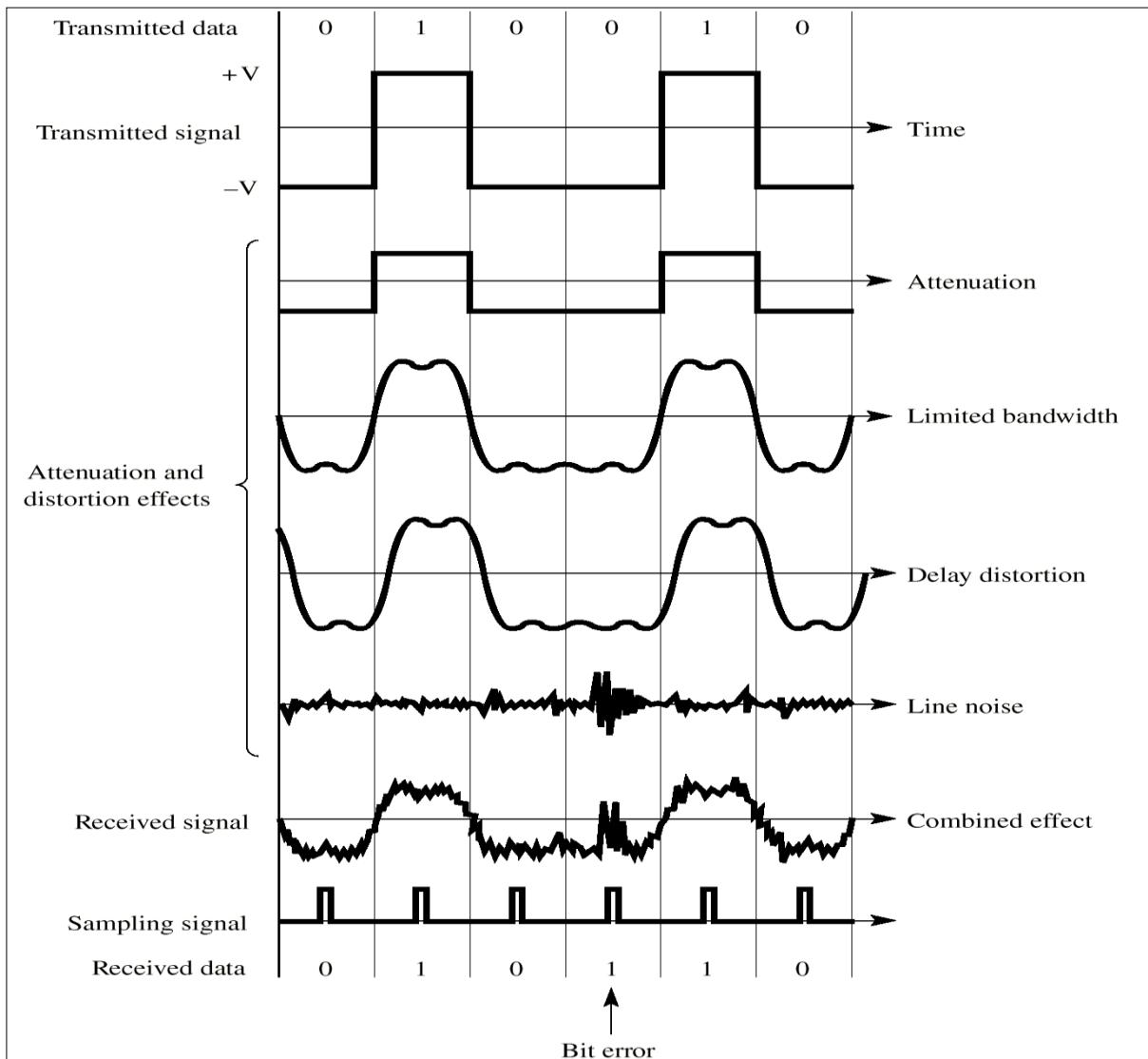
**Digitale transmissie** echter is begaan met de **INHOUD** van het signaal, i.t.t. analoge transmissie. Een digitaal signaal kan slechts over een beperkte afstand worden verzonden alvorens ruis en andere storingen de **integriteit** van de data verstören. Om grotere afstanden te bekomen worden **repeaters** gebruikt. Een repeater ontvangt het signaal, regenerert de data, en verzendt een nieuw signaal om de verzwakking teniet te doen.

De voordelen van digitale transmissiesystemen, zowel voor lange afstand als binnenin 1 gebouw liggen voor de hand :

- **Digitale technologie** : de VLSI ontwikkeling in digitale systemen kent een ongeëvenaarde groei en bijgevolg daling in kostprijs. Analoge ontwikkelingen volgen deze trend niet zo sterk.
- **Data integriteit** : repeaters, i.t.t. amplifiers, kennen geen cumulatieve verstoring van het signaal door bvb. ruis. Dit betekent dat de data makkelijker en verder kan worden getransporteerd zonder aantasting van zijn integriteit.
- **Capaciteit** : Op de hogesnelheids'trunks' worden verschillende signalen gemultiplexed om de capaciteit van het kanaal zo economisch mogelijk te gebruiken. Digitale multiplexing (door TDM) is veel eenvoudiger uit te voeren dan analoge (door FDM).
- **Beveiliging** : Encryptie-technieken laten toe om zowel digitale als analoge data die gedigitaliseerd is, te beveiligen.
- **Integratie** : door analoge data ook digitaal voor te stellen kunnen beiden tezamen op één medium worden getransporteerd ➔ integratie van spraak, video én data.

### 1.1.4 TRANSMISSIONEERVERVORMING

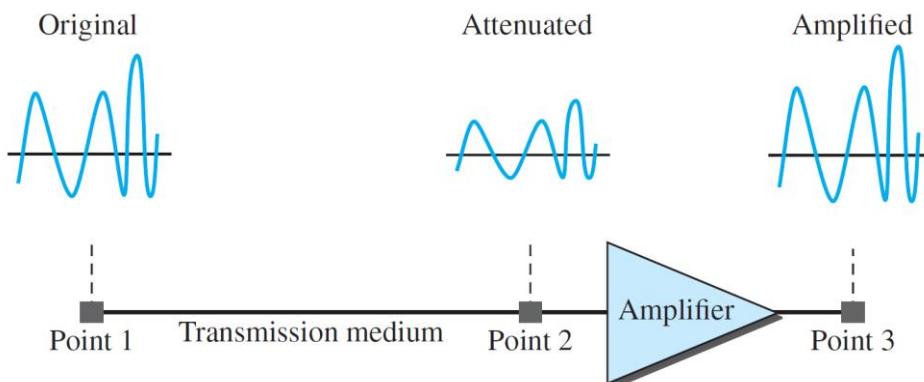
Elk signaal is onderhevig aan **verzwakking**, beperking van de **bandbreedte** van het medium, **vertragingen** en **ruis**. Voor analoge signalen betekent dit kwaliteitsverlies terwijl digitale signalen last krijgen van bitfouten.



Figuur 32 Effecten van verzwakking en vervormingen op een digitaal signaal.

## VERZWAKKING

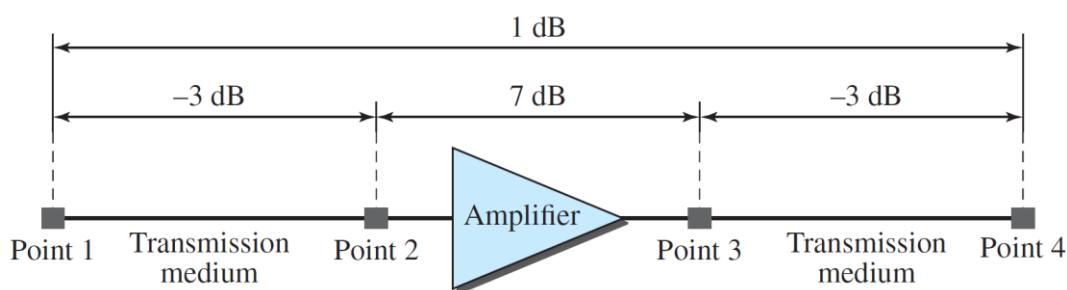
Om dit effect in te perken wordt de lengte van het transmissiemedium meestal beperkt, of voorzien van versterkers (A) of 'repeaters' (D). Aangezien dat verzwakking toeneemt met de **frequentie** (skin effect) zal hierdoor ook het signaal vervormen en moeten de versterkers een '**equalizing**'- functie bevatten om dit tegen te gaan. Voor geleide media is de verzwakking logaritmisch en bijgevolg uitgedrukt in dB/m.



Figuur 33 verzwakking van een signaal.

De verzwakking wordt meestal uitgedrukt in  $\text{dB} = 10 \log (P_2/P_1) = 20 \log (V_2/V_1)$ .

Op deze manier kan met versterkingen en verzwakkingen zondermeer optellen:  $\text{dB} = -3 + 7 - 3 = +1$ .



Figuur 34 dB calculatie.

## BANDBREEDTE beperking.

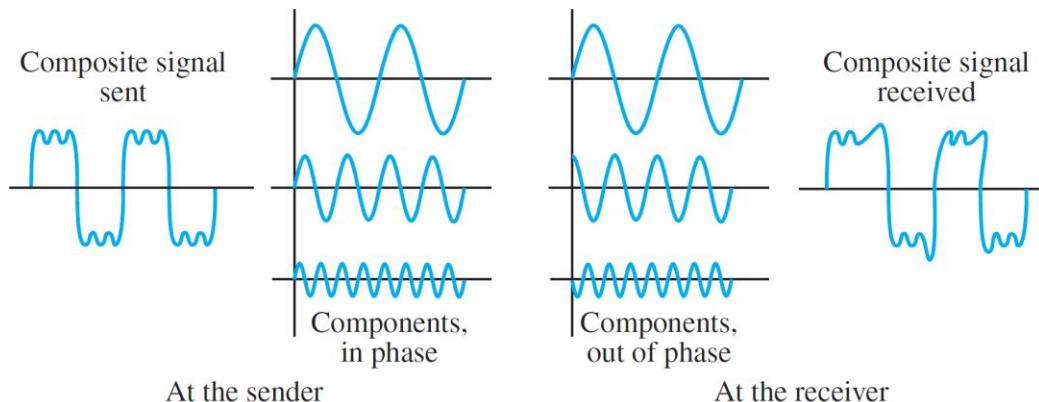
Dit is reeds uitvoerig besproken in pt. 1.1.2 Frequentiespectrum en bandbreedte op p.32 e.v.

Elk transmissiemedium heeft een gedefinieerde bandbreedte. We moeten dus m.b.v. fourieranalyse het effect nagaan dat het medium zal hebben op het verzonden signaal : enkel de frequentiecomponenten die door het kanaal worden doorgelaten blijven over.

## VERTRAGING VAN HET SIGNAAL → VERVORMING OF DISTORTION

De voortplantingssnelheid van een **frequentiecomponent** van het signaal is afh. van zijn frequentie. Het gevolg is dat bij de ontvanger de verschillende frequentiecomponenten met een fase aankomen die verschilt met de fase die ze hadden bij de zender.

Voor een gelimiteerde bandbreedte is de snelheid typisch het grootst rond de centerfrequentie en trager aan de rand van de band. Als het signaal in de ontvanger aankomt is het dus vervormd.



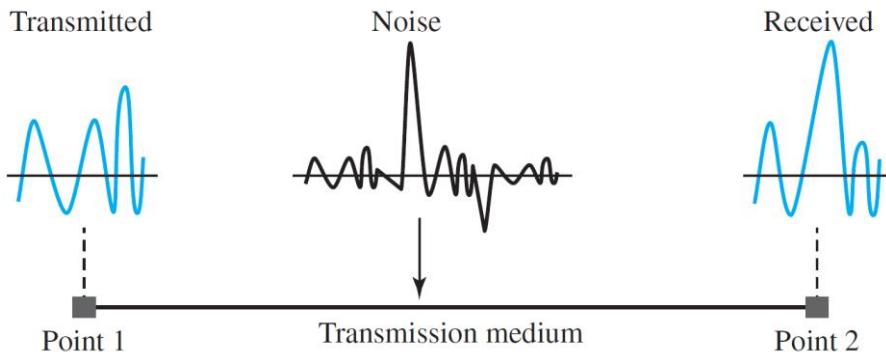
Figuur 35 vervorming van een signaal.

Voor digitale signalen wordt dit effect des te erger naarmate de bitsnelheid toeneemt : de vertraagde frequentiecomponenten beginnen de volgende bit te verstören. Dit wordt ook wel '**InterSymbol Interference**' of ISI genoemd en is een belangrijke begrenzing voor de bitrate op een kanaal.

Aangezien het ontvangen signaal wordt gesampled in het midden kan dit leiden tot verkeerde interpretaties.

## RUIS

Een ontvangen signaal bestaat dus uit het verzonden signaal, **vervormd** door **bovenstaande** effecten, plus een reeks **ongewenste** signalen opgepikt tijdens de transmissie → **ruis**. Deze ruis is **dé** beperkende factor in de performantie van een communicatiesysteem.



Figuur 36 Ruis.

Ruis wordt typisch gedefinieerd door de SNR, Signal to Noise Ratio.

$$\text{SNR} = P_{\text{signaal}} / P_{\text{ruis}} \text{ of in dB : } \text{SNR}_{\text{dB}} = 10 \log (P_{\text{signaal}} / P_{\text{ruis}})$$

Ruis kan onderverdeeld worden in 4 categoriën :

- thermische ruis
- intermodulatieruis
- crosstalk
- impulseruis

**Thermische ruis** is het gevolg van de thermische agitatie van elektronen en is aanwezig in alle elektronische apparaten en media. Het spectrum ervan is uniform verdeeld over alle frekwenties → ‘witte ruis’ en is een functie van de temperatuur. Het kan niet worden uitgeschakeld en plaatst zodoende een bovengrens aan de kanaalcapaciteit → zie verder, Shannon’s capaciteitsformule.

Wanneer 2 signalen  $f_1$  en  $f_2$  eenzelfde medium delen kan er **intermodulatieruis** ontstaan wanneer er een niet-lineair systeem aanwezig is bvb. in zender, ontvanger of medium. Normaal zijn deze componenten lineair, maar door oversturing of slecht functioneren ontstaan som- en verschilfrekwenties,  $f_1+f_2$  en  $f_1-f_2$ , of veelvouden ervan.

**Crosstalk** is bvb. hoorbaar wanneer op de achtergrond een ander telefoongesprek wordt gehoord, meestal bij oude, analoge centrales. Het is een ongewenste capacitieve koppeling tussen 2 naast elkaar gelegen ‘twisted pairs’ en zal bvb. in LAN omgevingen moeten worden tenietgedaan, zie ook cross talk : NEXT bij 10/100base op p.57, en cross talk : NEXT en FEXT bij 1000base/10GBASE op p. 59.

Alle bovenstaande ruistypes zijn berekenbaar en/of te bestrijden door R&D ingenieurs. **Impulseruis** is echter niet-continu en bestaat uit onregelmatige en hevige pulsen of spikes van korte duur. Het wordt opgewekt door elektromagnetische storingen zoals bliksems, schakelelementen van vermogensturingen, of

zelfs fouten in het communicatiesysteem. Het is meestal geen groot probleem voor analoge data, spraak of video, maar is de belangrijkste oorzaak van fouten in een digitaal datacommunicatiesysteem, juist omdat van zijn onberekenbaarheid.

### 1.1.5 TRANSMISSIONSCAPACITEIT.

Een zeer belangrijk gegeven in datacommunicatie is de snelheid waarmee we bits over een kanaal kunnen sturen. Dit hangt af van 3 factoren:

1. De beschikbare bandbreedte
2. Het type signalen we sturen
3. De kwaliteit van het kanaal, of hoeveel ruis er onderweg wordt opgepikt.

Er zijn 2 benaderingen om de datarate te berekenen:

- a. Nyquist voor een ruisvrij medium
- b. Shannon voor een medium met ruis.

#### NYQUIST BANDBREEDTE

Nyquist bepaalde reeds in 1924 het **absoluut maximum** in informatiesnelheid ( $C$ ) voor een ruisvrij! kanaal :

$$\text{Nyquist : } C = 2 * B * \log_2 M$$

In een omgeving **zonder** storingen is de informatiesnelheid ( $C$ ) evenredig met de bandbreedte ( $B$ ) en het aantal verschillende symbolen ( $M$ ).

Bvb. een telefoonkanaal met een bandbreedte van 3000Hz. Als er 2 niveaus zouden worden gestuurd, bv. NRZ kan max. 6000 bps (hier = baud) bereikt worden, ... als het kanaal ruisvrij zou zijn!

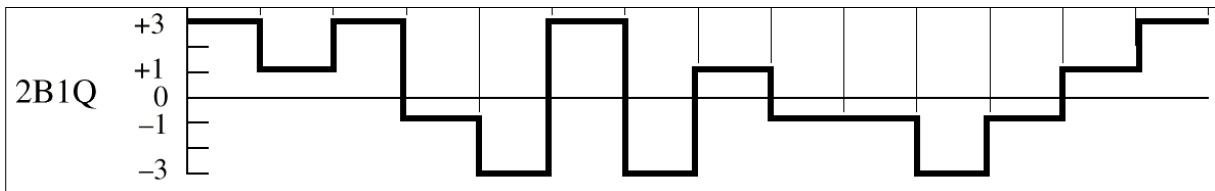
Vb.  $M = 2$ , dan is  $C_{max} = 2 * 3000 * \log_2(2) = 6000 \text{ bps!!!}$

Om deze beperkingen te omzeilen kunnen we  $M$  vergroten en dus **meer dan 2** signaalniveaus gebruiken. Dit betekent dat elk signaalelement **meer dan een bit** voorstelt.

Algemeen, als  $M$  = aantal signaalniveaus, is  $m$  het aantal bits per signaal element :

$$m = \log_2 M$$

Praktisch voorbeeld : de 2B1Q code = 2 Bits per 1 Quat (= quaternair niveau  $\rightarrow M=4$ )  $\rightarrow m=2$



Figuur 37 2B1Q baudreducerende code

Er worden per signaal niveau 2 bits overgedragen. Bvb. +3 = '10', +1 = '11', -1 = '01' en -3 = '00'. (het eerste bit is het teken  $+1^L$ ,  $-0^L$ , het 2<sup>e</sup> de amplitude :  $1V=1^L$ ,  $3V=0^L$ )

Hier:  $M = 4$ , dan is  $C_{max} = 2 * 3000 * \log_2(4) = 12000 \text{ bps}!!!$

De snelheid waarmee het signaal wijzigt is de 'signaling rate' of 'baudrate',  $R_s$ : uitgedrukt in 'baud'.  $R_s = \#$  signaalveranderingen per seconde. ( $= 1 / T$ , de tijd van een 'signaal-slot' uit Figuur 37.). Men kan stellen dat  $R_s$  een maat geeft voor de nodige **bandbreedte** van het kanaal.

De datasnelheid ( $R$ ), de 'data rate', is dan in bits per seconde bps :

$$R = R_s \log_2 M$$

Op deze manier kunnen meer bps overgezonden worden dan de baudrate of m.a.w. de bandbreedte van het medium toelaat  $\rightarrow$  baudreducerende codes !

(Deze code wordt gebruikt op de U-interface van ISDN, een twisted pair lijn van enkele km, waarover 160kbps moet, maar slechts ...80baud.)

Het lijkt volgens Nyquist,  $C = 2 \times B \times \log_2 M$ , alsof we met een gegeven bandbreedte  $B$  een onbeperkte datarate kunnen realiseren door  $M$  steeds op te voeren. Theoretisch klopt dit wel maar praktisch is er een grens: de ontvanger moet al de verschillende signalelementen van elkaar kunnen onderscheiden.

Men zal daarom moduleren bvb. door QAM, bvb. V32bis = 14400bps modem :  $M = 128 \rightarrow 7\text{bit} = 6 \text{ data} + 1 \text{ trellisbit}$ .  $\rightarrow$  Hier is BAUD  $\neq$  bps. Meer info vind je ook in pt. 1.5.1 Analoge telefonielijnen op p. 116. Ook pt. 1.5.3 xDSL / ADSL, (Asymmetric) Digital Subscriber Line vanaf p. 126 behandelt deze werkwijze. We zien dat een DSL slot van 4KHZ wordt gemoduleerd tot 15 bits wat overeenkomt met  $2^{15} = 32768$  verschillende signalelementen ! En dit voor bv. ADSL1 zelfs al met 250 carriers op 1 oude UTP kabel!

In de realiteit zorgt ruis er wel voor dat een ontvanger niet zo duidelijk een signalelement kan onderscheiden. Daarom ... Shannon.

## SHANNON'S CAPACITEITSFORMULE.

In 1948 zette Shannon het werk van Nyquist verder en breidde het uit tot een kanaal dat onderhevig is aan witte ruis :

$$\text{Shannon-Hartley : } C = B * \log_2 \left( 1 + \frac{S}{N} \right)$$

In een omgeving met (witte) ruis ( $N$ ) is de overdrachtscapaciteit ( $C$ ) afh. v. de bandbreedte ( $B$ ) en de signaal ( $S$ ) over ruis ( $N$ ) verhouding.

Merk op dat er geen indicatie is van  $M$ , de modulatiediepte. Dat betekent dat ongeacht  $M$ , de maximale capaciteit bepaalt wordt door de karakteristieken van het kanaal.

*Vb. een telefoonlijn met bandbreedte 3000 Hz en S/N van 20 db of S/N = 100. Dit heeft een maximale overdrachtscapaciteit van :*

$$C = 3000 * \log_2(1+100) = 19974 \text{ bps !!!}$$

Gelukkig is de S/N verhouding meestal beter : 30dB →  $C = 3000 * \log_2(1+1000) = 29901 \text{ bps.}$

In de praktijk maak je gebruik van beide formules om je modulatie te berekenen. Stel dat we een kanaal hebben met een bandbreedte  $B = 1\text{MHz}$  met een SNR of S/N = 63.

$$C = 1\,000\,000 * \log_2(1+63) = 6 \text{ Mbps !!!}$$

Dit is de theoretische bovenlimiet. We kiezen bv. 4Mbps. Dan kunnen we de benodigde modulatiediepte  $M$  berekenen door Nyquist:  $C = 2 * B * \log_2 M$ .

$$C = 4 \text{ Mbps} = 2 * 1\,000\,000 * \log_2 M.$$

$M$  is dan 4, bv. De 2B1Q code.

## PERFORMANTIE.

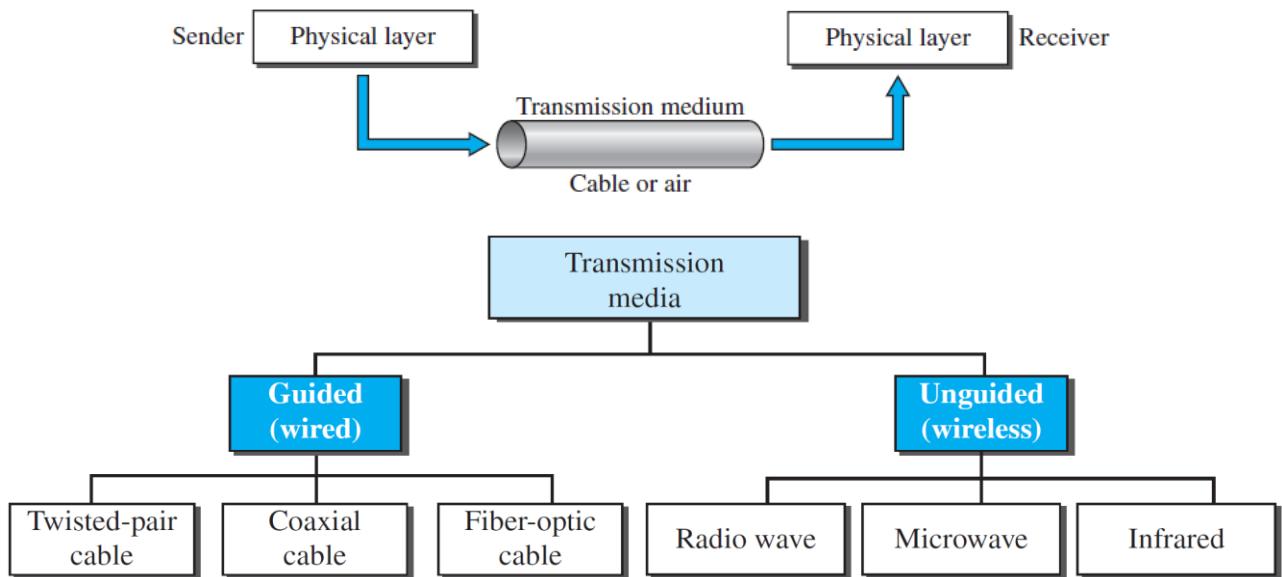
De Shannon formule berekent de theoretisch maximum haalbare kanaalcapaciteit bij aanwezigheid van **witte ruis**, maar praktisch haalt men dit niet omdat er hier geen rekening wordt gehouden met de andere soorten ruis, noch met verzwakking of 'delay distortion'.

Het lijkt nu dat alles opgelost kan worden door het signaalniveau,  $S$ , sterk op te krikken. Merk echter op dat dan door oversturing intermodulatieruis optreedt waar hier geen rekening mee wordt gehouden.

We hebben het ook reeds over een ander type ruis gehad : de crosstalk. (Ontstaat door capacitieve koppeling.) De meest ingrijpende is de NEXT, zie ook p.57.

## 1.2 TRANSMISSION MEDIA

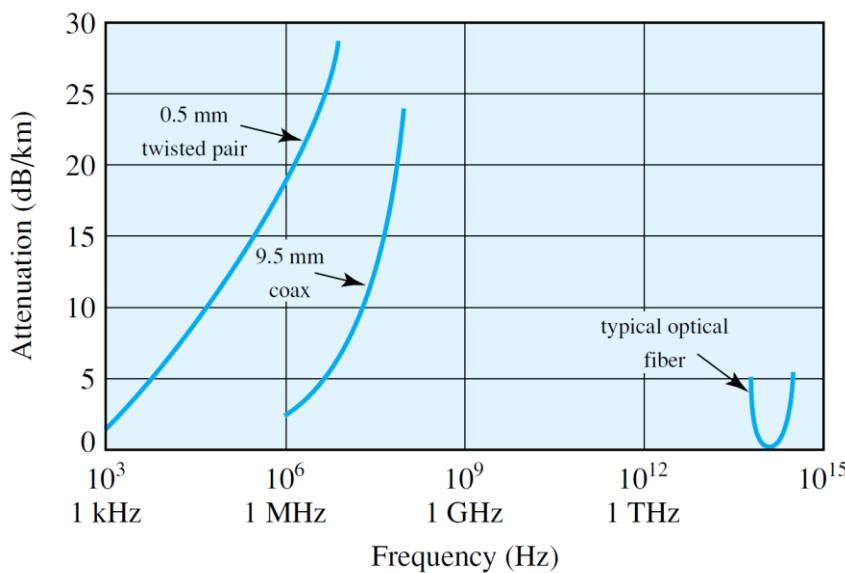
Het **type** transmissiemedium bepaalt de maximum capaciteit (bps) die kan worden overgezonden.



Figuur 38 transmissiemedia, guided vs unguided.

Er moet onderscheid gemaakt worden tussen draadgebonden en draadloze communicatie, ook ‘guided’ resp. ‘unguided media’ genoemd. We zullen hier vooral de ‘guided’ media bekijken met in onderstaande figuur een kwaliteitsvergelijking ervan voor lange afstanden :

In functie van de frekwentie ziet de verzwakking er zo uit :



Figuur 39 Verzwakking van transmissiemedia i.f.v. de frekwentie.

Moderne ‘high speed’ UTP haalt bv. bruikbare bandbreedtes tot 250 MHz (CAT6), 500 MHz (CAT6a), 600 MHz (CAT7), 1GHZ (CAT7a) ... wel voor een bruikbare lengte van **100 m** !. M.a.w. UTP komt aardig dicht tegen coax kabels qua kwaliteiten, maar nog veraf van fiber met zijn zeer lage verzwakking (0,2dB/km) én zeer hoge frekwenties.

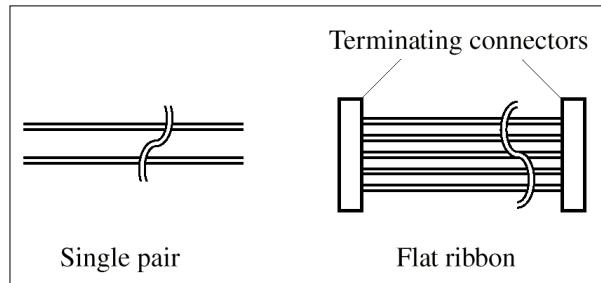
We zien dat fiber veruit over de beste papieren beschikt, alleen ... de kostprijs ervan is niet in deze tabel opgenomen.

Volledigheidshalve moeten we hierbij opmerken dat de twisted pair in deze figuur ‘low quality’ twisted pair is zoals hij (vroeger) gebruikt werd in telefonie-bundels. De verzwakking staat ook in dB/Km !

## 2-DRAADSVERBINDING.

Als voorbeeld voor de besprekking van kabel-imperfecties nemen we deze meest eenvoudige vorm van

medium. Hij kan enkel gebruikt worden voor verbindingen tot  $\pm 50\text{m}$  tegen beperkte data-rates :  $\pm 19,2\text{k}\text{bps} \rightarrow$  bv. RS232.



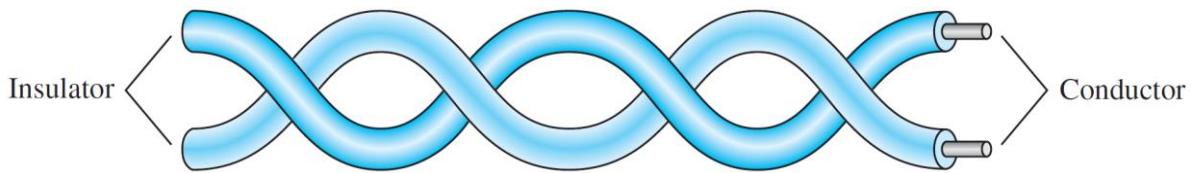
Figuur 40 Een 2-draadsverbinding

Een belangrijk probleem dat we zien ontstaan is de **capacitieve koppeling** tussen de geleiders in een kabel, beter bekend als '**cross talk**'. 2 geleiders met een dielectricum ertussen vormen een capaciteit en deze gaat geleiden bij hogere frekwenties :  $Z=1/\omega C$ .

Bovendien maakt de structuur van de kabel hem sterk onderhevig voor het oppikken van **ruissignalen** van andere bronnen : **EMI**, ElectroMagnetische Interferentie.

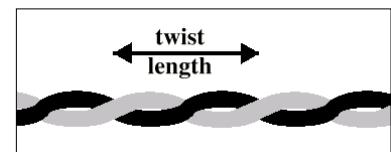
Betere oplossingen dienen dus te worden gezocht.

### 1.2.1 TWISTED PAIR



Figuur 41 Twisted pair kabel

Dit type kabel wordt gestuurd door ‘balanced’ of ‘differential’ transmitters en receivers (zie ook pt. 1.4.7 op p. 109). Enkel de verschilspanning (/stroom) wordt gemeten. Als nu 1 van de draden dichter bij een ruisbron is, zal 1 ‘twijn’ verder dat omgekeerd zijn waardoor dit het ruiseffect neutraliseert. Hoe dichter de kabels getwijnd zijn hoe beter deze ruisbestendigheid is. De opgenomen EMI: straling wordt opgenomen door **beide** geleiders waardoor de verschilspanning beperkt blijft.



Figuur 42 Twisted pair kabel ‘twist length’.

#### UTP

Bovendien worden in 1 kabel bv. 4 paren onderling nog eens getwijnd waardoor de **capacitieve** koppeling, of crosstalk ook wordt beperkt. Dit levert bijgevolg nog betere resultaten op. De ‘twist length’ is verschillend tussen nabijgelegen paren, en dit ook om de crosstalk tussen de nabij gelegen paren (in 1 kabel) te beperken.

#### Unshielded twisted pair

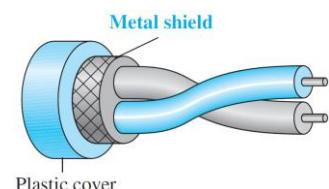


Figuur 43 Twisted pair kabel (UTP)

Deze uitvoering, beter bekend als **UTP** (‘Unshielded Twisted Pair’) is veruit de meest gebruikte (koper-) bekabeling voor datacommunicatie, zowel in telefonie als netwerkomgevingen.

#### STP

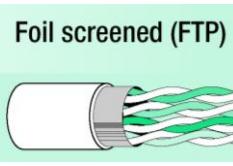
Om nog beter eigenschappen te verkrijgen voorzien sommige fabrikanten een ‘shielding’ rondom de geleiderparen → **STP** = ‘Shielded Twisted Pair’. Dit verbetert de kwaliteiten omdat ruis nu veel meer moeite heeft om door het metalen shield te dringen. Het is wel lastiger aansluiten + duurder.



Figuur 44 Twisted pair kabel (STP)

#### FTP

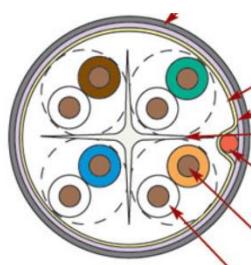
Ook een ‘folie’ rondom de volledige kabel wordt soms gelegd om dezelfde reden → **FTP** = ‘Foiled Twisted Pair’.



Figuur 45 Twisted pair kabel (FTP)

Mogelijk, bv. CAT6A, CAT7, gebruikt men beide shielding technieken → **SFTP** = ‘Shielded and Foiled Twisted Pair’ of **SSTP** = ‘Shielded and Screened Twisted Pair’.

Tussen de paren is ook soms een ‘geometrie kruis’ voorzien om de paren te optimaliseren voor NEXT verlies.



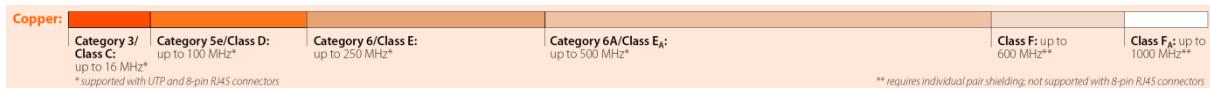
Figuur 46 Twisted pair kabel (SFTP)

We gaan hier dieper in op het gebruik en karakteristieken van deze bekabeling omwille van de belangrijke rol die hij speelt in de communicatiewereld :

### UITVOERINGEN / CATEGORIEN

In 1991 heeft EIA dit type kabel voor het eerst gestandaardiseerd in ANSI/EIA/TIA-568. Ondertussen uitgegroeid met amendementen -A, -B, -C, -D tot ANSI/TIA-568-Dx levert dit volgende kabels op voor de LAN wereld:

	CAT 3 Class C	CAT 5(e) Class D	CAT 6 Class E	CAT 6A Class EA	CAT 7 Class F	CAT 7A Class FA
<b>Bandbreedte</b>	16 MHz	100MHz	250MHz	500MHz	600MHz	1000MHz
<b>Type</b>	UTP	UTP	UTP / FTP	UTP / FTP	SFTP	SFTP
<b>Rel. cost</b>		1	1,2	2	2,2	???



Figuur 47 Vergelijking tussen UTP CATx bekabeling.

De getallen zijn een indicatie van de worst case specs voor een lengte van 100m, wat de typische maximale afstand is voor LAN's.

Reken daar anno 2018 nog een opkomende CAT8 type kabel bij voor LAN 40Gbps implementaties, met een bandbreedte van ... 2000 MHz.

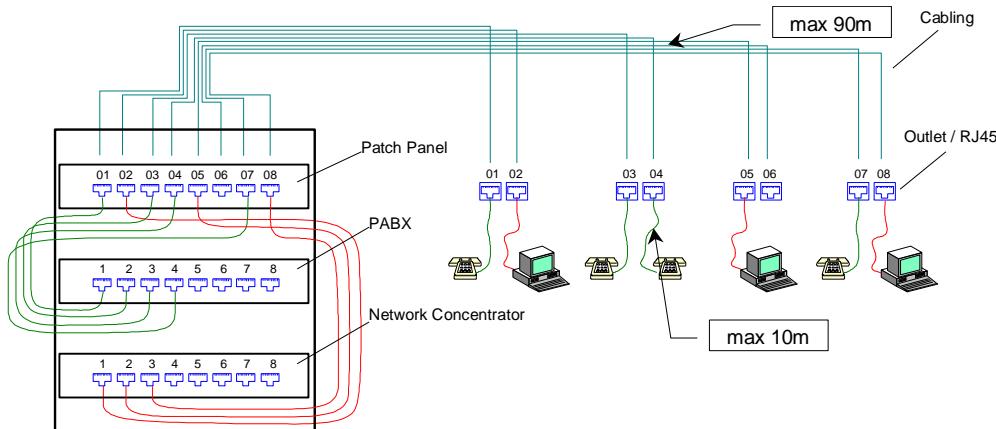
Het meest in gebruik is misschien nog altijd de CAT5e kabel maar nieuwe LAN-installaties worden meer in **CAT6** kabel uitgevoerd, en zelfs CAT7.

**CAT3** wordt ook wel eens **VG** of 'voice grade' kabel genoemd, **CAT5 → DG** of 'data grade'. Het verschil tussen beide is de '**twist**' lengte : **7,5 tot 10cm** voor CAT3 tegen **0,6 tot 0,85 cm** voor CAT5. CAT3 ging mee tot fast ethernet (100base T4), CAT5e tot gigabit ethernet (1000 base T).

Vanaf 10Gig ethernet schakelt men over op CAT6 en CAT6A of hoger.

## BEKABELING (IN HUIS)

Standaardisatie van bekabeling leidt tot het gebruik van twisted pair met een **fysische STERvorm** met maximale afstanden die overbrugd worden van **100m** (90m tot patch).

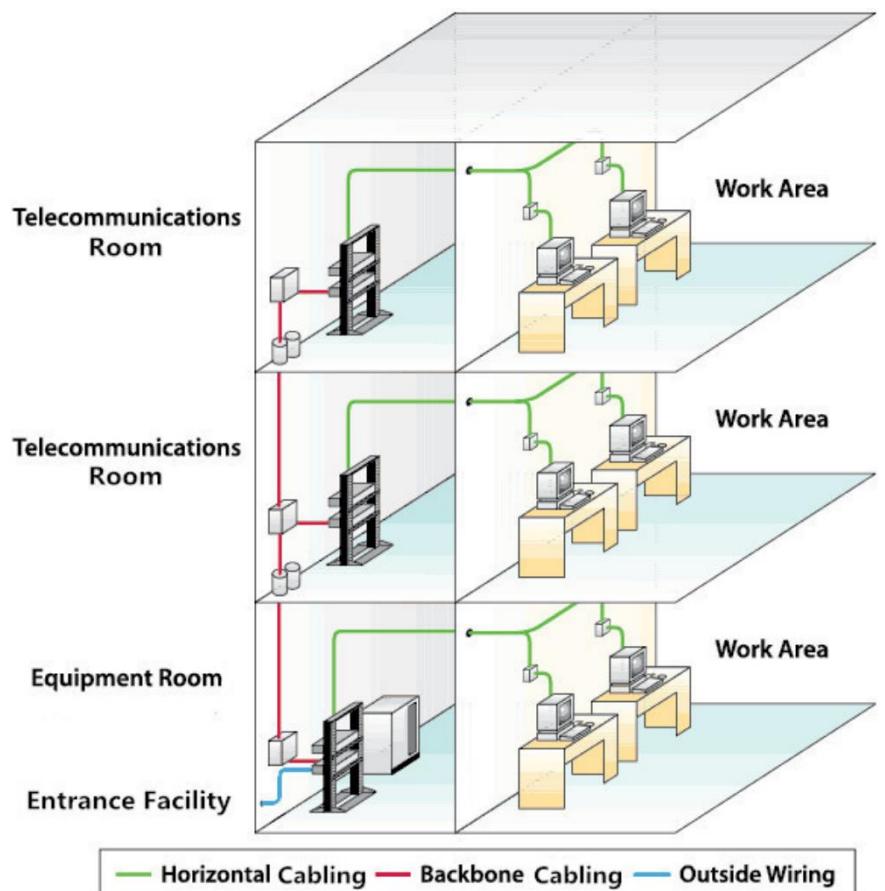


Figuur 48 Patch panel / horizontale bekabeling

We merken dat zowel telefoon als computers via eenzelfde bekabeling aangesloten worden.

Voor LAN's deelt men de bekabeling in 3 delen :

1. **horizontale bekabeling** typisch UTP CAT5/6/7.
2. **verticale bekabeling**, of building backbone max 500m, meestal fiber maar soms ook UTP als de afstanden het toelaten (<100m).
3. **campus backbone** : Een verbinding tussen verschillende gebouwen. Meestal uitgevoerd in **fiber** omwille van de snelheid, de afstanden (>100m ... 10km) maar ook de ongevoeligheid voor blikseminslag!

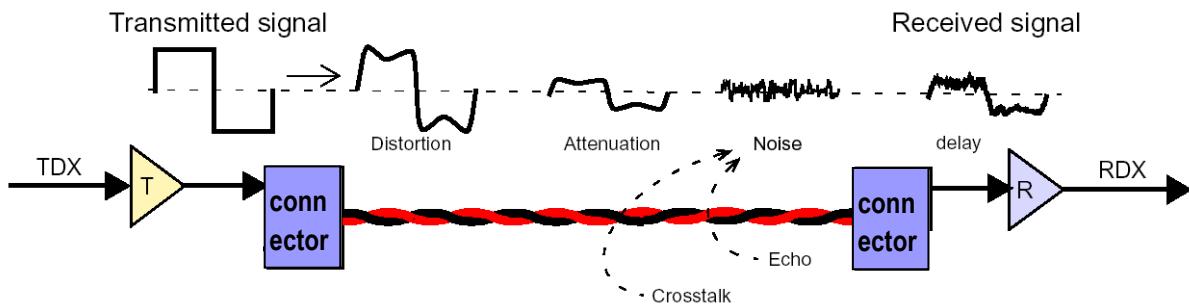


Figuur 49 LAN bekabelingsstructuur

## 1.2.2 ELEKTRONISCHE BEPERKINGEN OP KOPERMEDIA.

Verschillende elektronische aspecten moeten in rekening worden gebracht om een koper geleider voor transmissie te gebruiken. De volgende karakteristieke eigenschappen gelden weliswaar ook voor coax, maar we behandelen ze hier, vooral op UTP.

Alhoewel we de meest algemene effecten, verzwakking en ruis, reeds hebben besproken in pt. 1.1.4 Transmissievervorming op p.43, geven we hier een **meer toegepast** beeld ervan.



Figuur 50 kabeleigenschappen van UTP.

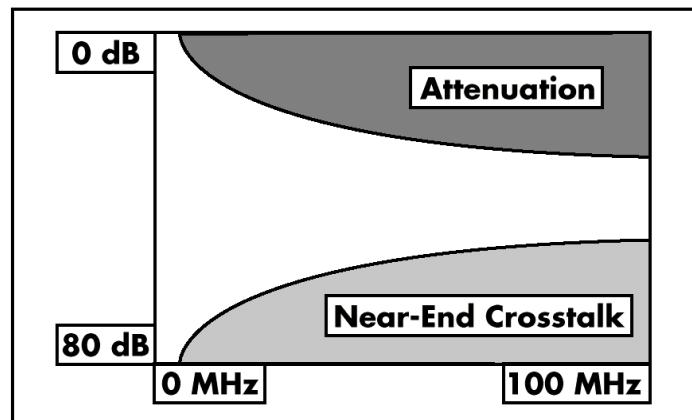
We zien dat UTP verbindingen af te rekenen hebben met **vervormingen**, bv. aan de connectoren waar de draden geen perfecte ‘twist’ meer vertonen. Ook lijdt elk transmissiesysteem aan **verzwakking** over de lengte van de verbinding (zelfs fiber zij het veel minder). Meer nog, deze verzwakking is functie van de frekwentie en zal dus toenemen bij hogere frekwenties. Zie ook Figuur 39 Verzwakking van transmissiemedia i.f.v. de frekwentie. op p.50.

Het grootste probleem zal echter de **ruis** blijken te zijn. Die bestaat enerzijds uit crosstalk vooral komende van nabijgelegen paren, zie verder. O.a. intelligente ‘crosstalk canceler’ schakelingen zullen dit effect moeten minimaliseren. Anderzijds zullen impedantie-‘mismatches’ door het niet exact afsluiten van de karakteristieke impedantie echo’s creeren en bijdragen aan de ruis. Ook deze ruis is functie van de frekwentie en zal toenemen bij hogere frekwenties.

### ACR : ATTENUATION TO CROSSTALK RATIO :

Deze 2 belangrijkste parameters hierboven besproken, geven een beeld van de bruikbaarheid van de kabel. Bedoeling is natuurlijk dat het ontvangen signaal (bovenste lijn) nog te onderscheiden is van de ruiscomponent (hier NEXT maar crosstalk in het algemeen).

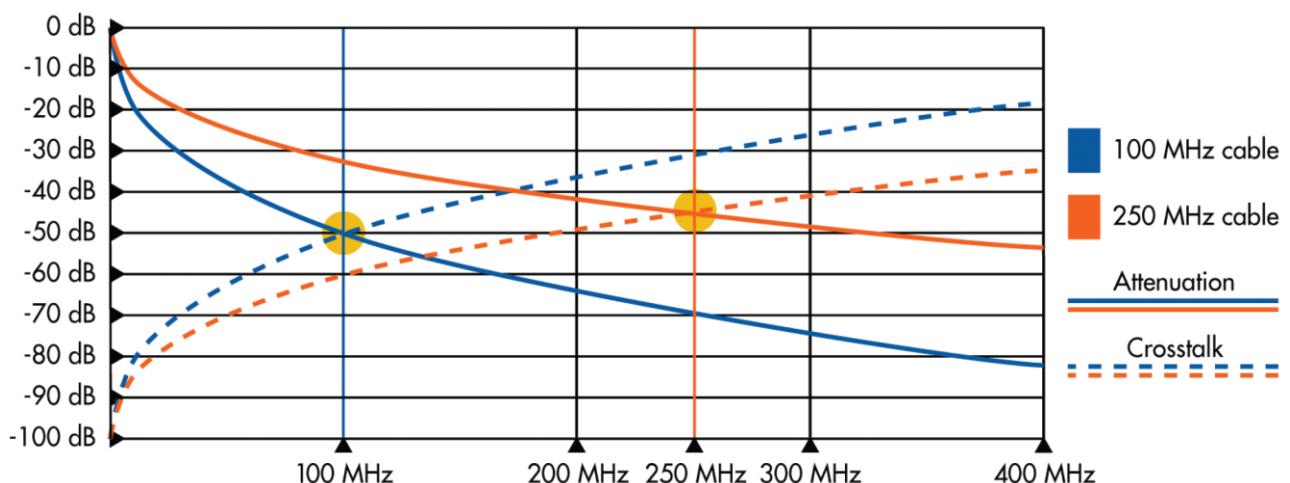
Verzwakking én NEXT beperken dus de bruikbare bandbreedte van het medium (hier bv. CAT5e kabel die bruikbaar is tot 100MHz).



Figuur 51 Attenuation to crosstalk ratio

Door technologische evolutie worden de kabeleigenschappen steeds beter. In den beginne was er CAT3 bekabeling, veel gebruikt in de USA, maar met een beperkt bruikbare bandbreedte (16MHz) → VG = 'Voice Grade'. (+/- 'geschikt voor analoge voice')

Toen Europa overgeschakelde (van coax) naar UTP als standaard, was de kwaliteit al veel beter → CAT5, 'Data Grade': ACR = 100MHz.



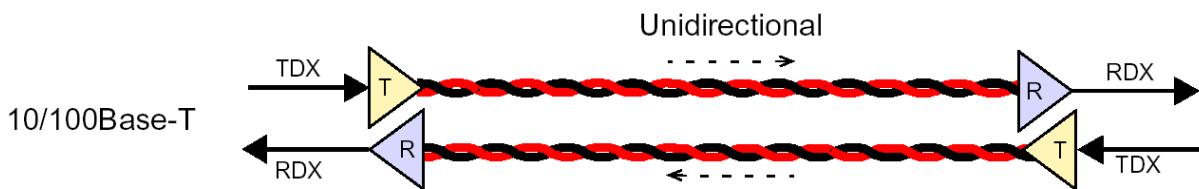
Figuur 52 Evolutie in UTP kabel

(CAT5e breidde de specs uit met FEXT metingen om de compatibiliteit voor gigabit ethernet te kunnen testen, zie verder.)

In bovenstaande figuur is een vergelijking gemaakt tussen CAT5 en CAT6 (250MHz) kabel.

Verdere evoluties vind je ook in bovenstaande tabel.

### CROSS TALK : NEXT BIJ 10/100BASE

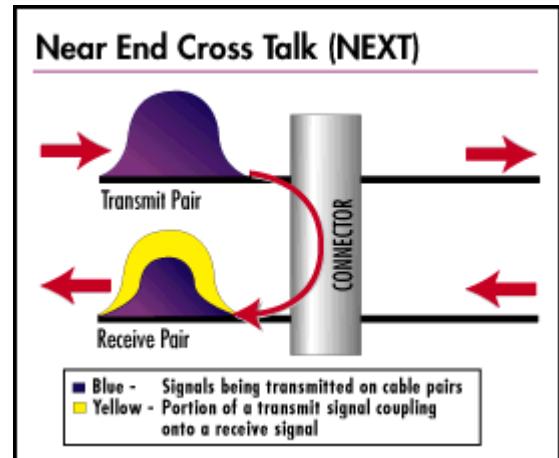


Figuur 53 signaalbanen bij ethernet 10/100BASE.

Voor NEXT dienen we een onderscheid te maken tussen de oudere 10/100 base netwerken en de Gbps ethernets. In het eerste geval wordt er op 1 paar in 1 richting signaal verzonden, van T → R.

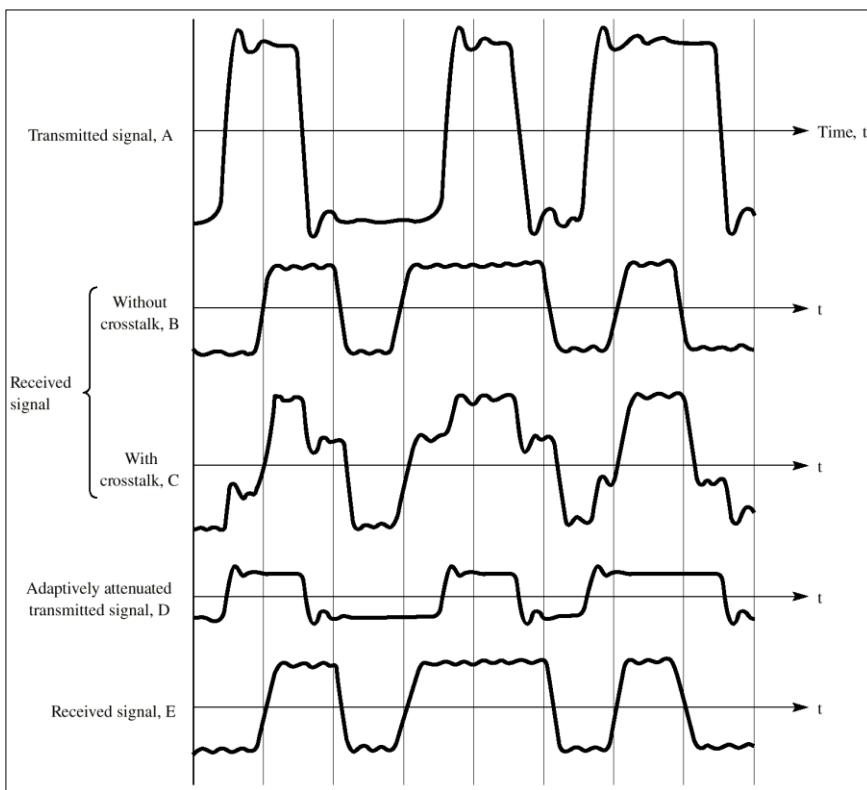
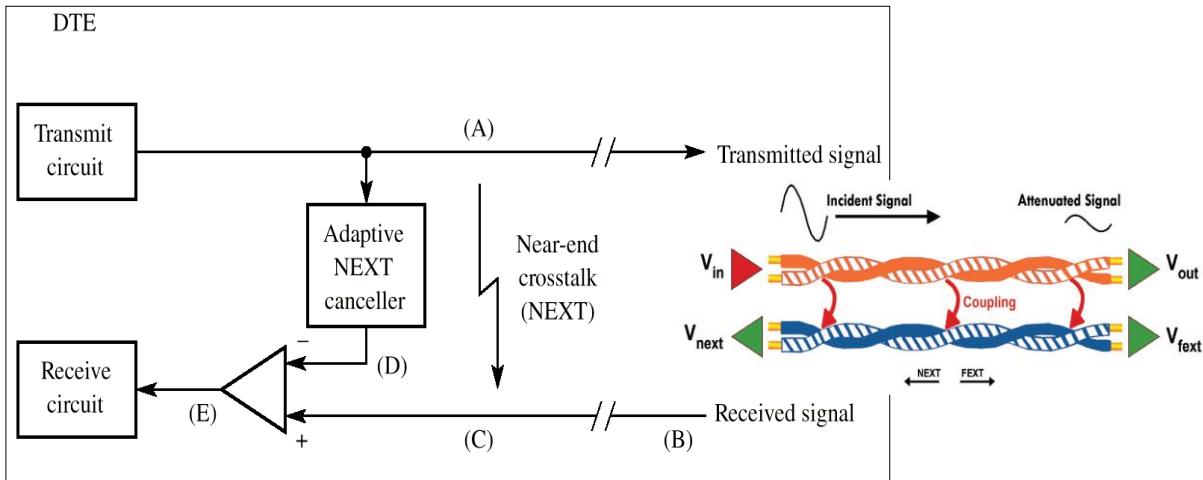
Op pag.51 e.v. hebben we reeds melding gemaakt van de 2 belangrijkste soorten 'storingen' op deze kabels: **inductieve** koppelingen, **EMI** en **capacitieve** koppelingen, **cross talk**. Van de eerste soort hebben we minder last, komt van externe bronnen, en als het toch optreedt is het meestal op een kort tijdstip waardoor 1 pakket fout kan worden overgezonden. Herzenden van dit ene pakket is de oplossing.

Crosstalk door **capacitieve** koppeling, NEXT, is echter inherent aan het systeem verbonden en moet nadrukkelijker bestreden worden. Het (relatief zwak) te ontvangen signaal wordt verstoord door de eigen zender die zijn veel sterker signaal op een nabijgelegen paar **capacitief** koppelt. T.t.z. niet alleen via de connector zoals naaststaande figuur laat vermoeden maar evenzeer door de nabijheid van de paren.



Figuur 54 Near End Cross Talk power spectrum

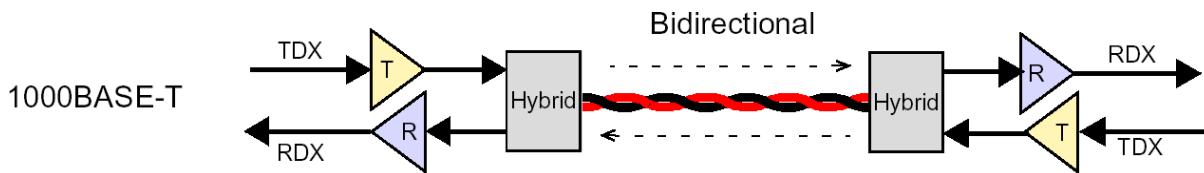
Speciale circuits zoals 'adaptive NEXT cancellers' zijn ontworpen om dit probleem te omzeilen. De canceller wekt adaptief een replica van het verzonden signaal op (hetgeen de crosstalk veroorzaakt) om het dan af te trekken van het ontvangen signaal.



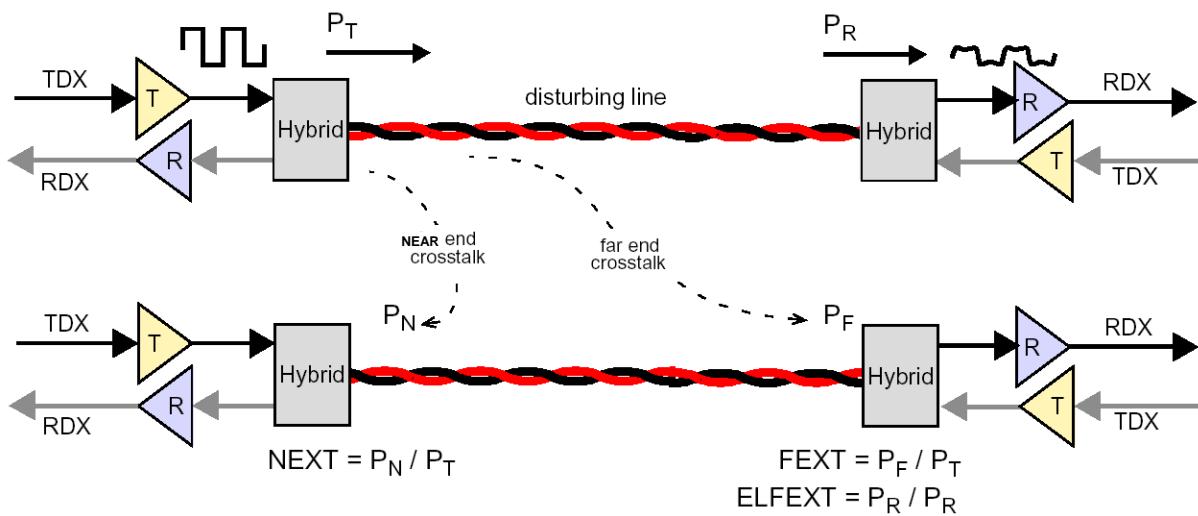
Deze techniek wordt uitvoerig gebruikt in UTP bekabeling met hoge bit rates.

Figuur 55 NEXT cancellers en golfvormen

### CROSS TALK : NEXT EN FEXT BIJ 1000BASE/10GBASE



In het geval van gigabit ethernet (en volgende) wordt er op 1 paar tegelijk in **beide richtingen** een signaal gezonden dmv. een hybride schakeling. Er is maar 1 paar getekend, maar in werkelijkheid gebeurt dit over alle 4 de paren, zie ook verder Figuur 58 op p.60.



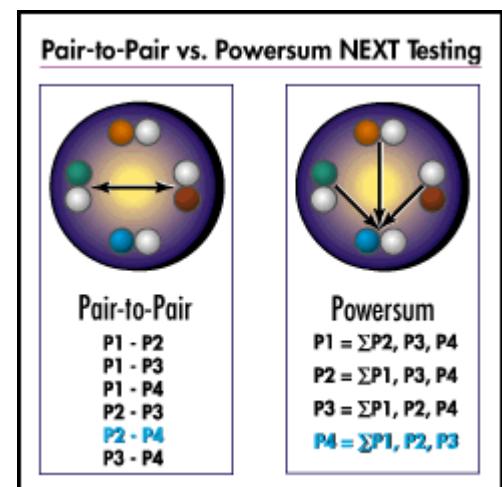
Figuur 56 Near End en Far end Cross Talk

In een kabel zitten meestal meerdere UTP paren (bv. 4) die nu **tegelijk** in werking zijn. Een zender van 1 paar zal zo de nabijgelegen ontvanger(s) verstören, 'near end' maar ook de ontvangers op het einde van de kabel, 'far end'. (Dit laatste, FEXT, is enkel voor gigabit-ethernet en volgende standaarden.)

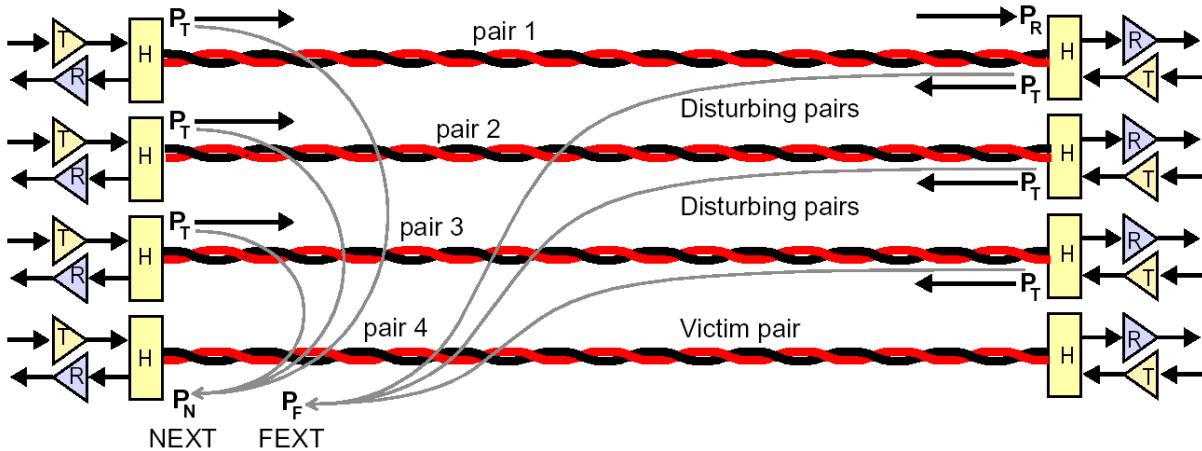
De beïnvloeding van de paren is nu nog een dimensie complexer:

- Powersum NEXT :**

Als er meerdere paren in de kabels signalen dragen, (waartoe men verplicht is met de hogere transporteisen voor gigabit ethernet en hoger) veroorzaken ze **allen** NEXT  
→ powersum.



Figuur 57 Powersum NEXT en FEXT

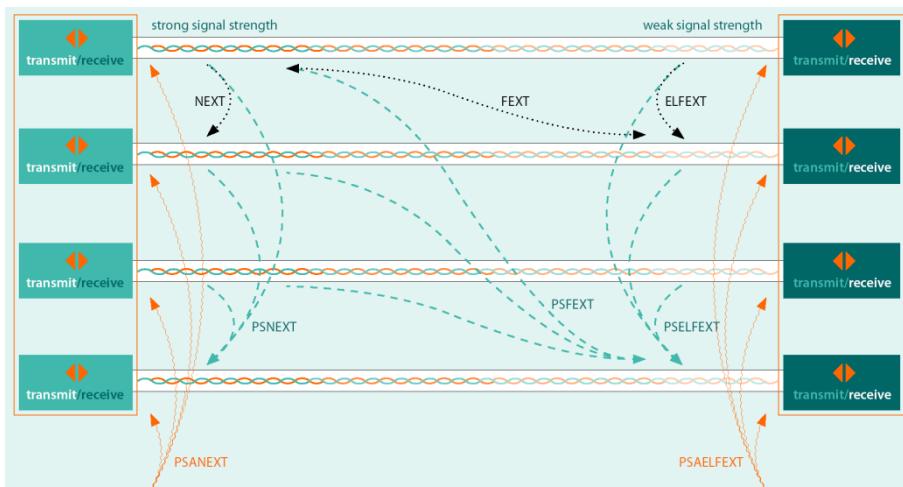


Figuur 58 Powersum NEXT en FEXT

- FEXT :** Zo kan men ook berekenen wat het effect is van de transmitter(s) op het einde van de transmissielijn. Dit zal afhankelijk zijn van de kabellengte. Meestal wordt 'ELFEXT' als parameter vooropgesteld, wat eigenlijk =FEXT/attenuation, om het effect op het ontvangen signaal te berekenen.  $ELFEXT = P_f/P_r$ .  $P_f$  = verstoringss vermogen, far end.  $P_r$  = ontvangen vermogen.

- PSAELFEXT :**  
er moet zelfs rekening gehouden worden met de instraling van de andere paren in de kabel : 'alien crosstalk'.

Figuur 59 PSAELFEXT

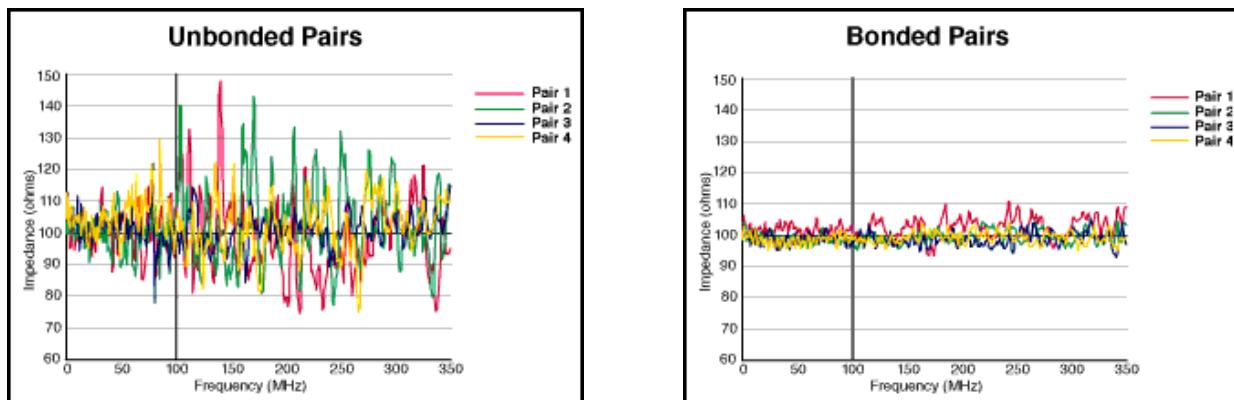


## VERZWAKKING OF ATTENUATION

### KARAKTERISTIEKE IMPEDANTIE VAN UTP

Karakteristieke impedantie ( $Z_0$ ) komt overeen met de ingangsimpedantie van een uniforme, oneindig lange transmissielijn. Ze is een functie van de frekwentie en onafh.v. de lengte van de kabel. Typische karakteristieke impedanties zijn voor coax : 50 (LAN) of 75Ω (TV), **UTP : 100 Ω** soms 120 of 150...

Typisch trachten kabelfabrikanten UTP kabel te maken met een **konstante** kar. imp. van **100Ω (CAT5)**. Op de grafiek is duidelijk te zien dat dit **niet konstant** is voor het volledige frekwentiebereik. De GEMIDDELDE afwijking mag 15% bedragen, pieken zijn er tot 30%- 40%.

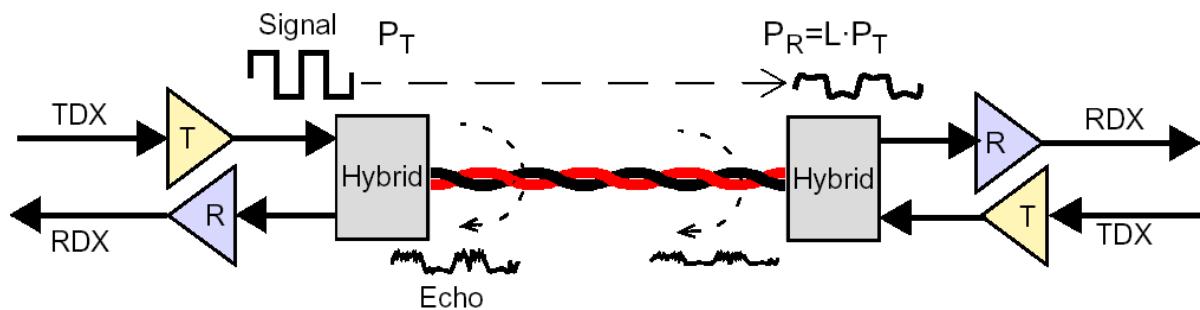


Figuur 60 Karakteristieke impedantie van twisted pair lijnen

### 'RETURN LOSS'

Een optimale transmissie steunt op een (bijna) perfecte kar. impedantie omdat de maximale vermogensoverdracht plaats heeft wanneer de bron of 'source' dezelfde impedantie heeft als de belasting of 'load'. Zowel zender als ontvanger moeten dus een impedantie hebben =  $Z_0$ .

Als de lijn- en transmitter-impedantie niet exact overeenkomen wordt een deel ervan gereflecteerd richting transmitter → **'Return Loss'**.



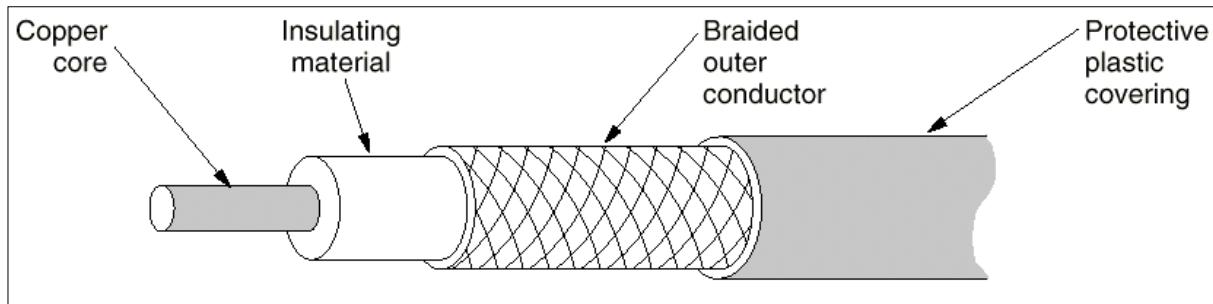
Figuur 61 echo/return loss en verzwakking/attenuation op UTP.

### 'INSERTION LOSS' EN 'ATTENUATION'

Van het vermogen dat in een transmissielijn wordt gepompt zal slechts een deel de ontvanger bereiken. Een deel ervan wordt dus gereflecteerd richting transmitter, een deel gedissipeerd in ohmse verliezen en een ander deel wordt uitgestraald, zeker bij hoge frekwenties (skin effect) → verzwakking of 'Attenuation'

### 1.2.3 COAX

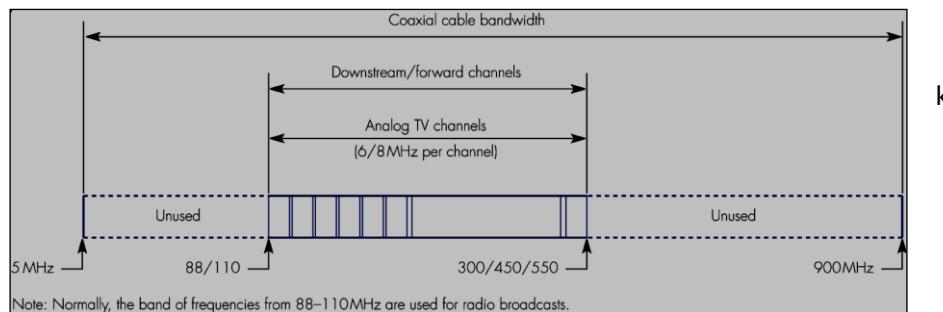
Zoals hierboven besproken zijn de beperkingen van UTP : zijn capacitieve koppeling en verzwakking door skin effect. Beide problemen worden beter opgelost in **coax** kabels. Bovendien wordt de elektromagnetische (EMI) uit- en instraling door de concentrische ligging van beide geleiders beperkt.



Figuur 62 Opbouw van een coax kabel.

De bandbreedte kan nu oplopen tot 800MHz (bv. kabel-TV) omwille van een veel betere ACR : verzwakking door skin-effect en ook EMI storing is lager.

Bv. TV distributie :  
door FDM (Frequency Division Multiplexing)  
men honderd TV kanalen + ... data op 1 kabel plaatsen.



Figuur 63 HFC coax bandbreedte

Op moderne HFC (Hybrid Coax Fibre) systemen worden onderaan (<88/110MHz) upstream-kanalen voorzien en boven de 600MHz de digitale video- en datakanalen voorzien.

(In het geval van digitale TV wordt op 1 frekwentieslot van 6MHz zelfs 4 SDTV, Standaard Definition TV, beelden gecomprimeerd wat het aantal mogelijke TV kanalen brengt tot enkele honderden.) Er moeten wel versterkers geplaatst worden om de km. Coax kabels hebben een diameter van 1cm tot 2,5cm, resp. soepele tot zeer stijve kabels.

Coax geraakt echter in onbruik door de relatief moeilijke verwerking en aansluiting.

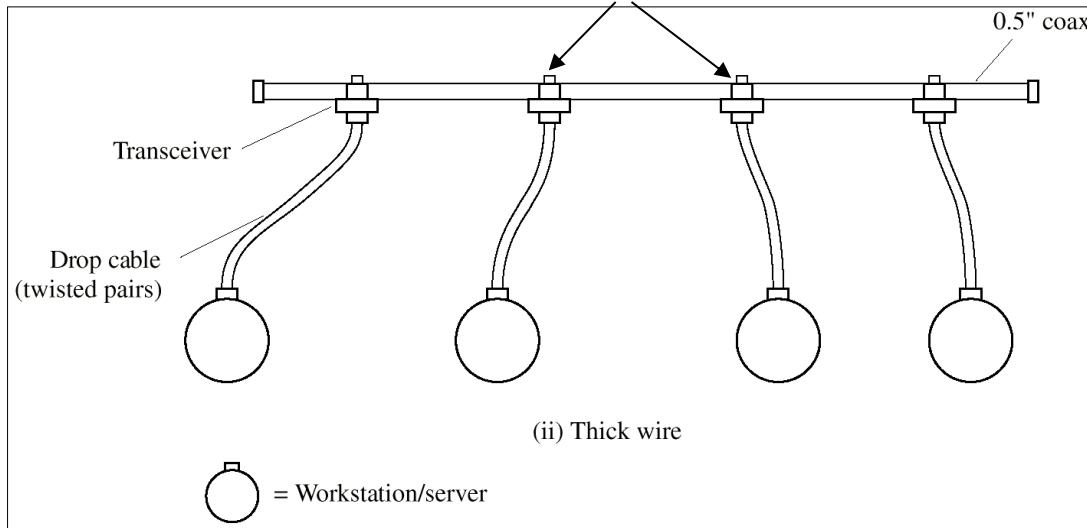
- PTT's en TV distributie verkiezen glasvezel voor veeleisende (nieuwe) verbindingen,
- LAN's → UTP wegens de beperkte afstand.

## GEBRUIK.

Oorspronkelijk zijn LAN's wél ontworpen op COAX. Ze vormden een **busstructuur** (zowel fysisch als logisch) en de signalen worden niet gemoduleerd → **basisband**. De karakteristieke impedantie van de coax is steeds  $Z_0=50\text{ohm}$ .

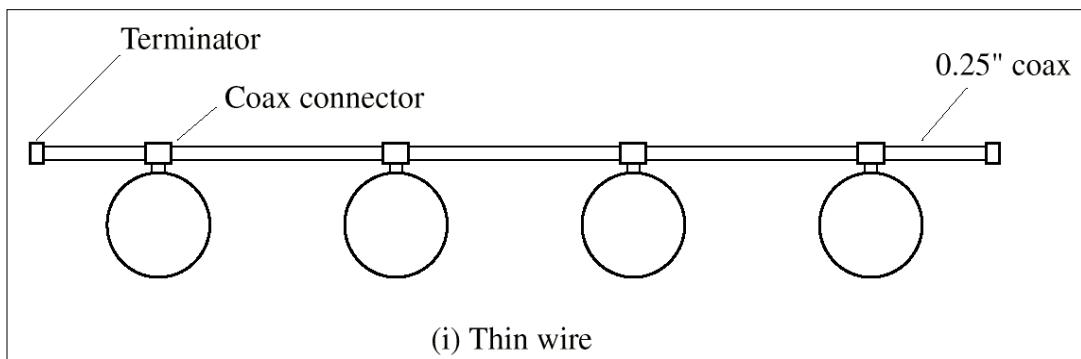
In historische volgorde :

- **10 base 5** : 'Thick ethernet'  $\phi 0.5"$ , 'Yellow cable' : **10Mbps over 500m** tss repeaters aftakking via "drop cable" van AUI (attachment unit interface)



Figuur 64 Praktische LAN-thick ethernet-bekabeling

- **10 base 2** : 'Thin ethernet'  $\phi 0.25"$  : **10Mbps over 200m** tss repeaters, max 5 segmenten, aftakking via "T-stuk", kort bij de soepele kabel die nu tot vlak achter de computer kan worden gebracht.  
M.a.w. AUI en drop cable zijn geïntegreerd op de netwerkkaart.



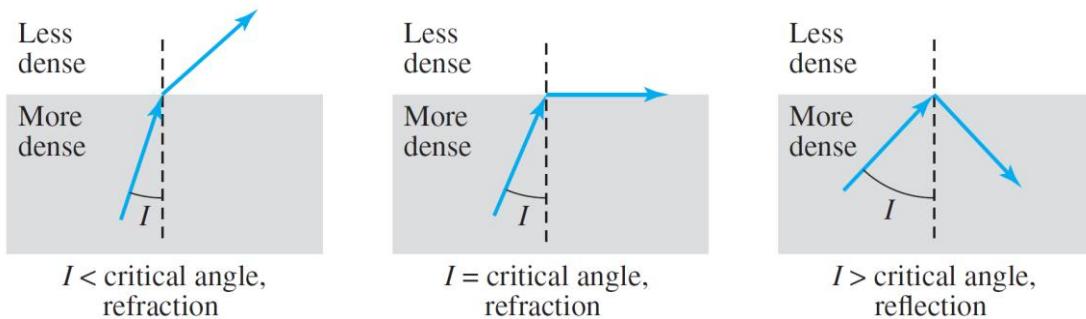
Figuur 65 Praktische LAN-thin ethernet-bekabeling

## 1.2.4 FIBER

De maximale frequentie en dus ook datasnelheid op koper is dus begrensd. Optische fibers hebben dat echter nog helemaal niet aangezien ze niet onderhevig zijn aan EMI en crosstalk.

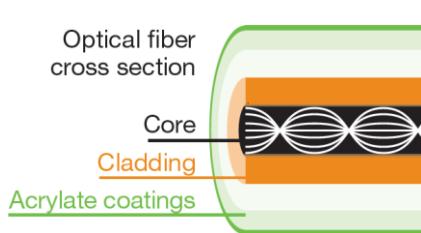
### WERKING

Ze worden gemaakt uit glas (soms plastic, industrie-automatiseringsomgevingen) waar een lichtbundel in wordt gestuurd. Deze lichtbundel reist in een rechte baan in een homogeen medium, maar bij de overgang naar een andere dichtheid van glas breekt de lichtbundel en verandert hij van richting.



Figuur 66 lichtrefractie in een fiber

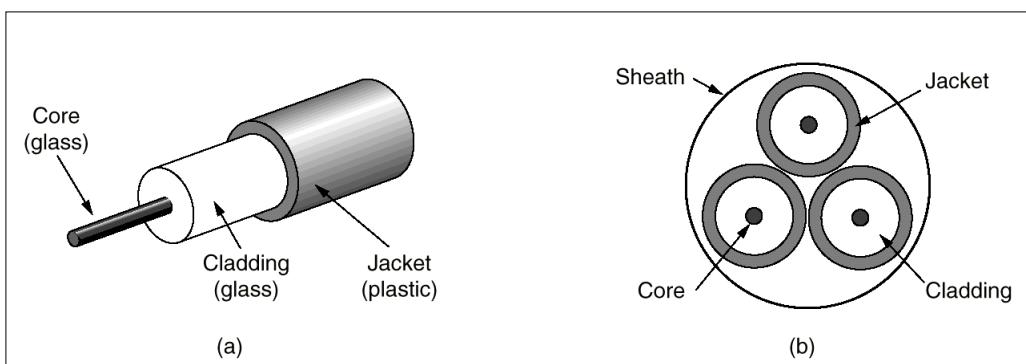
Zoals te zien is in bovenstaande figuur, als de invalshoek groter wordt dan de kritische hoek krijgen we een volledige weerkaatsing van het licht.



De fiber zelf bestaat dus uit 2 delen : een glazen **kern** (core) en een glazen **mantel** (cladding) met een lagere brekingsindex.

Stuur je het licht in de kern onder de kritische hoek zal er dus geen licht ontsnappen uit de kern. De verliezen hiervan zijn veel kleiner als degene die we kenden bij kopergeleiders. Zie ook Figuur 39 op p.50 ... 0,2dB/km !!

Figuur 67 Fiber doorsnede.



Figuur 68 Opbouw van een fiber

Een optische kabel bestaat uit een aantal (8, 12, ...) glas fibers, één voor elk te transporteren signaal per richting. Ze worden omhuld met een beschermende acrylaat coating (jacket) die bovendien beschermt tegen invallend licht en water.

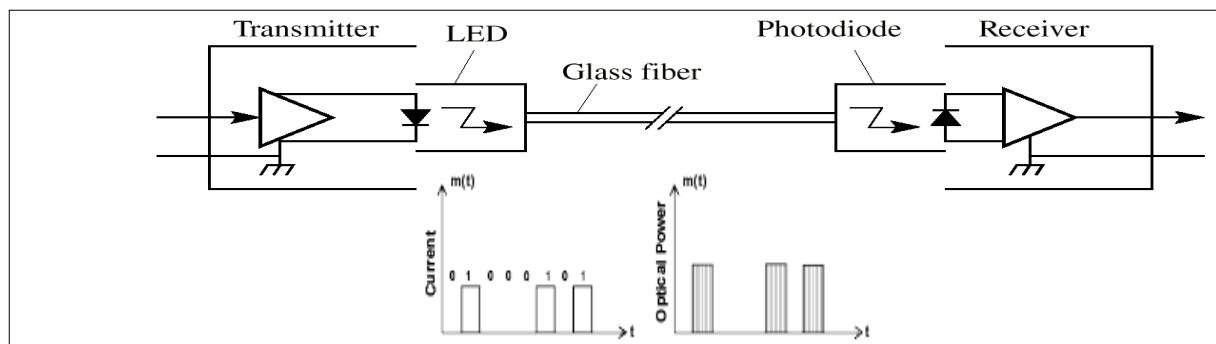
'Plastic fibers' zijn goedkoper en kunnen gebruikt worden voor korte afstanden. Ze worden omwille van hun immunitet veel gebruikt in fabrieksautomatisatie, maar dan meestal tegen veel lagere snelheden. Bovendien is hij fysisch moeilijk 'af te tappen' wat de beveiliging verhoogt.

Praktisch worden glasvezelkabels niet dieper dan 1m onder de grond gelegd om knaagdieren enz. te vermijden. Transoceanische kabels worden ofwel in goten ingegraven door een 'zeeploeg', of als er weinig kans is op vissers, haaien, ... los op de zeebodem gelegd.

Glasvezels worden op 3 manieren verbonden. In volgorde van lichtverlies en reflecties :

- via **connectoren** : 10 à 20% lichtverlies
- via een **mechanische las** : beide einden wordt !recht! afgesneden en in een huls tegen elkaar gelegd. De uitlijning wordt geoptimaliseerd door er een lichtbundel door te sturen.
- via een **smeltkoppeling** : de 2 fibers worden gesmolten en verbonden. Minimaal verlies.

### STURING



Figuur 69 Omzetting van elektrische signalen naar licht.

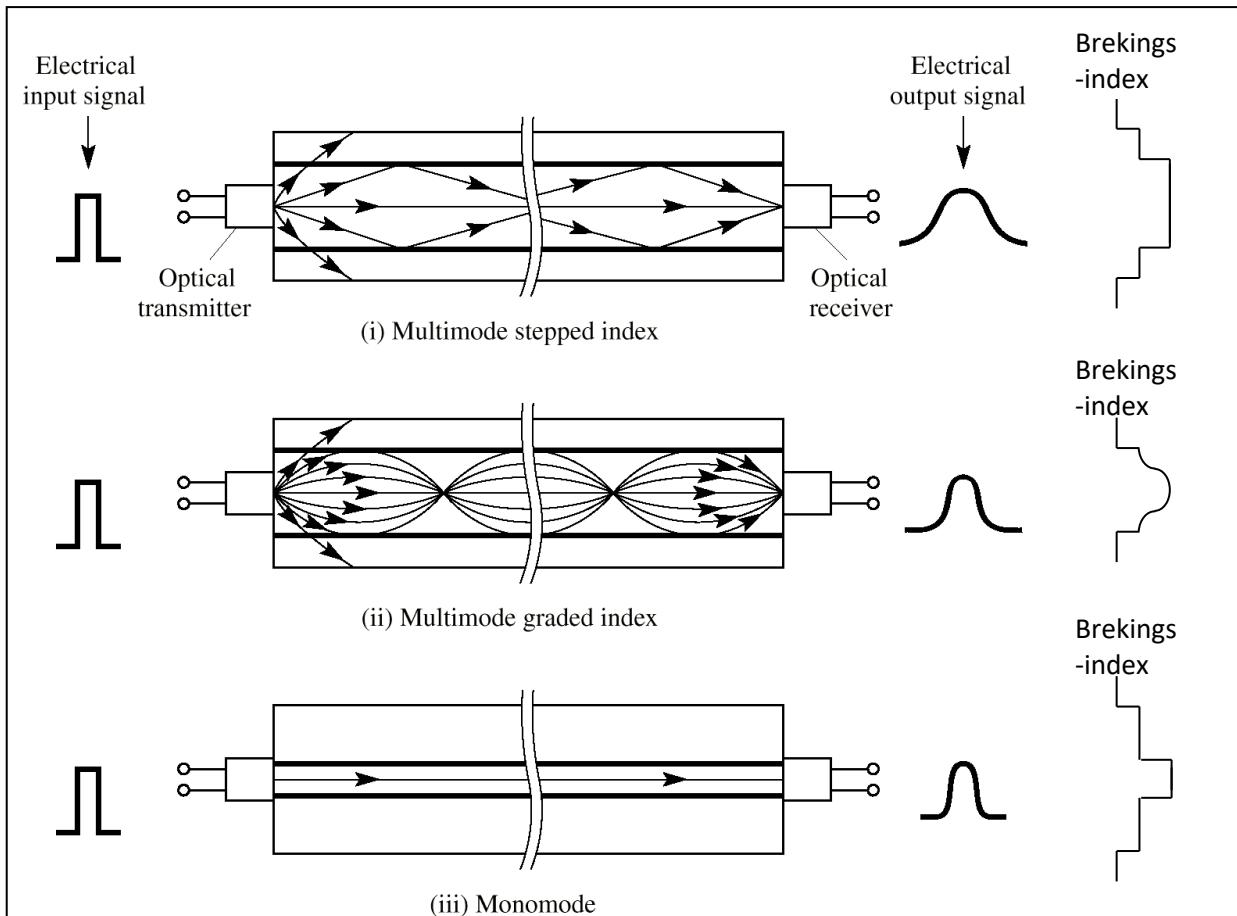
Elektrische signalen worden omgezet naar licht.

Bovenstaande figuur geeft een mogelijke codering van het lichtsignaal door 2 soorten lichtomzetters.

Andere coderingen zijn diegene die we ook voor elektrische signalen zullen zien: RZ, AMI, manchester, ... zie pt. 1.3.3 op p. 81

### TYPES GLASVEZEL :

Er zijn 3 hoofdtypen te onderscheiden :

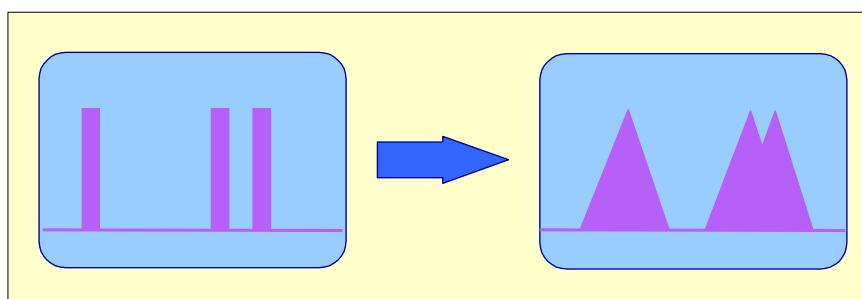


Figuur 70 Transmissiemodes in een fiber

#### 1. Multimode step index : kern/mantel diameter 62,5/125µm of beter 50/125µm

De kern heeft een constante brekingsindex. Afh. van de ingestuurde hoek legt het licht **verschillende afstanden af** → ‘multimode’ betekent dat het licht meerdere propagatiepaden heeft en de verschillende bundels dus met verschillende vertragingen bij de ontvanger toekomen.

Hierdoor **vervormt** de puls → dispersie. Men moet dan grotere ‘gaps’ laten tussen de pulsen om ze te kunnen herkennen door ISI (InterSymbol Interference) → verbreding puls leidt tot **beperkte bandbreedte**.



Figuur 71 Dispersie van pulsen in een multimode fiber

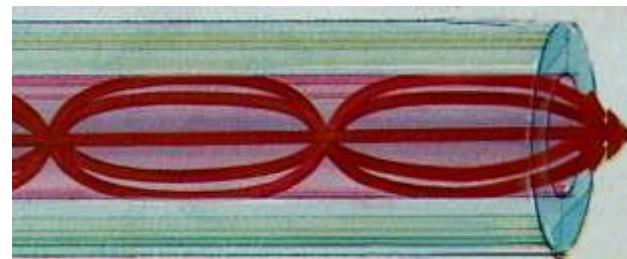
Kenmerken multimode :

- Brekingsindex van kern en mantel zijn uiteraard verschillend, doch **uniform constant**.
- oudere low cost vorm, tot max +/- 3km, bvb. ethernet tot 550m.
- Meestal polymeer of HCS vezels, gestuurd door goedkope LED's
- De oudste uitvoering
- Toegepast in LAN's.

## 2. Multimode graded index : kerndikte **50 – 62,5** µm. bvb. 62,5/125µm of 50/125µm .

Een eerste verbetering : de brekingsindex van de kern is **NIET constant** maar "graded". Hierdoor wordt het licht sterker gebroken, maar reist ook sneller, naarmate het zich verder van de kern bevindt. De langere afgelegde weg wordt gecompenseerd door een hogere snelheid waardoor de puls smaller wordt → hogere bandbreedte.

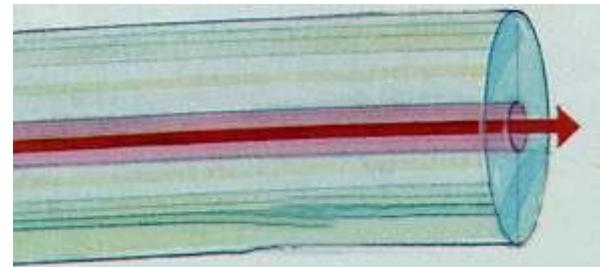
- Meest gebruikte low cost vorm.
- Uitvoering in glas.
- Toegepast in LAN's



## 3. Monomode step index : kerndikte **3 – 5 – 8 - 10** µm. bvb. 8/125µm

Dit type fiber wordt ook wel 'single' mode fiber genoemd. SMF.

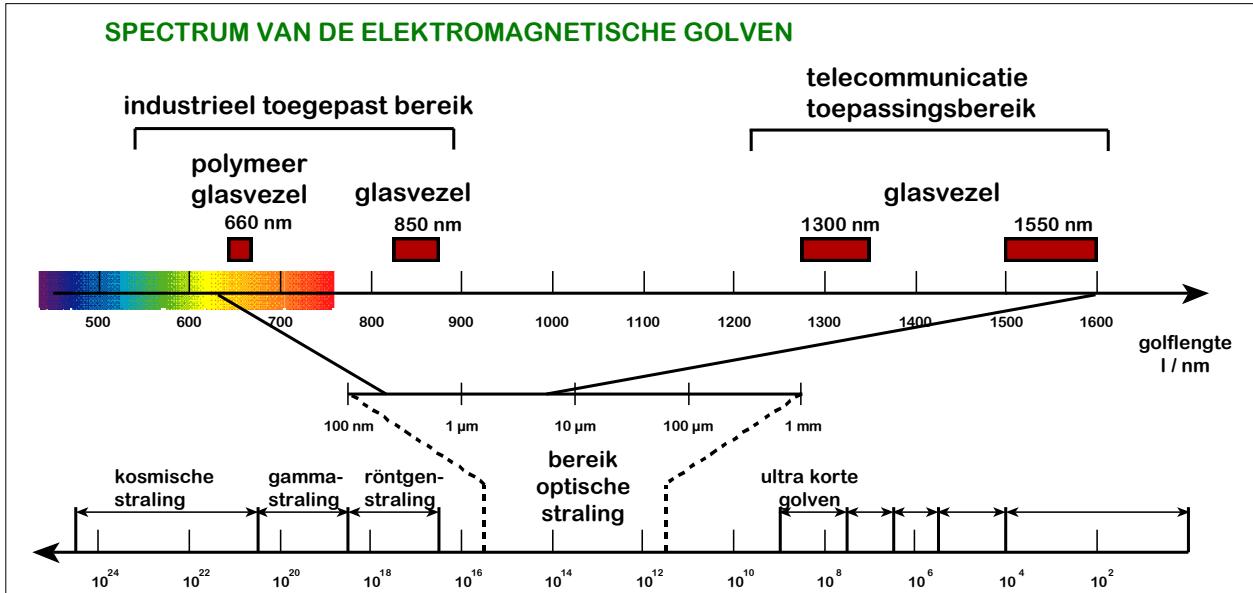
Een laatste verbetering : de brekingsindex van de kern en mantel is terug constant → "step", maar de kern is ZO **smal** dat het verschil in afgelegde weg geminimaliseerd wordt. De **kerndiameter** heeft dezelfde **grootteorde** als de **golflengte** van het licht : 8µm tegen 1550nm=1,5µm (1,3µm)!



De sturing in een monomode fiber gebeurt door **laser** (LD) diodes, zodanig dat het monochrome, coherente (zelfde fase), sterk gebundelde licht dezelfde weg aflegt.

- hoogste bandbreedte
- Zeer moeilijk aansluiting
- gebruikt voor (zeer) lange afstanden / intercontinentaal verkeer.

## SPECTRUM



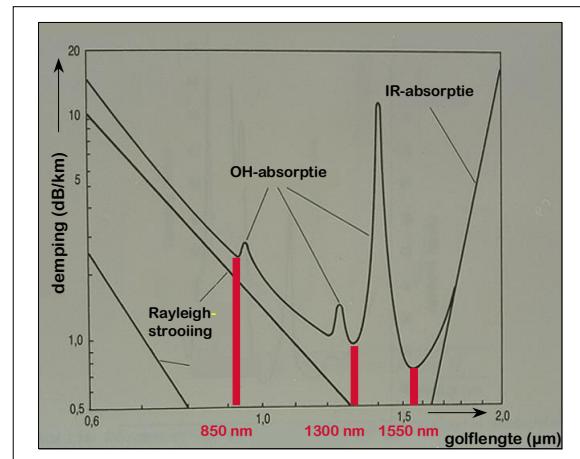
Figuur 72 spectrum van de gebruikte EM golven door glasvezels

We zien hier de gebruikte golflengtes (frekventies) van het licht. Het licht door een polymeervezel (minder belangrijk) ligt in het zichtbaar spectrum, voor glasvezel ligt het erbuiten.

Glasvezel is uit ... glas gemaakt. In de renaissance mocht glas max. 1mm dik zijn om er nog door te kijken, nu zou men door een glasvezel gemakkelijk de bodem van de oceaan zien... moet zijn doorlaatkarakteristiek in het zichtbaar licht liggen. De verzwakking is echter afh. v. de golflengte.

De demping van een glasvezel kent **3 minima** :

- 850 nm : 3 dB/km
- 1300nm : 0.5 dB/km
- 1550 nm **0.3dB/km** → veel langere afstanden! → > 100km



Figuur 73 Dempingsverloop van een glasvezel

Het voordeel aan **850nm** is dat het licht kan opgewekt worden door IR-LED's i.p.v. lasers.

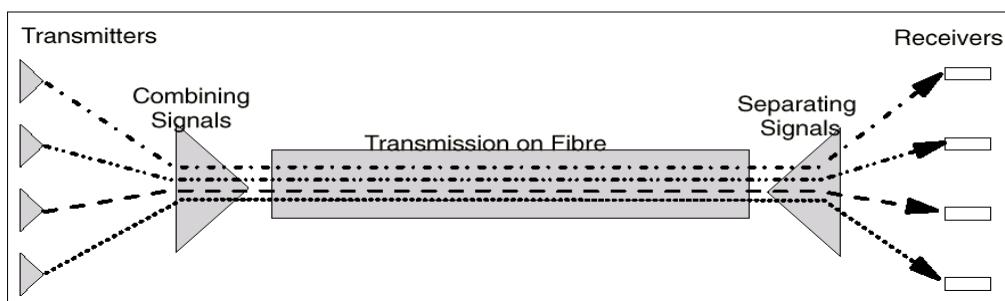
De transmitters bestaan dus afh. v. h. vereiste spectrum uit LED's of (Injection) Laser Diodes (iLD), beiden halfgeleiders. De resp. ontvangers zijn fotodiodes of fototransistoren. LED's zijn goedkoper, hebben een groter temp.bereik en 'leven langer' dan iLD's. De meeste 'local' fibers worden gestuurd door 850nm LED's. Voor grotere afstanden worden laserdiodes gebruikt.

De lengte van de lichtimpulsen die door de glasvezel worden geleid worden tijdens de voortplanting gespreid. Deze **strooiling** of **dispersie** is ook afh.v. de golflengte en minimaal vanaf 1300nm → hogere snelheden én langere afstanden, maar lasers nodig → prijs ↗

Ter info: de demping van een **polymeer**-vezel kent een minimum op **660nm** : ong. 200 db/km. Dit is relatief veel vandaar dat dit enkel gebruikt wordt voor korte afstanden! : < **100m**, in fabrieksautomatisatie voor zijn storingsongevoeligheid.

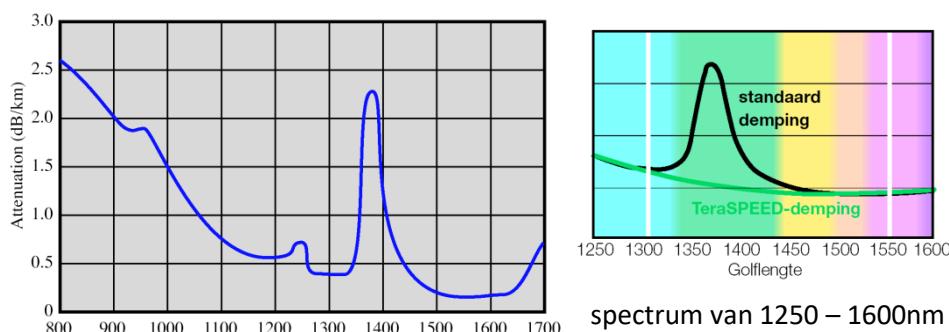
### WAVELENGTH-DIVISION MULTIPLEXING.

Tot nu toe hebben we enkel beschouwd dat de toegepaste lichtbronnen slechts **1 kleur** licht uit zenden, op een golflengte met de minste demping. Indien men in dit geval meerdere informatiekanalen wenst over te zenden moet men multiplexen door **TDM**, Time Division Multiplexing.



Figuur 74 Principe van Wave Division Multiplexing.

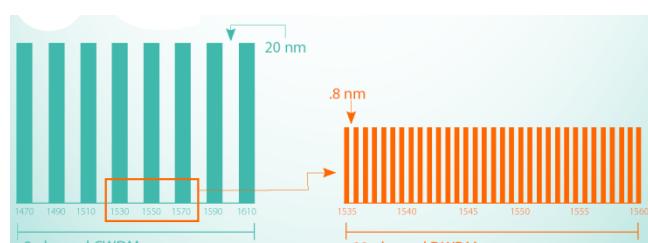
De laatste evoluties spitsen zich toe op het gebruik van meerdere golflengtes tegelijk op 1 fiber → **WDM** : Wave Division Multiplexing. Om het licht op de fiber te krijgen, moet met prisma's gewerkt worden. (meestal in het 1500nm bereik omdat dit breder is).



Figuur 75 Verzwakking in een fiber ifv.  $\lambda$ .

Watermoleculen zijn onreinheden die zorgen voor 'scattering' en absorptie.

Er zijn ruwweg 2 soorten WDM : DWDM (Dense -) en CWDM (Coarse-). De laatste is een goedkoper alternatief dat meer tolerantie verdraagt in de zender en ontvangers.



DWDM hanteert een spacing van **0,8nm** per kleur/kanaal waar CWDM gestandardiseerd is door ITU op 20nm.

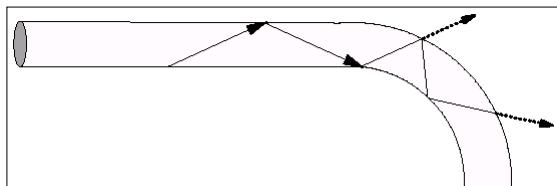
Figuur 76 DWDM vs CWDM spacing

Sommige fiberfabrikanten slagen erin om ZWP (Zero water peak) SMF te fabriceren die de verzwakkingspiek bij 1400nm elimineert en zodoende het volledige spectrum van 1250 – 1600nm bruikbaar maakt.

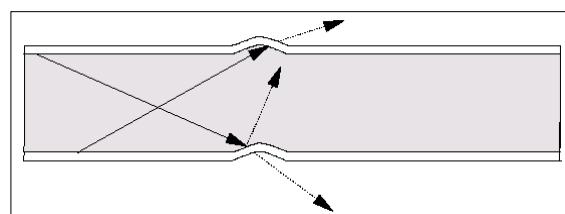
## FIBER PROBLEEMEN

### BOCHTEN EN ONZUIVERHEDEN

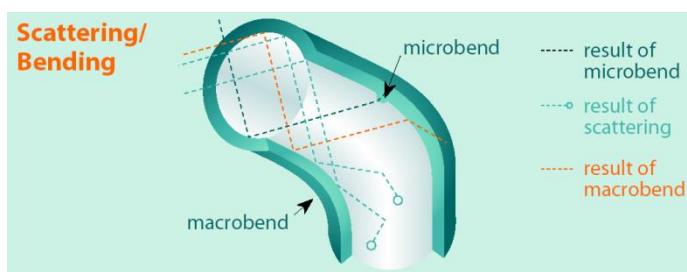
Bij de installatie moet nauwlettend in de gaten worden gehouden dat de fiber geen te scherpe bochten ('bends') vertoont. Zoniet zou de kritische hoek overschreden worden en wordt er licht verloren. Dit licht



komt terecht in de mantel en kan in het slechtste geval aankomen in de ontvanger, weliswaar met een verschillende snelheid, wat resulteert in dispersie van het signaal. Een bocht van  $180^\circ$  zou minimum 10cm diameter moeten hebben (2 à 3 cm is kritisch).



Micro-bochten ontstaan wanneer de fiber tegen een onregelmatig oppervlak wordt gedrukt. Ook hier verliest de fiber licht. Soms (bv. in labs) wordt dit zelfs doelbewust gedaan om 'hogere orde modes' en 'cladding modes' te verliezen.



'scattering' zijn onzuiverheden aanwezig in de fiber door variatie in densiteit en samenstelling van het glas.

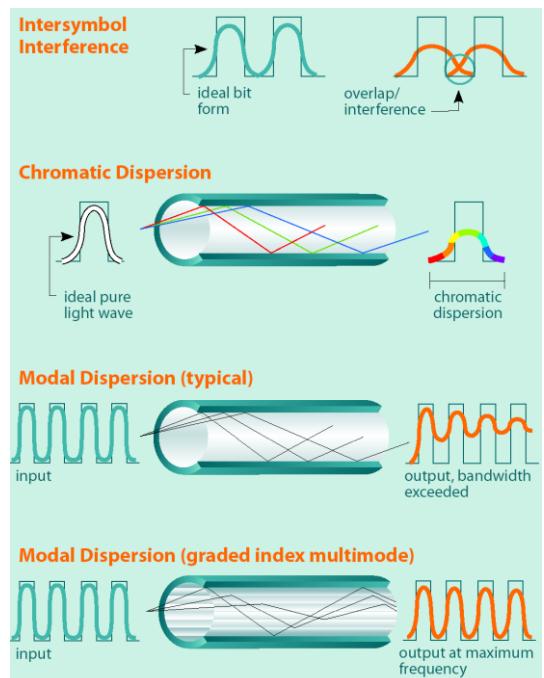
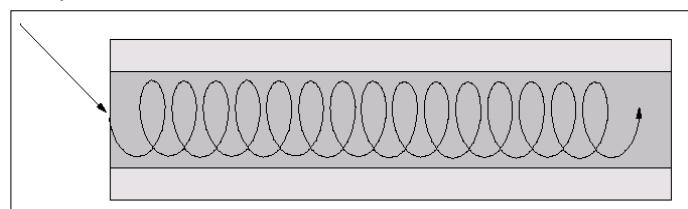
Figuur 77 Bochten en onzuiverheden in een fiber.

### DISPERSIE IN EEN FIBER.

Om ISI te vermijden moet vooral de dispersie tot een minimum beperkt worden. Chromatic dispersie treedt vooral op bij 850nm bronnen (LEDs). Lasers minimaliseren dit door monochrome licht te sturen. Dit is een belangrijke oorzaak van de beperkte afstand die 850nm systemen kunnen overbruggen (<550m).

Modale dispersie heeft als oorzaak dat de verschillende lichtbundels een andere weg afleggen.

Men moet inderdaad het 3 dimensioneel karakter van de fiber en de propagatie van het licht in rekening brengen in Figuur 70. In dit 'licht' kunnen de afgelegde afstanden sterk verschillen voor de multimodefibers. Graded indexen zullen dit compenseren.



Figuur 78 dispersie in een fiber.

Figuur 79 Spirale propagatie van licht in een multimodefiber.

## FIBER TOEPASSINGEN

Fibers kennen een aantal voordelen en nadelen voor gebruik in telecommunicatie.

### Voordelen :

- Zeer hoge transmissiesnelheden : 40 ... 100 ... Gbps. Momenteel zijn het vooral de transceivers die hier een beperkende factor vormen, i.t.t. het medium zelf.
- Zeer lage verzwakking of ‘attenuation’: gevoelig lager dan koper : bv. 0,2-0,5 dB/km tegen 10-tallen dB/100m voor UTP op de gebruikte frekwenties → te gebruiken voor langere afstanden
- volledige elektromagnetische immuniteit : geen storingen van motoren (impulseruis), noch crosstalk.
- volkomen elektrisch geïsoleerd : er zijn geen potentiaalkoppelingen.
- Licht gewicht.

### Nadelen :

- Installatie is duur, ook lassen en repareren.
- Gevoelig voor VOCHT!

## 10G ETHERNET STANDAARDEN.

De vermelde afstanden hierboven zijn voor de 1Gbps of 1000base... standaard. Bij 10G ethernet zijn de afstanden en mogelijkheden gewijzigd. We zien wel duidelijk dat lange afstanden gereserveerd zijn voor OS1/2 of Single Mode Fiber SMF.

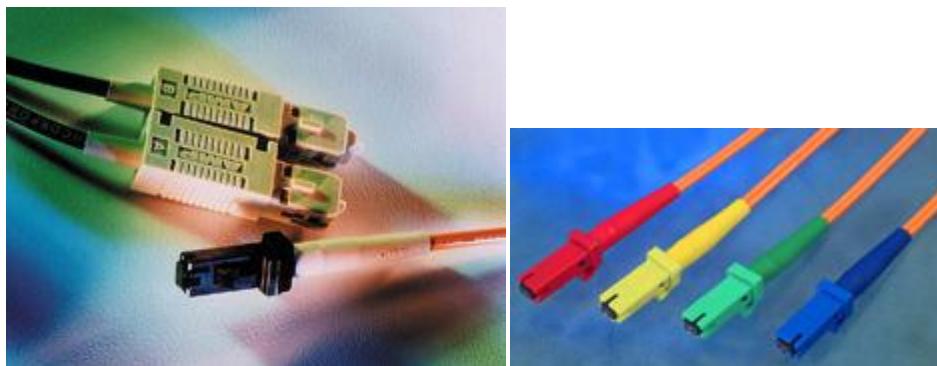
Electronics	Media Type	Maximum Distance	Est. Cost
FIBER	OM1 MMF 62.5 µm	0 m	\$\$
	OM2 MMF 50 µm	100 m	
	OM3 MMF 50 µm	200 m	
	OS1/2 SMF	300 m	
1310 nm CWDM 10GBASE-LX4	OM1 MMF 62.5 µm		\$\$\$\$
	OM2 MMF 50 µm		
	OM3 MMF 50 µm		
	OS1/2 SMF	10 km	
1310 nm Serial 10GBASE-LR 10GBASE-LW	OM1 MMF 62.5 µm		\$\$\$
	OM2 MMF 50 µm		
	OM3 MMF 50 µm		
	OS1/2 SMF	10 km	
1550 nm Serial 10GBASE-ER 10GBASE-EW	OM1 MMF 62.5 µm		\$\$\$\$\$
	OM2 MMF 50 µm		
	OM3 MMF 50 µm		
	OS1/2 SMF	40 km	
1310 nm Serial (EDC) 10GBASE-LRM	OM1 MMF 62.5 µm		\$\$\$*
	OM2 MMF 50 µm		
COPPER	OM3 MMF 50 µm		\$*
	OS1/2 SMF		
10GBASE-T	Twisted Pair		\$*
10GBASE-CX4	Twinax		\$\$

\*cost projection

Figuur 80 vergelijkende tabel voor 10G ethernet.

## FIBER CONNECTOREN

In volgende tabel vindt u de meest gebruikte fiber connectoren. In het lab vindt je meestal de SC connectoren, met op de nieuwste toestellen de zeer compacte MT-RJ connector.



Figuur 81 Fiber SC en MT-RJ connectoren

Voor lange afstanden en op patchpanelen word dikwijls gebruik gemaakt van ST connectoren. Allen aanwezig in het lab.

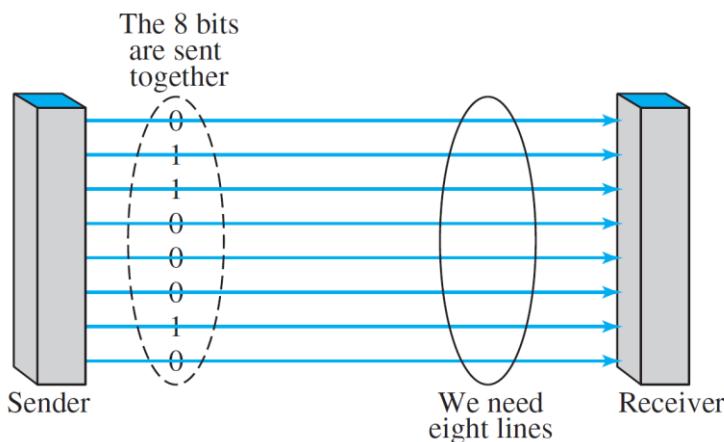
Connector	Insertion Loss	Repeatability	Fiber Type	Applications
FC	0.50-1.00 dB	0.20 dB	SM, MM	Datacom, Telecommunications
FDDI	0.20-0.70 dB	0.20 dB	SM, MM	Fiber Optic Network
LC	0.15 db (SM) 0.10 dB (MM)	0.2 dB	SM, MM	High Density Interconnection
MT Array	0.30-1.00 dB	0.25 dB	SM, MM	High Density Interconnection
SC	0.20-0.45 dB	0.10 dB	SM, MM	Datacom
SC Duplex	0.20-0.45 dB	0.10 dB	SM, MM	Datacom
ST	Typ. 0.40 dB (SM) Typ. 0.50 dB (MM)	Typ. 0.40 dB (SM) Typ. 0.20 dB (MM)	SM, MM	Inter-/Intra-Building, Security, Navy
<b>Installing Fiber Optic Connectors</b>				

## 1.3 DATATRANSMISSIE

We beschouwen hier enkel de transmissie in **digitale** signalen uit pt. 1.1.3 Analog en digitale transmissie. van p. 40. De transmissie van analoge signalen wordt verwezen naar de cursus telecommunicatie.

### 1.3.1 PARALLELLE TRANSMISSIONSMETHODES

In dit werk wordt enkel (op dit stukje na) gesproken over **seriële** communicatie, niet over parallelle.



Principieel is het simpel: door meerdere draden tegelijk te gebruiken kan men (op het eerste zicht) hogere transmissiecapaciteiten krijgen. Bv. in deze figuur een byte per 'clock-tick'.

Figuur 82. Parallelle transmissie.

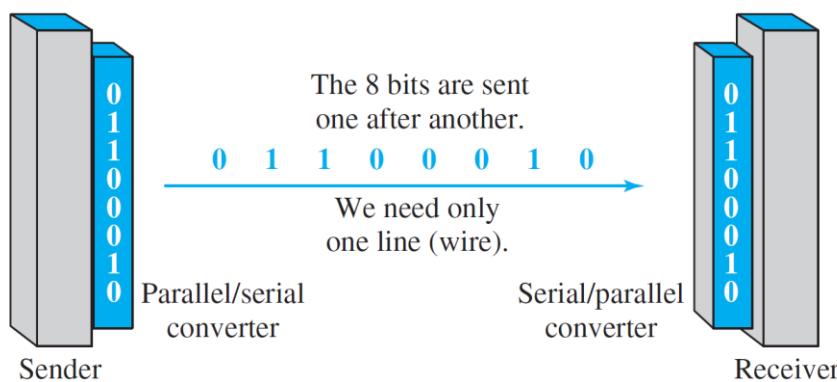
In de praktijk heeft men ondertussen geleerd dat dit zeker niet altijd winstgevend is: de nabijgelegen draden beïnvloeden elkaar (crosstalk, zie ook pt. 1.2.2 p. 55 e.v.), waardoor de signaling rate afneemt. Dit is zeker zo voor langere afstanden. Zo heeft men typisch de evolutie gekend voor harddisk drive aansluitingen van PATA ➔ SATA interfaces: men kan sneller de data oversturen door een serieel kanaal.

Min of meer de enige plaats waar men anno 2018 nog parallelle transmissie kent is op een printplaat, bv. een motherboard van een PC, waar de data- en adress-bussen bestaan uit naast elkaar gelegen banen. Zeer korte afstanden dus. Ook PCIe interfaces op PC-motherboards kennen een parallelle structuur.

Maar zelfs daar, op printplaten voor de communicatie tussen chips, zullen de meeste connecties nu ook al serieel uitgevoerd worden!

Een andere situatie waar parallelle communicatie voorkomt is bv. vanaf de gigabit-ethernet standaard. Om dit toe te kunnen passen op de bestaande CAT5 kabel heeft men besloten om de 4 aanwezige UTP paren tegelijk te gebruiken, i.p.v. 1 paar per richting. Gevolg was dat de grootste problemen kwamen van de crosstalk: NEXT en FEXT. Zie ook pt. cross talk : NEXT en FEXT bij 1000base/10GBASE op p. 59.

### 1.3.2 SERIËLE TRANSMISSIEMETHODES



Seriele transmissie is eenvoudig: de bits volgen elkaar op op het kanaal.

Figuur 83 Seriele transmissie.

Data wordt, ook serieel, meestal overgezonden tussen 2 DTE's in vaste lengte eenheden van **8 bits**. Een boodschap kan dan bestaan uit een blok van dergelijke bytes en wordt meestal **frame** genoemd.

Als de datacommunicatie vlekkeloos moet verlopen dient de ontvanger het volgende te onderscheiden :

- 1) de start van elke **bit** cel (om in het midden hiervan te kunnen sampelen)
- 2) de start van elk element (karakter of **byte**)
- 3) de start van elk **frame**

Deze 3 taken worden ook resp.

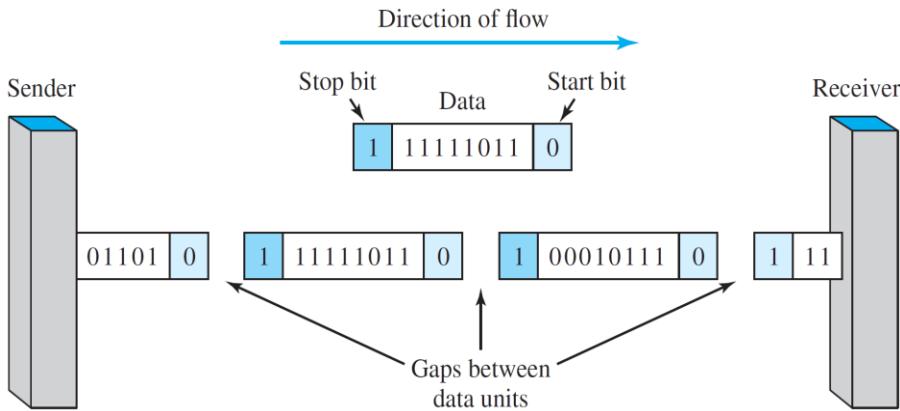
- 1) **bit- of clock-synchronisatie** (1)
- 2) **karakter- of byte- synchronisatie** (2) en
- 3) **block- of frame-synchronisatie**.

De oplossingen voor bovenstaande taken worden op 2 verschillende manieren opgelost: **asynchrone** of **synchrone transmissie**. Afh. of de zender- en ontvanger-klokken met elkaar gesynchroniseerd zijn, of onafhankelijk zijn van elkaar.

- **Asynchrone** transmissie : de ontvanger synchroniseert bij het begin van elke **byte**, zie RS232, en heeft dus **geen bitsynchronisatie**.
- **Synchrone** transmissie : de ontvanger synchroniseert bij het begin van elk **frame** en moet in de pas blijven gedurende dit hele frame. Hij ziet m.a.w. een **continue bitstroom** gedurende de hele periode van het frame. Er moet nu een mechanisme toegevoegd worden om bit (en byte) synchronisatie te verkrijgen.

## ASYNCHRONE TRANSMISSIE

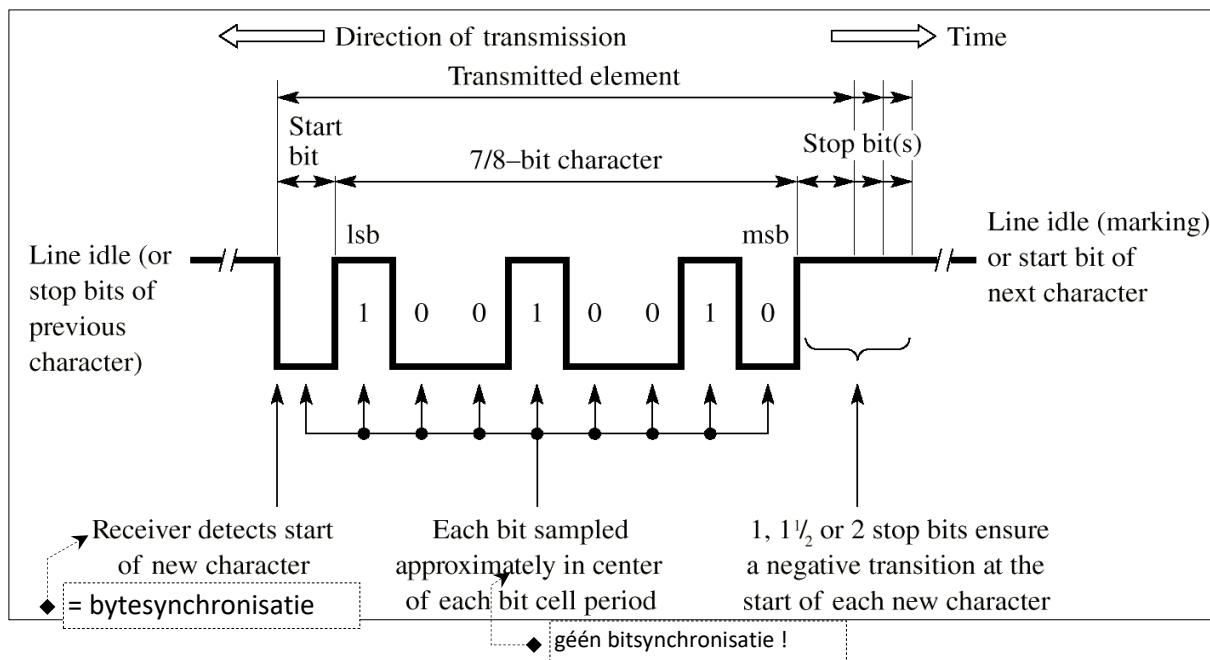
Deze methode wordt gebruikt in de oudere standaarden zoals RS232, RS422, RS485, die nog veelvuldig worden gebruikt voor verbindingen met een lage capaciteit en omwille van de grote verspreiding een vermelding waard zijn. Bv. moderne routers komen nog steeds met een CONsole connector op RS232.



Figuur 84 Asynchrone seriele transmissie.

Bv. de verbinding van een toetsenbord met een computer: de gebruiker tikt en de ascii-code van de aangeslagen toets wordt opgestuurd met onregelmatige tussenpozen. Maar het kan ook gebruikt worden om bv. een tekstbestand (bv. router config) door te sturen waarbij de bytes wel in blok ter beschikking zijn en dus doorgestuurd worden zonder 'gaps'.

Wanneer data met een 'random' tijdsinterval wordt overgezonden en er geen synchronisatie is tussen de zender- en ontvangerklokken, dan moet de ontvanger op elk **karakter** synchroniseren. Dit gebeurt door de byte te laten voorafgaan door 1 startbit, en te laten volgen met (minstens) 1 stopbit. Is er geen nieuwe byte beschikbaar dan blijft dit niveau (=stopbit) behouden: 'idle'.



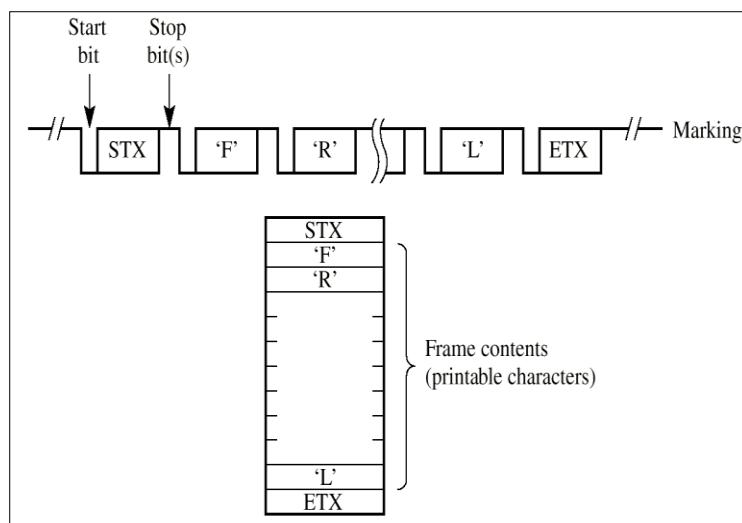
Figuur 85 Asynchrone transmissie.

Merk op dat de 'direction of transmission (of flow)' in Figuur 85 tegengesteld is aan die van Figuur 84.

Nogmaals, er is **géén bitsynchronisatie** → ‘A-synchroon’. Zender en ontvanger dienen met dezelfde parameters te worden ingesteld : baudrate, #databits, pariteit en #stopbits zodat ze hetzelfde patroon verwachten.

De ontvanger detecteert de ‘startflank’ waarop hij, afh. v. zijn instellingen, de volgende bits gaan sampelen bij voorkeur in het midden van de bittijd. Is hij (een beetje) uit synchronisatie, dan zal bij de volgende stop-start flank de ontvangstklok gesynchroniseerd worden → **byte synchronisatie**.

## FRAME-SYNCHRONISATIE



Als er een **blok** karakters wordt overgezonden volgt de volgende startbit onmiddellijk op de stopbit.

Figuur 86 frame synchronisatie bij asynchrone transmissie.

Normaal moet er dan voor **frame**-synchronisatie gezorgd worden. Hierbij wordt het blok karakters tussen 2 speciale **controlekarakters** geplaatst : **STX** (start of text) en **ETX** (end of text). Als de ‘echte’ data enkel uit ‘printable’ karakters bestaat is er geen probleem : STX en ETX komen **niet** voor temidden van het frame.

Wordt er echter **data** overgezonden dan is het **wél** mogelijk dat STX en ETX middenin het frame voorkomen. Hiervoor moet een oplossing gezocht worden ... en gevonden in laag 2 van het OSI model, de datalink-laag. Zie **transparantie** in dat hoofdstuk.

## SYNCHRONE TRANSMISSIONE.

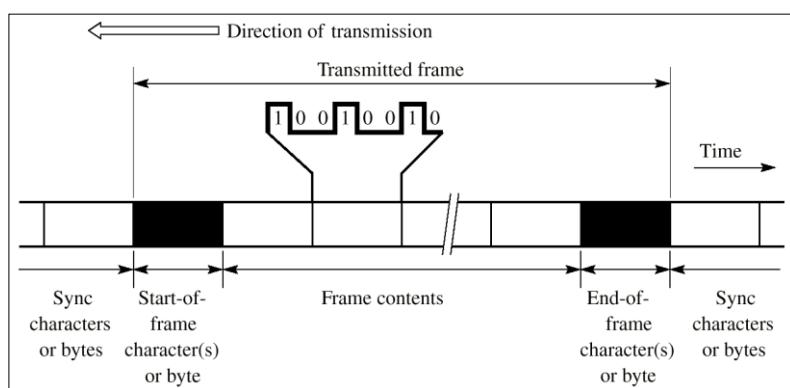
Voor hogere snelheden moet er een **synchronisatiemechanisme** toegevoegd worden → synchrone transmissie. Er wordt niet meer per karakter gesynchroniseerd → **start en stopbit per byte verdwijnen**. Er is geen scheiding meer tussen de laatste bit van vorig karakter en de eerste van het volgende → een continue bitstroom. Het is de taak van de ontvanger om de bits te groeperen en bytes en frames te onderscheiden. De verticale onderverdelingen in onderstaande Figuur 87 staan er dus niet echt.

We maken een onderscheid in de lijnconfiguratie tussen:

- **P2P** (Point to Point) verbindingen die op het medium slechts 1 zender en 1ontvanger hebben
- **MAC** (Multiple ACcess) verbindingen waar meerdere zender/ontvangers op het medium aanwezig zijn.

## LIJNCONFIGURATIE

### P2P.

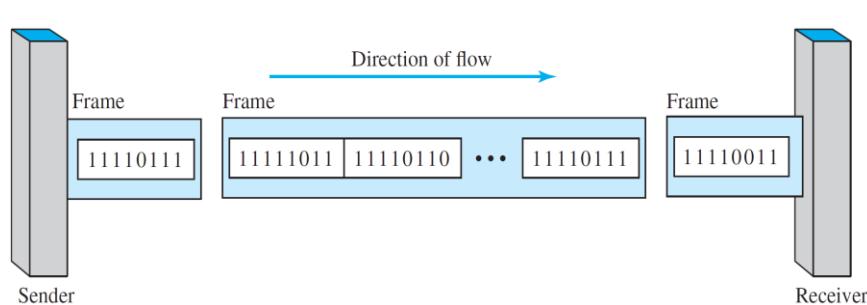


In de 'idle' toestand tussen 2 frames worden continu 'idle' karakters gezonden om de ontvanger toe te laten bit- en byte- synchronisatie te behouden

Figuur 87 framesynchronisatie bij synchrone P2P transmissie.

### MAC.

Aangezien het medium nu 'verdeeld' moet worden zal een zender, na dat het zijn frame heeft verzonden, het medium verlaten zodat een andere partij een boodschap kan zenden. Er zit dus een 'interframe gap' tussen de frames. (in onderstaande figuur kan per frame de 'flow'- richting omkeren).



Er worden dan eerst een aantal 'sync' bytes (8 of meer) gestuurd om de ontvanger(s) in synchronisatie te krijgen  
(zie ook Figuur 105 op p.91)

Figuur 88 framesynchronisatie bij synchrone MAC transmissie.

**SOF** en **EOF** duiden resp. **begin** en **einde** van het frame aan (Figuur 87). Natuurlijk moet ook hier de **transparantie** verzorgd worden : **SOF** en **EOF** karakters moeten onderscheiden worden van evt. identieke databytes, zie hoofdstuk datalinklaag.

## BANDBREEDTE.

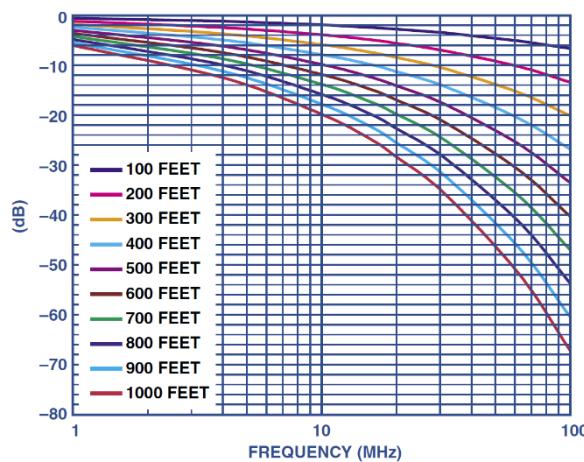
Synchrone transmissie wordt zowel toegepast in **LAN**- als in **WAN**-omgevingen, weliswaar met een **ander synchronisatiemechanisme** omwille van de specifieke problemen in beide situaties.

Laat ons de vergelijking maken op basis van UTP, het meest populaire kopermedium in beide omgevingen.

We kennen reeds de belangrijkste parameter van UTP: de bruikbare **bandbreedte** uitgetest ook via ACR.

Zie pt. ACR: Attenuation to Crosstalk Ratio : op p.56.

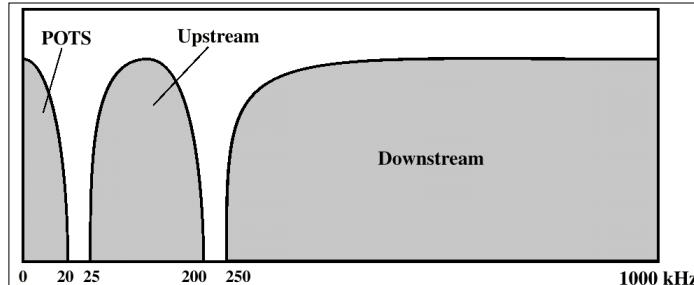
## LAN BANDBREEDTE.



UTP voor **LAN's** kent bandbreedtes vanaf **100MHz** tot ... **1000MHz**, zie UTP pt. Uitvoeringen / CATEGORIEN op p.53.

We hernemen hiernaast Figuur 27 van p.38. CAT 5 kabel heeft op 300 feet, 100m een verzwakking van -20dB.

## WAN BANDBREEDTE.



UTP voor **WAN's** heeft een veel lagere bandbreedte wegens de typisch veel langere afstanden (km's ipv. 100m) + de lagere kwaliteit: twist afstanden van 7cm ipv. 0,5cm. Een typische bruikbare bandbreedte voor **WAN UTP** is dan **1MHz**.

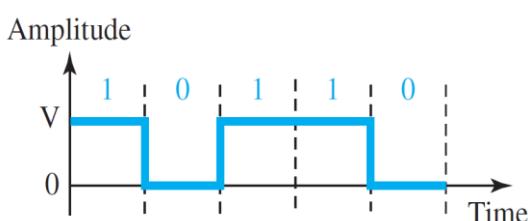
Zie daarvoor Figuur 39 op p. 50 of Figuur 158 op p. 126. Die we hierboven hernemen.

Het gevolg is dat **LAN**-omgevingen **veel meer bandbreedte** ter beschikking hebben dan **WAN** omgevingen.

### DC WANDER OF BASELINE WANDER

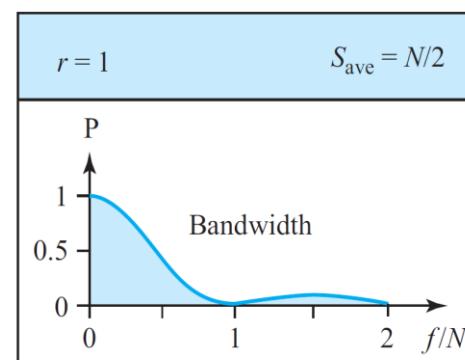
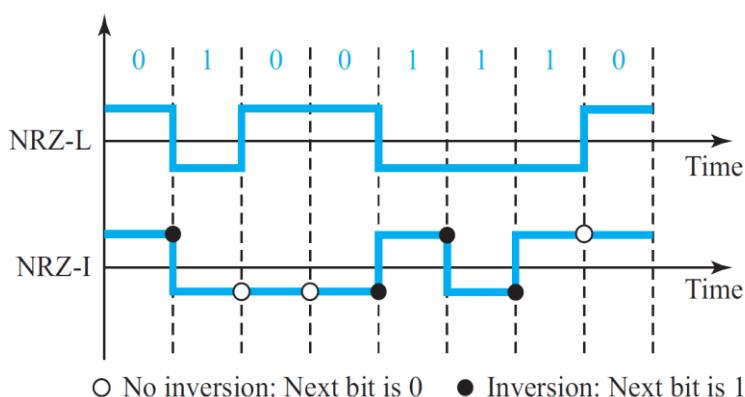
Een ander probleem dat typisch in WAN omgevingen moet bestreden worden is het 'opladen' van het medium. 2 geleiders, gescheiden door een isolator, vormt een capaciteit die belangrijke dimensies kan aannemen bij langere verbindingen, WANs.

Om een digitaal signaal te decoderen berekent een ontvanger een gemiddelde spanning van het binnenkomend signaal : de '**baseline**'. Hij zal dan de signaalelementen 'sampelen' tov. deze baseline.



Bij sommige lijncoderingen, bv. unipolaire NRZ (Non Return to Zero), zal bij een lange sequentie van bv. 1 –en de lijn opladen waardoor het moeilijk wordt om nog andere signaalelementen te onderscheiden.

Figuur 89 een unipolaire NRZ lijncode.



Figuur 90 een polaire NRZ lijncode.

Een oplossing kan zijn om een polaire lijncode NRZ-L (-Level) te gebruiken die + en - spanningen kent langs beide zijden van de as. (hier 0 = +V, 1 = -V).

Of een NRZ-I (-Invert) code die een 1 codeert als een overgang bij het begin van de bittijd en een 0 als GEEN overgang bij het begin van de bittijd.

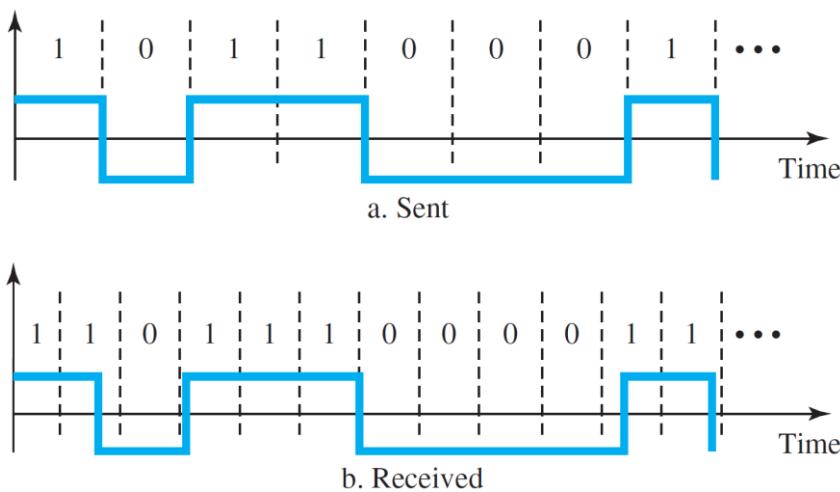
We hebben hier meteen een '**normalised bandwidth**' powerspectrum (fourier) bij geplaatst voor de eerste beperking van het medium. We zien dat het merendeel van het spectrum ligt voor de 1 wat betekent dat als je deze codes gebruikt voor een baudrate  $N$  (hier = datarate) van 1Mbps, je een bandbreedte  $f$  nodig hebt van 1MHz.

(Eigenlijk zie je een kleine component tot 2 MHz, maar bij gebrek hieraan zal het signaal licht vervormen).

We zien ook dat deze NRZ lijncodes een grote DC component hebben (bij contiguous 1's of 0's voor NRZ-L en bij contiguous 0's voor NRZ-I) wat de DC wander niet ten goede komt. Ook de (bit-)synchronisatie is een probleem in dit geval! We zullen dus schema's zien waarbij we vermijden dat dit gebeurt.

## BITSYNCHRONISATIE

Uiteraard blijft het belangrijkste probleem de synchronisatie. Systemen die niet gesynchroniseerd zijn geven volledig foute bitstromen bij hun ontvanger.



Figuur 91 foute synchronisatie bij digitale transmissie.

De manier waarop de LAN- of WAN- omgevingen dit oplossen verschilt nu omwille van hun specifieke kenmerken bij bovenstaande problemen:

- ↳ **LAN's** : klokcodering, de klok wordt 'gemoduleerd' op de data.  
(als er voldoende bandbreedte voorradig is op het medium ...  $f \nearrow$ )
- ↳ **WAN's** : DPLL, de ontvanger synchroniseert door een DPLL-sturing.

In de loop van volgend betoog zal je ook opmerken dat de technieken die WAN's hierin toepassen om hun bandbreedte optimaal te benutten uiteindelijk ook ingang zullen kennen in de LAN wereld bij de meest recente en dus meest veeleisende standaarden.

Aangezien deze synchrone transmissiemethode veruit de belangrijkste is in het hele betoog van dit werk worden de synchronisatiemechanismen uitgebreider, en dus in afzonderlijke paragrafen hierna besproken.

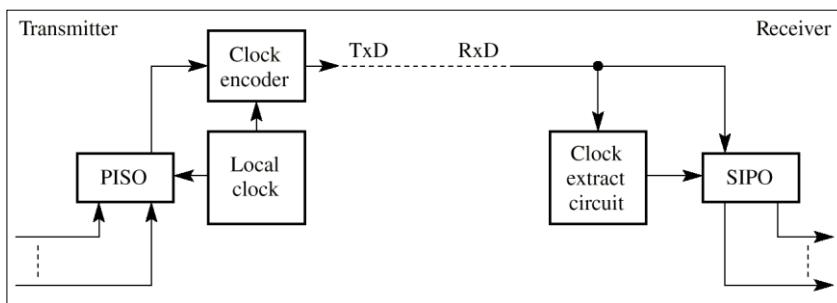
### 1.3.3 BITSYNCHRONISATIE BIJ SYNCHRONE TRANSMISSIONE

Zoals hierboven gesteld, Figuur 87 en Figuur 88, ontbreken start- en stopbits bij synchrone transmissie waardoor een **continue bitstroom** ontstaat.

Synchronisatie kan nu op 2 manieren worden bekomen :

- **LAN** : door clock- of **timinginformatie** op het verzonden signaal te **superponeren**
- **WAN** : de ontvangerklok wordt door een **DPLL** (Digitale PLL) gesynchroniseerd op basis van bit-overgangen in het ontvangen datasignaal

#### BITSYNCHRONISATIE IN LAN'S → KLOK-CODERING.



Het kloksignaal wordt bij de zender 'bovenop' het datasignaal gecodeerd... en bij de ontvanger wordt het dan 'geëxtraheerd'.

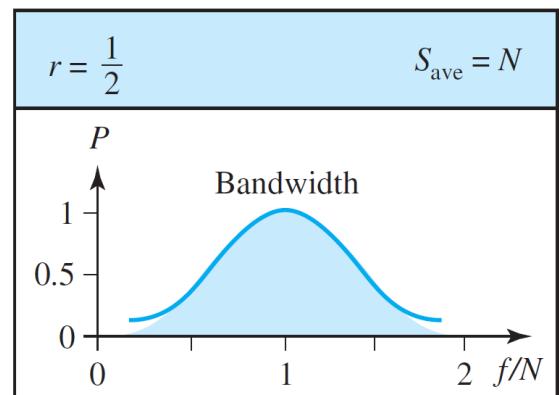
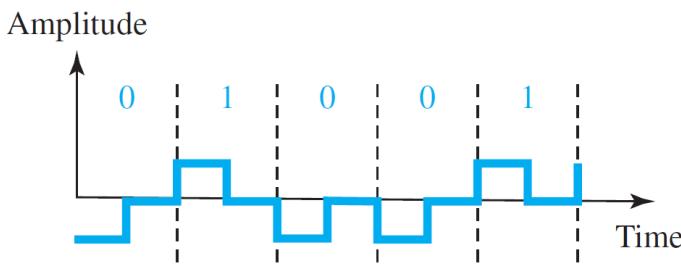
(PISO = Parallel In Serial Out, de data).

Figuur 92 Synchronre transmissie,

klokcodering.

In tegenstelling tot de 'polaire' NRZ-L en NRZ-I schema's van Figuur 89 en Figuur 90 van p. 79 waar er geen klok is gesuperponeerd zal bv. de 'bipolaire RZ' code dit toepassen.

#### RZ RETURN TO ZERO



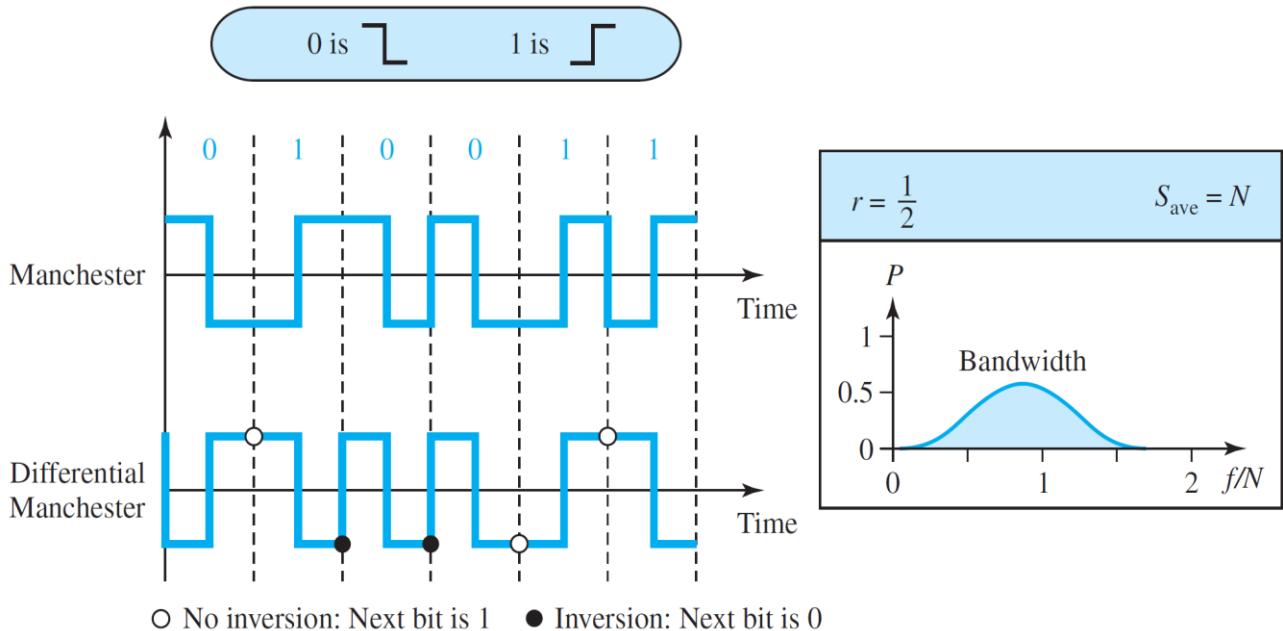
Figuur 93 Een bipolar RZ coderingsschema.

Een '0' geeft een negatieve puls waarna wordt teruggekeerd naar zero, een '1' een positieve. Men heeft dus in elke bittijd een flank waarop men kan synchroniseren. Er is ook geen DC wander probleem bij gelijkmatig verdeelde 0'en en 1'en.

Een probleem is wel dat men 3 niveau's moet ondersteunen én dat het bandbreedte gebruik vrij hoog ligt. Daarom wordt het meestal vervangen door de 'biphase' manchester code (of differential manchester code).

### MANCHESTER EN DIFFERENTIAL MANCHESTER CODE.

Het idee van een transitie in het midden van de bittijd (= kloksynchronisatie) is hier weerhouden, maar slechts met 2 niveau's:



Figuur 94 Manchester en differential manchester coderingsschema's.

Manchester: een positieve flank in het midden van de bittijd =  $1^L$ , een negatieve =  $0^L$ .

Differential manchester: (ook wel bi- $\phi$ -S (biphase Space) code genoemd)

- er is altijd een overgang in het midden van de bittijd +
- in het begin van de bittijd: een overgang =  $0^L$ , geen overgang =  $1^L$ .

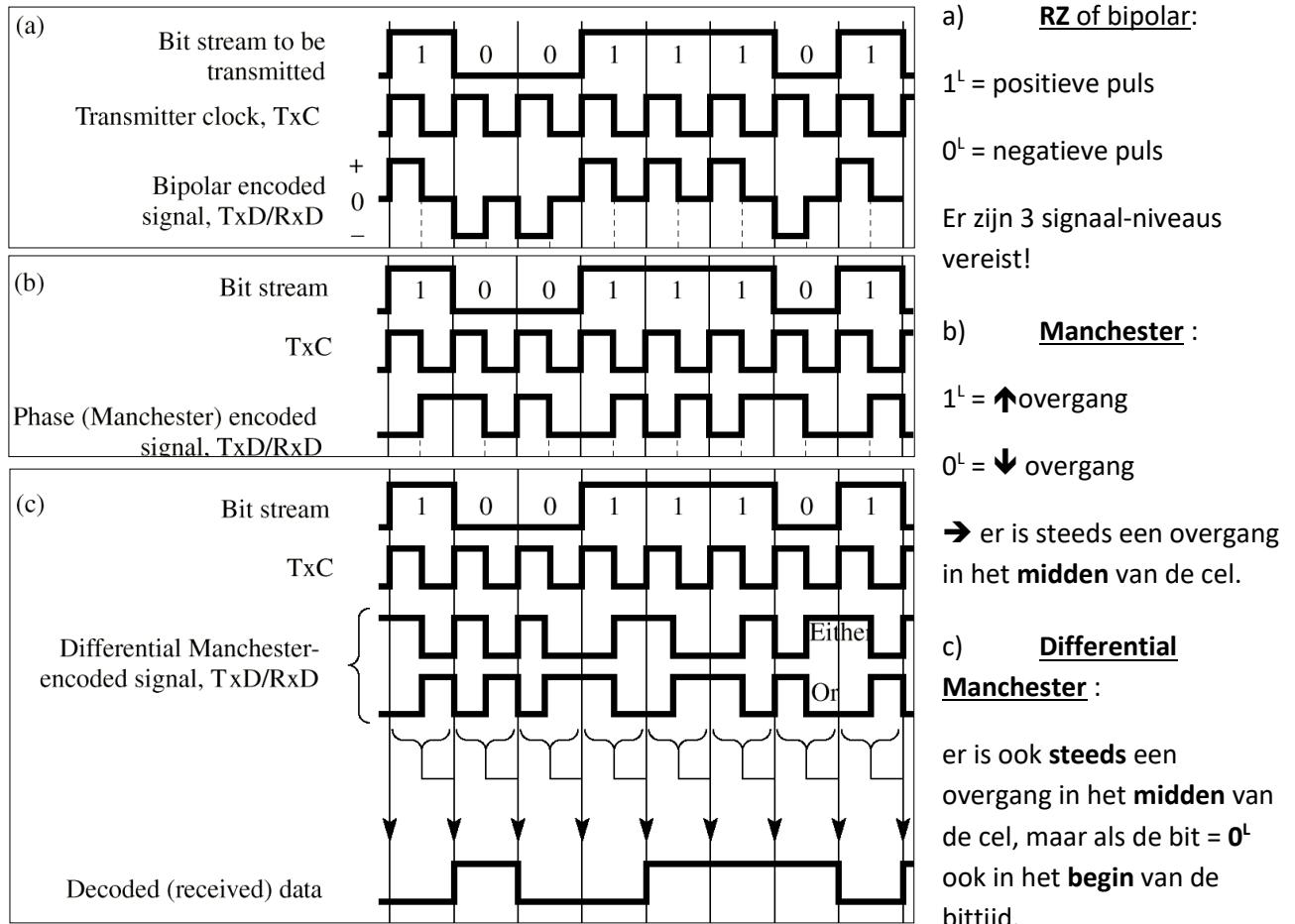
Er is voor beiden geen probleem van DC wander + enkel 2 niveau's.

De vereiste bandbreedte is ong 1,6x de genormaliseerde frekuentie (=f/N of f/R). Manchester code wordt bv. gebruikt in de eerste versie van ethernet met een datarate R = 10Mbps, wat bijgevolg een bandbreedte vereist van  $1,6 \times 10M \rightarrow 16MHz$ .

In een eerste versie van ethernet hadden de toen beschikbare UTP kabels (CAT3 en CAT5) wel voldoende bandbreedte ter beschikking (resp. 16 MHz en 100MHz) maar voor fast ethernet (N of R = x10) werd dit een probleem. Men moest bewuster omspringen met het bandbreedte gebruik, wat resulterde in andere codes die afgeleid werden van de 'WAN code's.

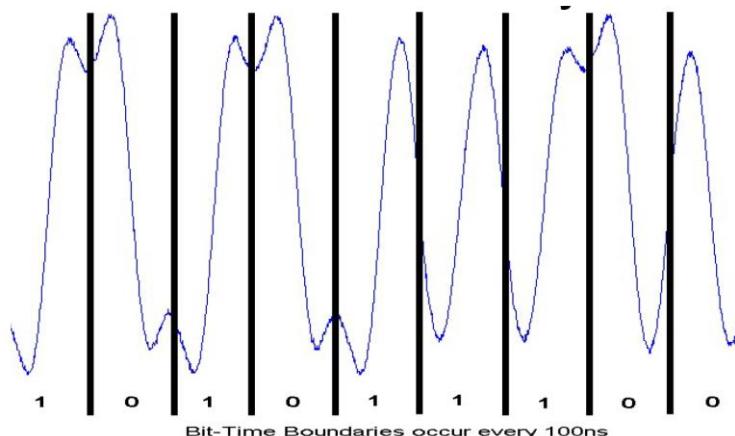
(Een klein voordeel aan differential manchester is dat voor eenzelfde bitpatroon perfect 2 complementaire signalen kunnen dienen... en bijgevolg de draden in een UTP kabel willekeurig kunnen aangesloten worden. Dit heeft trouwens niks te maken met 'cross-' of 'straight- cables' : daarbij worden de PAREN al of niet gekruist)

## RESUME LAN-CODES



Figuur 95 Klokcoderingen bij synchrone transmissie in LAN's.

In werkelijkheid zal de beperking van de bandbreedte (afh. v. het type UTP) er voor zorgen dan **niet** alle harmonischen van het verzonden signaal (bv. typisch de hoogste) worden overgedragen, waardoor de steile flanken aflatken.



Praktisch verloop van de **manchester**-code op een begrensd medium, bv. 100MHz CAT5 met een datarate van 10Mbps . Enkel de eerste harmonischen van bovenstaand theoretisch signaal worden getransporteerd.

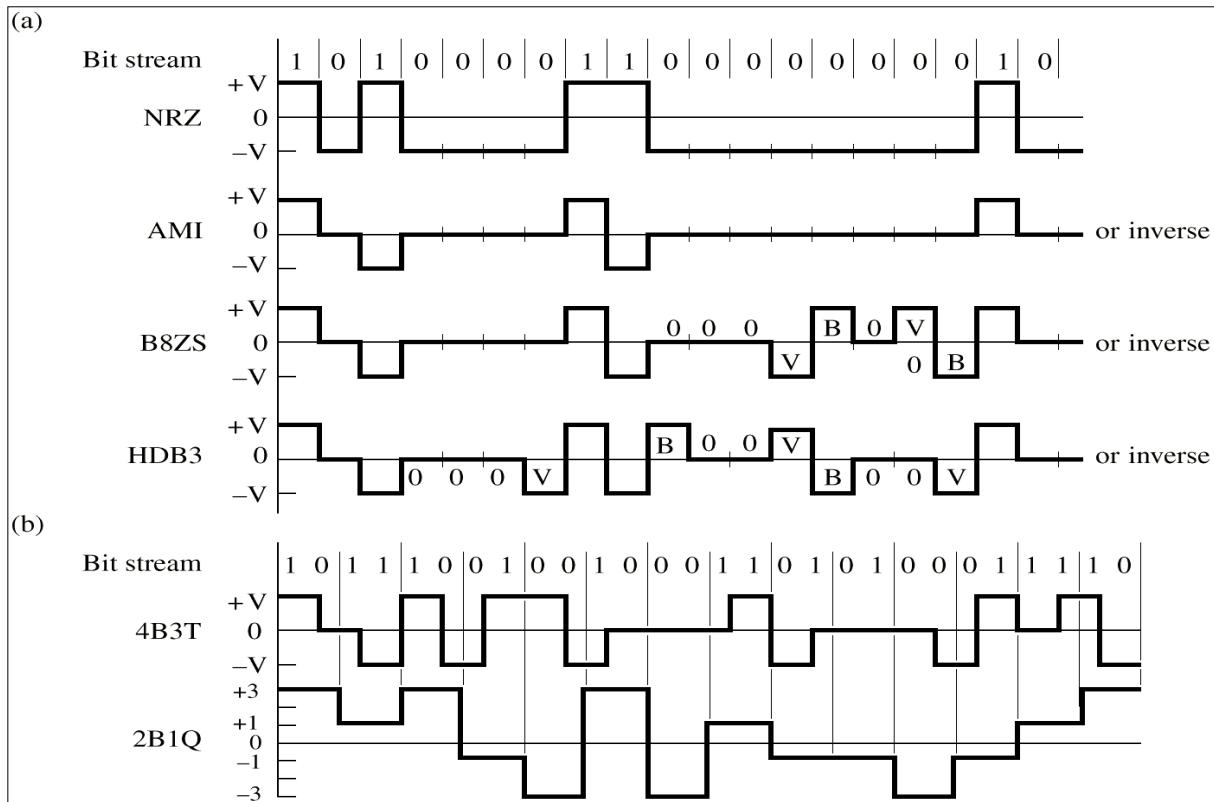
Figuur 96 manchester code, een praktisch signaal.

## BITSYNCHRONISATIE IN WAN'S.

We merken op dat bv. manchester-code 2x zoveel overgangen kent dan een normaal NRZ-signal, zoals bv. bij RS232 (vergelijk ook met onderstaande figuur). Hieruit kan men afleiden dat manchestercodering 2x de transmissiebandbreedte vraagt t.o.v. NRZ.

In WAN omgevingen zijn de afstanden uitgedrukt **km's** (bv. een 'local loop' van 5km), over UTP kabels van minder goede kwaliteit (= 'twistlength'). Hier is bandbreedte wel degelijk een probleem: bv. **1MHz** !

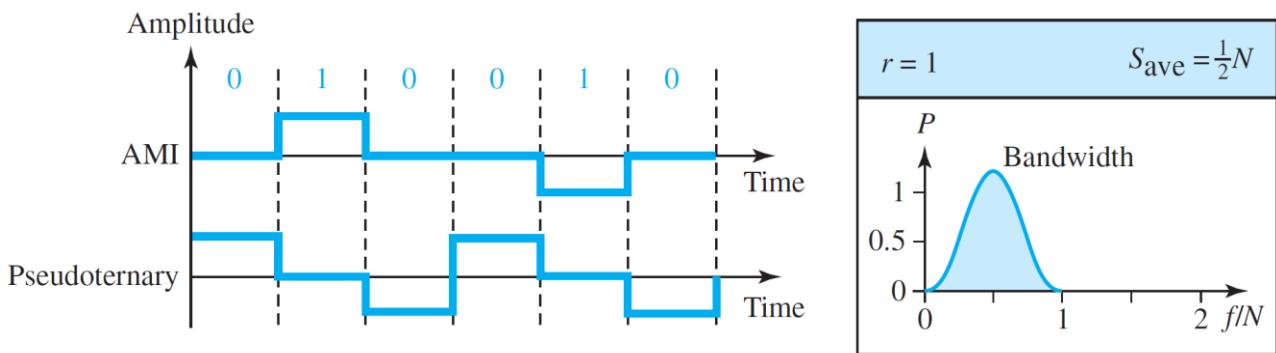
Dat resulteerde in de volgende coderingsschema's.



Figuur 97 Coderingsschema's gebruikt in WAN omgevingen.

De schema's uit Figuur 97 a) zijn allen 'bipolaire' schema's en kunnen bijgevolg geïnverteerd worden, afh.v. het niveau waarop gestart is. Ze gebruiken 3 niveaus (+V, 0, -V).

De schema's uit Figuur 97 b) zijn 'multilevel' schema's en dienen vooral om de baudrate (en dus de vereiste bandbreedte) te reduceren.

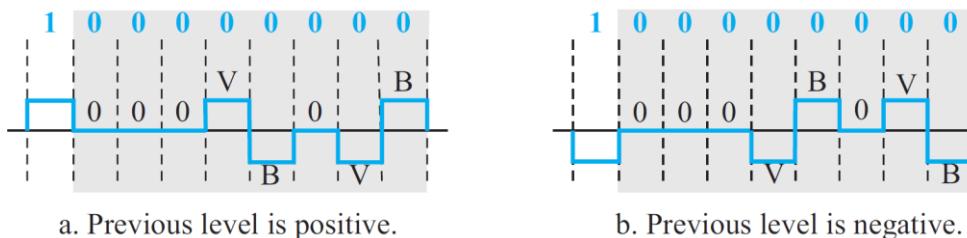
**AMI**

Figuur 98 AMI en ASI Coderingsschema's.

Bij **AMI**, Alternate Mark Inversion, wordt het signaal van **polariteit veranderd** bij elke  $1^L$ . Een nadeel zoals zo dadelijk zal blijken is dat een lange reeks van logische 0-en geen signaal meer geeft waardoor de ontvanger uit bitsynchronisatie kan geraken.

De vereiste bandbreedte is wel beter dan manchester, en ook deze code heeft geen DC-wander probleem.

Een alternatief schema is **ASI** (Alternate Space Inversion) of Pseudoternary waarbij de polariteitsverandering plaats heeft bij een  $0^L$ . Dit schema wordt gebruikt bij ISDN, de S/T interface waarop ISDN apparaten worden aangesloten.

**B8ZS**

Figuur 99 B8ZS Coderingsschema.

Om het probleem van AMI (verlies van bitsync bij een lange reeks 0-en) op te lossen wordt dan **B8ZS** afgeleid: Bipolar with 8 Zero Suppression. Van zodra er **8 opeenvolgende  $0^L$**  optreden wordt dit gecodeerd als **000VB0VB** waarbij B een normale polariteitswissel voorstelt en V een '**violation**'. Een violation breekt de normale AMI regel: een  $0^L$  geeft wel een puls, maar dit keer niet de 'alternerende' zoals een  $1^L$  zou doen. Zie ook Figuur 97 op p.84.

**HDB3**

Een volgende verbetering is High Density Bipolar 3, **HDB3**, Zie Figuur 97

Elke reeks van **4 opeenvolgende  $0^L$**  creëert een violation V op de 4<sup>e</sup>  $0^L$ . Het nadeel zou nu zijn dat een heel lange reeks van 0-en dezelfde pulsen zou creëren en bijgevolg een DC niveau induceren → **DC wander**. In dat geval ( bv.  $8 \times 0^L$  of meer) wordt per **reeks** van  $4 \times 0^L$  de code **B00V** opgewekt.

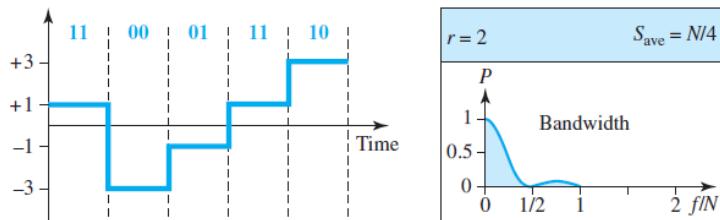
Deel b) van Figuur 97 zijn "**baud rate reduction**" codes. Ze werden ontworpen om hogere bitrates te verkrijgen per gegeven bandbreedte. We hebben reeds gezien in pt. 1.1.5 op p. 47 dat de baudrate of

'signaling rate' de snelheid is waarmee de signaalelementen van een code wijzigen. Het geeft een maat voor de vereiste bandbreedte.

Als we er nu voor zorgen dat 1 signaalelement meerdere bits kan vertegenwoordigen drukken we voor een gegeven bitrate de vereiste baudrate, of krijgen we voor een gegeven baudrate (=bandbreedte) een hogere bitrate. Dit natuurlijk ten koste van de complexiteit van zender en ontvanger.

→ meer dan 1 bit wordt voorgesteld per tijdseenheid of signaalelement.

### 2B1Q



Figuur 100 2B1Q Coderingsschema.

**2B1Q** betekent: **2 Bits worden voorgesteld door 1 Quaternair signaalelement.** (= signaalelement met 4 mogelijke posities).

Het wordt bv. gebruikt op de U-interface bij ISDN, een 'local loop' twisted pair lijn van enkele km, waarover 160kbps moet, maar slechts ... 80kbaud.

Algemeen worden ze **mBnL** codes genoemd : **m** bits worden voorgesteld door **n** pulsen van **L** Levels, waarbij  $m > n$  en  $L > 2$ . Een ander veelgebruikt schema is 4B3T.

### 4B3T

Een combinatie van 4 bits worden voorgesteld door 3 ternaire levels.

Zie Figuur 97 b) : '1011' = + 0 -, '1001' = + - +, ...0001 = 0 - +, 0111 = - 0 + .

Het is niet moeilijk om in te zien dat:

- **4B = 4bits →  $2^4 = 16$**  mogelijke combinaties moeten voorgesteld worden.
- Dit kan door **3T = 3 ternaire niveaus →  $3^3 = 27$**  mogelijkheden. **3T > 4B !!**

Er wordt nu een schema uitgewerkt om de mogelijke problemen te vermijden:

- **DC wander.** De DC component van een gemiddeld signaal verwijderen
- **Kloksynchronisatie:** zorgen voor voldoende signaalovergangen om de ontvanger in bitsynchronisatie te houden.
- Met ongeldige codes: **fouten** opsporen.

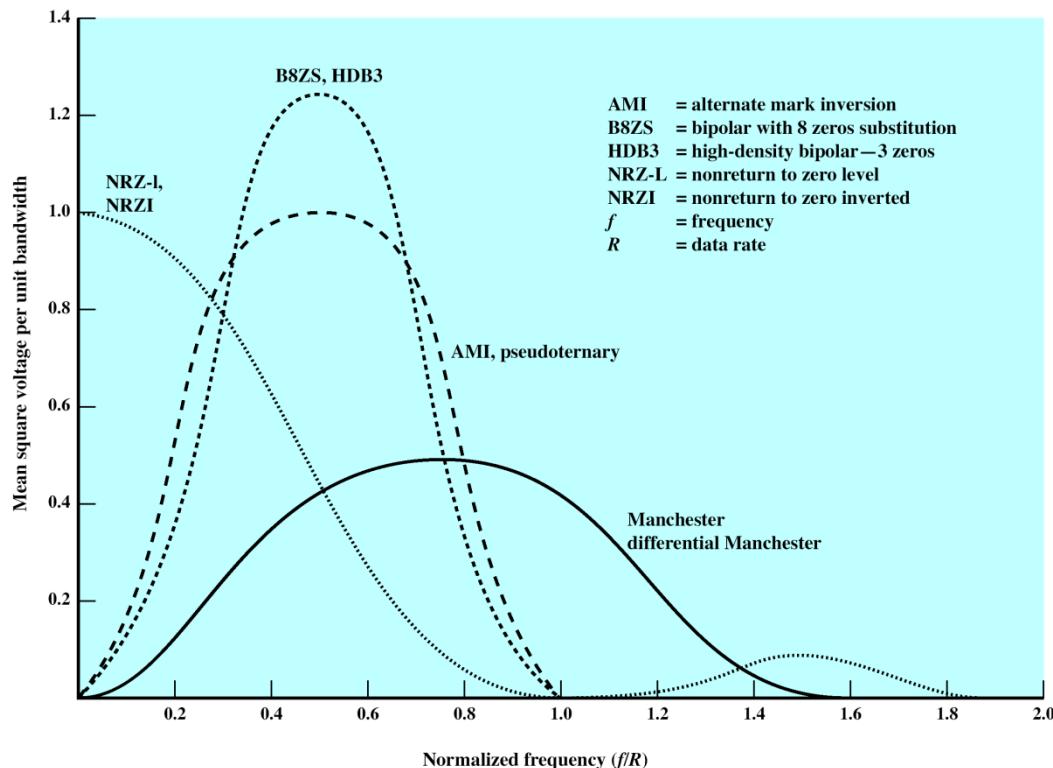
Gelijkwaardige lijncodes werden uiteindelijk ook gebruikt in **LAN** omgevingen om dezelfde redenen: de beschikbare bandbreedtes werd met de steeds hogere datarates wel degelijk een probleem, waardoor men moest terugvallen op de technieken geleerd in de **WAN** omgeving. We zullen deze lijncodes zoals 8B6T, MLT3, 4B/5B, 4D-PAM5, 8B10B bespreken bij hun resp. interface standaard.

## POWER SPECTRUM

Van al deze codes kunnen we de bandbreedte eisen vergelijken. Een vermogenspectrum wordt opgetekend. De meeste transmissielijnen hebben problemen om signalen aan de grens van hun doorlaatspectrum goed te transporteren, daarom heeft een goed ontwerp het meeste vermogen in het midden van het spectrum. In dat geval zal het signaal minder vervormd aankomen bij de ontvanger.

Onderstaande figuur geeft het vermogen ifv. de genormaliseerde frekwentie ( $=f/R$ ). We zien dat NRZ signalen (BV. RS232) een DC component bevatten wat nadelig is voor DC wander.

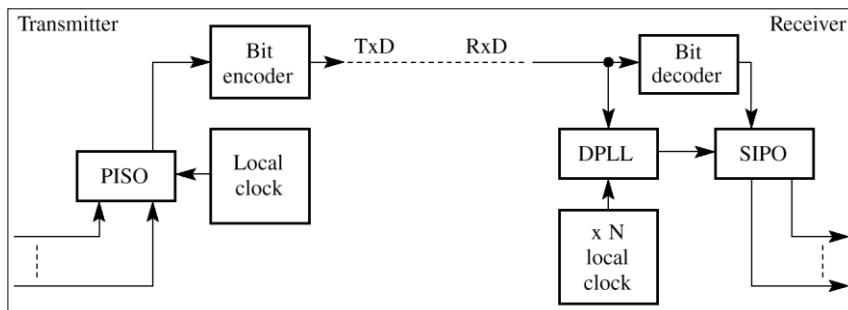
De multilevel codes (AMI, HDB3, B8ZS) hebben een goede verdeling in het spectrum en de Biphase codes (manchesters) hebben dan weer een opvallend groot bereik nodig waardoor ze voor WAN verbindingen niet interessant zijn.



Figuur 101 Powerspectrum van AMI en manchestercodes

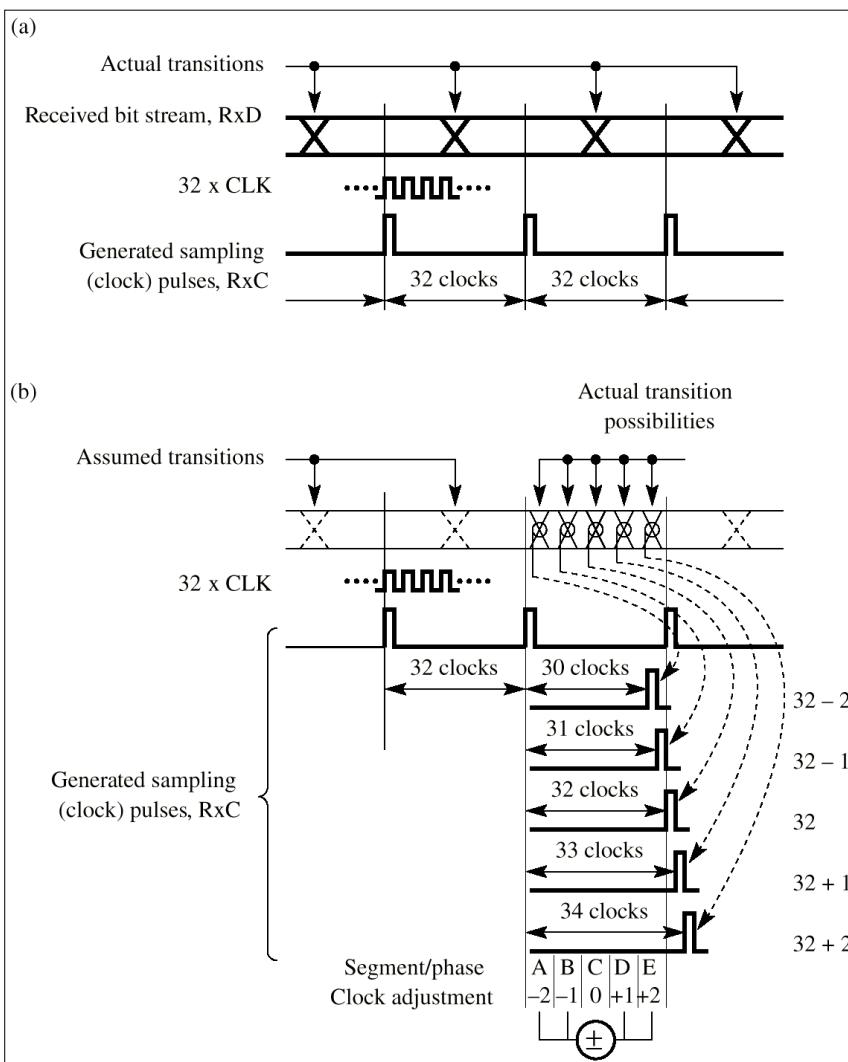
## DPLL

We hebben gezien dat de lijncodes ontworpen werden om ontvangers in bitsynchronisatie te houden door (geregeld) flanken te op te wekken. Een Digital Phase Locked Loop moet deze ontvangerklok in bitsynchronisatie houden.



Er moeten voldoende  $\uparrow$  of  $\downarrow$  overgangen aanwezig zijn om op regelmatige tijdstippen te kunnen synchroniseren. Hiervoor wordt de te verzenden data door een 'scrambler' gestuurd die continue reeksen van 1 of 0 verhindert.

Figuur 102 DPLL kloksynchronisatie.



De ontvangerklok werkt typisch op  $32 \times$  de bitsnelheid  $\rightarrow N=32$ .

Zijn de ontvangerklok en de binnengkomende bitstroom in synchronisatie (a) dan zal de bit gesampled worden 16 pulsen na een overgang, m.a.w. **middenin de bit-cel**.

Driften ontvangerklok en bitstream uit elkaar dan zal de DPLL corrigeren afh.v. de plaats van de **echt** optredende flank.

Hier voor wordt de bitperiode ingedeeld in 5 **segmenten A, B, C, D en E** met elk hun resp. klokaanpassing :  
 $-2, -1, 0, 1, 2$ .

(In praktijk zijn de segmenten A en E groter en B, C en D kleiner, met aangepaste correcties : A en E = 10, B en D 3 of 4.)

Figuur 103 DPLL werking.

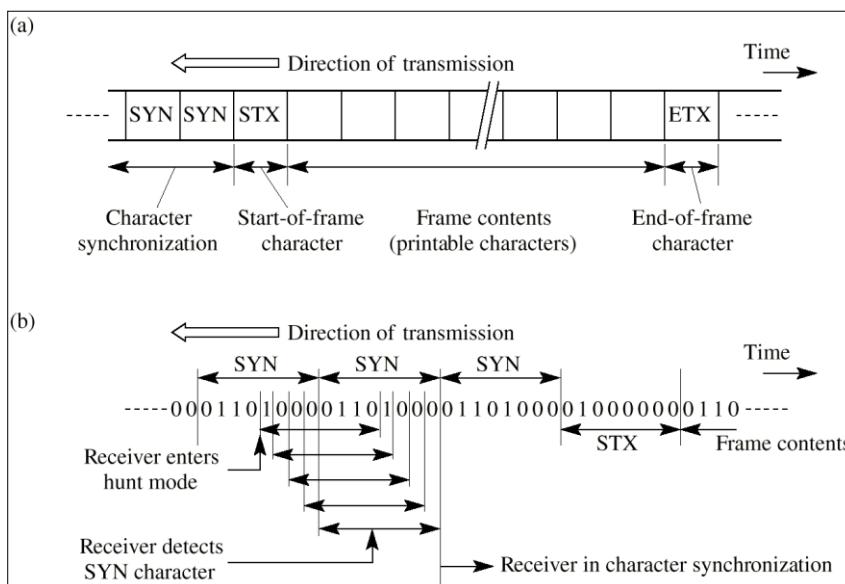
### 1.3.4 KARAKTER- EN FRAME-SYNCHRONISATIE BIJ BYTE-GEORIËNTEERDE SYNCHRONE TRANSMISSION

In praktijk zijn er 2 synchrone controle schema's :

- **byte-georiënteerd** (pt. 1.3.4) (of karakter-georienteerd)
- **bit-georiënteerd** (pt. 1.3.5).

Ze gebruiken beiden dezelfde **bitsynchronisaties** uit voorgaande paragrafen, maar andere **karakter-** en **frame-synchronisaties**.

**Byte-georiënteerde synchrone transmissie** wordt meestal gebruikt voor het oversturen van bvb. **ASCII** karakters. We starten bvb. zonder signaal.



Het **frame** wordt net zoals bij asynchrone transmissie tussen **STX** en **ETX** geplaatst.

Vóór STX worden (een aantal) **SYN** karakters gestuurd om eerst bit- en vervolgens byte-synchronisatie te verkrijgen.

Van zodra de ontvanger in bitsynchronisatie is komt hij in 'hunt' mode, op zoek naar een SYN byte.

Figuur 104 Karakter en frame-synchronisatie bij byte-georiënteerde synchrone transmissie.

Hij plaatst hierbij een venster van 8 bits op de ontvangen bitreeks en controleert of hierin het **SYN**-karakter te vinden is. Indien niet verschuift hij het venster per volgende ontvangen bit. Van zodra hij **SYN** herkent is de ontvanger in **byte- of karaktersynchronisatie!** Hij leest de volgende bits per 8 stuks.

#### Framesynchronisatie.

Van zodra hij karaktersynchronisatie bekomen heeft zoekt de ontvanger karakter per karakter naar het **STX** karakter dat de **start** van het **frame** aanduidt. Een keer dit ontvangen, verwerkt hij de **frameinhoud** waarbij hij zoekt naar **ETX**, = het einde hiervan. In een 'point to point' (=P2P) link kan de zender daarna **SYN** karakters blijven sturen om bit- en byte-synchronisatie te behouden. In een 'multiple access' (=MAC) of shared medium stopt de zender met zenden.

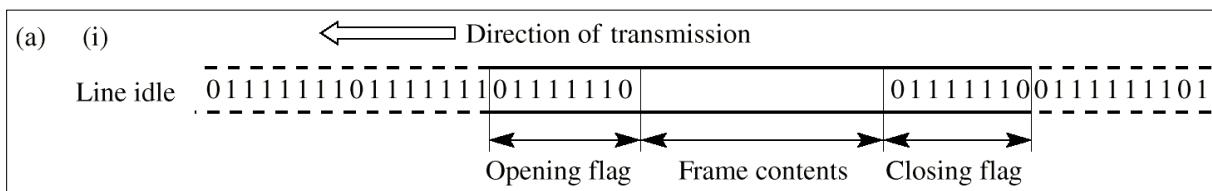
Er moet ook hier evt. voor **transparantie** gezorgd worden, m.a.w. onderscheid maken tussen de **STX-ETX framegrenzen** en evt. voorkomende **STX-ETX databytes**. Zie verder.

### 1.3.5 KARAKTER- EN FRAME-SYNCHRONISATIE BIJ BIT-GEORIËNTEERDE SYNCHRONE TRANSMISSIE.

Om problemen met transparantie t.o.v. STX en ETX te vermijden, en omdat STX en ETX typische ASCII controle karakters zijn waardoor de toepasbaarheid van dit karakterschema wordt beperkt, zijn bit-georiënteerde schema's ontworpen.

We maken onderscheid tussen P2P – en MAC- verbindingen omdat de zender al of niet kan blijven zenden.

#### P2P VERBINDINGEN



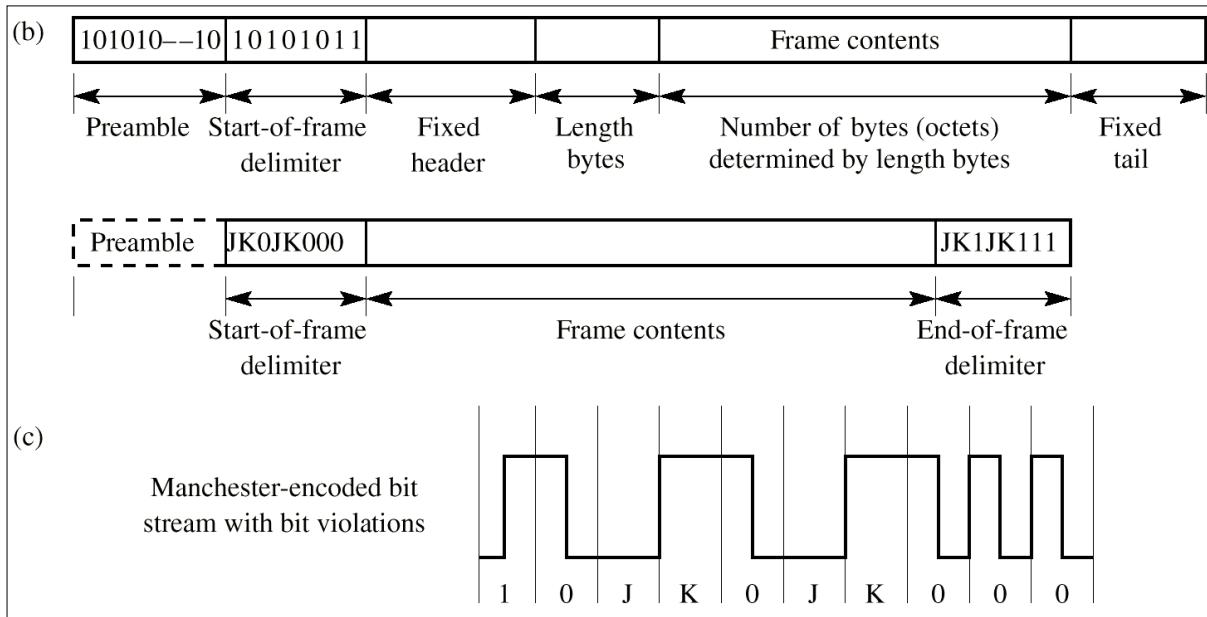
Figuur 105 Karakter en frame-synchronisatie bij bit-georiënteerde synchrone transmissie

Figuur 105 a) wordt gebruikt bij P2P verbindingen.

In de 'idle' toestand wordt **continue** de sequentie '0111 1111' gestuurd om **bitsynchronisatie** te behouden. **Start-** en **einde** van het **frame** worden aangeduid door '0111 1110', de 'flag' genoemd.

Datatransparantie zal erin bestaan om na 5 opeenvolgende  $1^L$  bits een  $0^L$  te 'stuffen' zodat noch 'idle', noch 'flag' in het frame kan voorkomen, zie verder pt. 2.1.4 op p. 152.

## 'SHARED MEDIUM' / LANS



Figuur 105 Karakter en frame-synchronisatie bij bit-georiënteerde synchrone transmissie

## BITS/BYTES TELLEN → ETHERNET

Figuur 105 b) wordt gebruikt bij 'shared medium' of 'multiple access' situaties zoals bv. LAN's.

Hierbij kan de zender **niet** continu 'idle' sturen, hij moet stoppen na zijn boodschap. Om de ontvanger(s) nu toe te laten **bitsynchronisatie** te bekomen wordt er eerste een '**preamble**' gestuurd, bestaande uit reeksen '10' koppels. Een keer in bitsynchronisatie zoekt de ontvanger, in 'hunt' mode, nu naar de combinatie '10101011', de 'start of frame' delimiter.

In het begin van het frame volgt dan een van te voren afgesproken vaste '**header**' die o.a. het adres van de bestemming bevat, gevolgd door de '**frame lengte**'. Er moeten nu alleen **bytes geteld** worden om het einde van het frame te kennen. Al deze bytes zijn per definitie **inhoud** van het frame.

Dit schema wordt bvb. gebruikt bij **ethernet** LAN's op basis van **manchester** bitcodering.

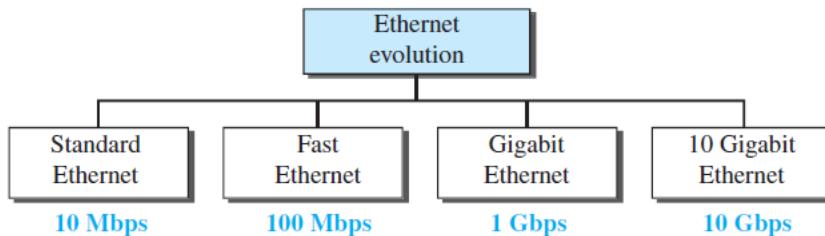
## ENCODING VIOLATIONS → TOKEN RING

Figuur 105 c) wordt eveneens gebruikt bij 'shared medium' of LAN- situaties. Start- en einde van het frame worden nu aangeduid door '**bit-encoding violations**'. Dit zijn bitpatronen die volgens de bitcodering **niet** kunnen voorkomen : bv. hier manchestercodering heeft verplicht steeds een **transitie** in het **midden** van de bit-cel. De 'J-violation' blijft de volledige cel op hetzelfde niveau als de vorige bit, de 'K-violation' zal gedurende de volledige cel het andere niveau aanhouden.

Om start en einde van het frame te verkrijgen zal de ontvanger na **bitsynchronisatie** op de **preamble** zoeken naar 'JK0JK000' (SOF) en later naar 'JK1JK111' (EOF). J en K kunnen **in het frame niet voorkomen**, waardoor transparantie is verzekerd. Dit schema wordt gebruikt bij **token ring** LAN's op basis van **manchester** bitcodering.

## 1.4 INTERFACE STANDAARDS OP DE FYSIEKE LAAG.

In deze standaards zijn de voorgaande theoretische beschouwingen verwerkt. Ze dienen eigenlijk als praktisch voorbeeld hiervan. We bespreken enkel de belangrijkste... vooral ethernet.



Figuur 106 ethernet evolutie

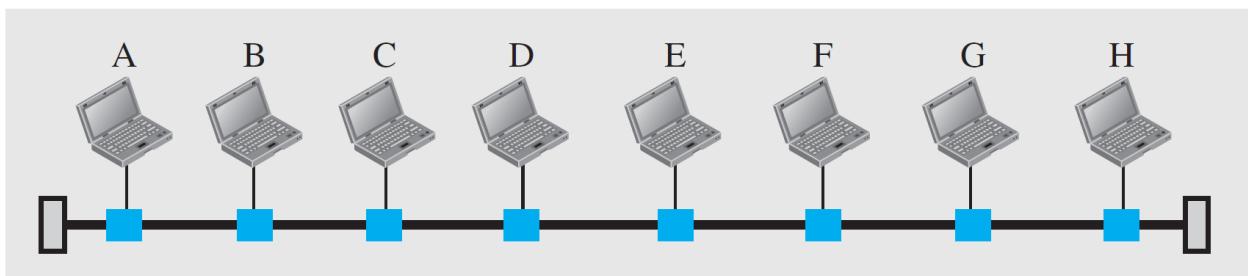
We bespreken in dit hoofdstuk (L1) enkel de fysieke parameters van de ethernet standaarden. De volledige standaard definiert ook layer 2, maar daarvoor verwijzen we naar hoofdstuk 2, pt. 2.5 op p. 188 e.v.

### 1.4.1 STANDAARD ETHERNET 10 BASE...

We kunnen nu nog geen bespreking geven van de **volledige** standaard aangezien die ook richtlijnen geeft over de datalink-laag wat pas in volgend hoofdstuk (2) aan bod komt. We komen hierop terug.

Standaard ethernet kende een aantal topologien die gebruik maakten van verschillende bekabeling.

#### 10 BASE 2 EN 10 BASE 5: BUS TOPOLOGIEN

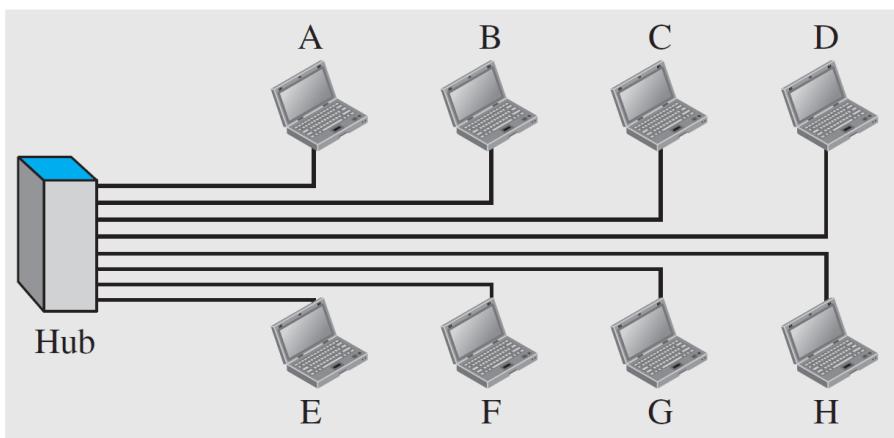


Figuur 107 ethernet bustopologie

Een ethernet segment wordt gekenmerkt door een reeks computers die letterlijk 'aan dezelfde draad of kabel hangen'. Het is een MAC (multiple acces) of shared medium. Er kan op 1 moment slechts 1 toestel zenden. Ze zien allen dezelfde bitstroom van deze zender voorbijkomen.

Op het einde van een segment plaatst men 'terminators' (= 50 ohm) die identiek zijn aan de karakteristieke impedantie van de coax waardoor het signaal a.h.w. 'verder loopt', of in werkelijkheid volledig gedissipeerd wordt.

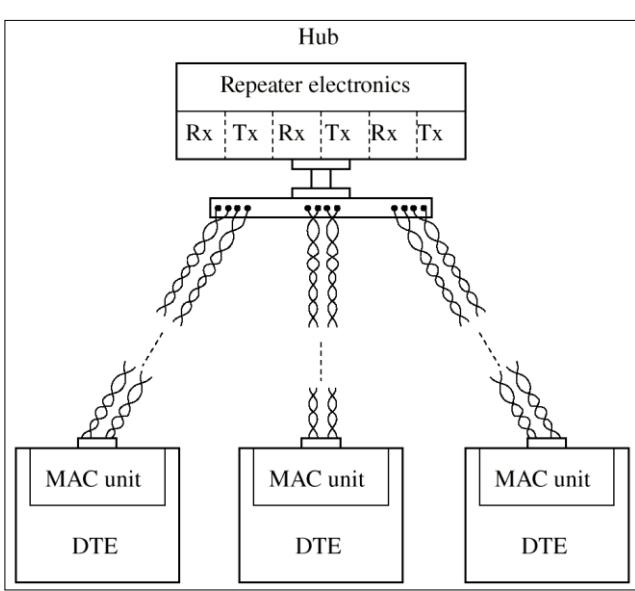
## 10 BASE T: EEN STER TOPOLOGIE



Figuur 108 ethernet 10base-T : stertopologie

De 'DTE's worden hier aangesloten met UTP kabels als een fysische **sterbekabeling** m.b.v. een hub. Signaaltechnisch is het nu zo dat in een hub een reeks elektronische repeaters (= versterkers) zitten die de ontvangen bits bit per bit samplen en hergenereren → ook dit is een '**shared medium**', MAC.

Men noemt de volledige Figuur 108 ook een 'ethernet-segment'.



Van de 4 paren in de UTP kabel worden er slechts 2 gebruikt : 1 om te zenden, 1 om te ontvangen. Zie ook Figuur 53 op p.57.

Het type UTP-kabel was:

- CAT3 (16MHz) in USA
- **CAT5** (100MHz) in Europa en USA.

De maximale lengte tussen hub en DTE is **100m**. Reden hiervoor is enerzijds verzwakking, zie Figuur 27 op p. 38, maar ook de noodzaak om collisions te detecteren, zie H2 datalinklaag.

Figuur 109 Sterbekabeling bij ethernet 10baseT

De signalen worden niet gemoduleerd, m.a.w. **basisband**, en zijn gecodeerd met **manchester** code tegen **10Mbps**.

De belangrijkste taak van een hub/repeater is om **een** binnenkomend signaal op gelijk welke poort te herhalen op **alle** andere poorten. Hij voorziet daarvoor adaptieve '**NEXT-cancellers**' op elke poort.

## FRAME FORMAAT ETHERNET

Standaard ethernet is steeds een shared segment waarbij zenders na het verzenden van een frame moeten stoppen en anderen de kans geven om een frame te sturen. (interframe gap = 'idle'). Voor elk nieuw frame moet er terug gesynchroniseerd worden.



Preamble	7 octets	Vóór er een frame wordt opgestuurd wordt een 'preamble' gestuurd van 7 bytes om de ontvangers in <b>bitsynchronisatie</b> te krijgen.
SFD	1 octet	
Destination address	6 octets	De preamble is van de vorm '10101010', gevuld door één <b>SFD</b> (Start of Frame Delimiter) = '10101011', zoals te zien was in Figuur 105 op p.91
Source address	6 octets	
Length indicator	2 octets	De 'length indicator' geeft de lengte van het dataveld aan, m.a.w. het einde van het frame is 4 bytes verder.
Data	≤ 1500	
Pad (optional)		Voor meer informatie verwijzen we naar volgend hoofdstuk, pt. Ethernet - IEEE 802.3 op p.185.
Frame check sequence	4 octets	

**Figuur 110** Frame vorm bij IEEE 802.3 / ethernet.

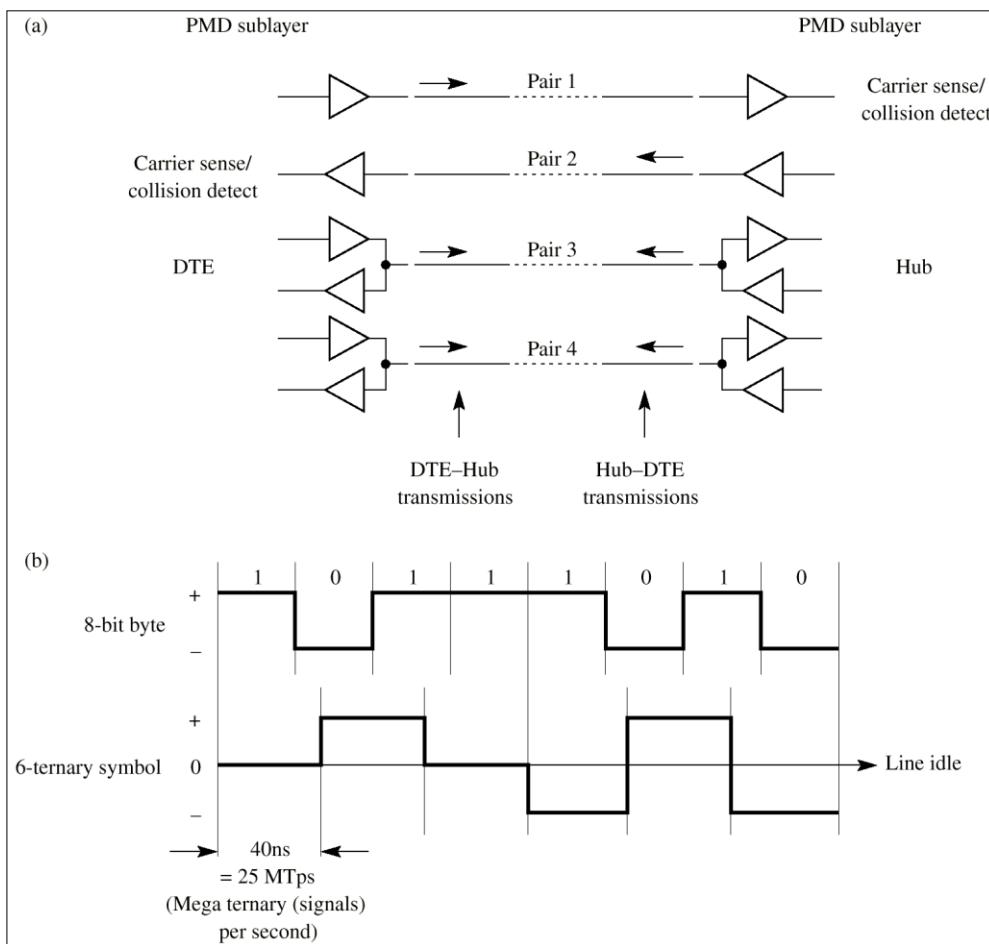
### 1.4.2 FAST ETHERNET.

Het doel van ‘fast ethernet’ bestond erin om netwerken (**10x**) sneller te maken, gebruik makend van **dezelfde (UTP) bekabeling** en **frame-vorm** (én medium access methode, zie verder). Verwonderlijk genoeg moest men hiervoor afstappen van coax bekabeling, het wordt enkel ondersteund op **UTP** (→ 100baseT) en fiber (→ 100baseF). Later, op p. 195 , zal blijken waarom.

Praktisch waren er (bij het ontwerp van Fast ethernet) voor UTP 2 bekabelingstandaards : **CAT3** en **CAT5**, met duidelijk verschillende **bandbreedtes**, resp. **16MHz** en **100MHz**. Gevolg is 2 soorten 100baseT : resp. **100baseT4** en **100baseTX**.

Met datasnelheden van 100Mbps is het **niet** aangewezen **manchester**-codering te gebruiken vanwege de te grote vereiste bandbreedte voor UTP ( $1,6 \times 100\text{Mbps} = 160\text{MHz}$ ). Men moet overstappen op baud reducerende codes, zoals besproken in Figuur 97 op p.84 en Figuur 101 op p.87.

## 100BASET4

(ook **100VG** genoemd, 'Voice Grade' = fast ethernet, 100Mbps op CAT3 kabel, USA)

Figuur 111 100 base T4 aansluiting en codering.

Er worden **4** paren gebruikt van de kabel bij gebrek aan bandbreedte . ➔ 2 'supplementaire' paren, bidirectioneel gebruikt. (de oorspronkelijke paren worden gebruikt voor collision detectie en carrier sensing, zie verder.) Er moet rekening gehouden worden met **powersum NEXT!**

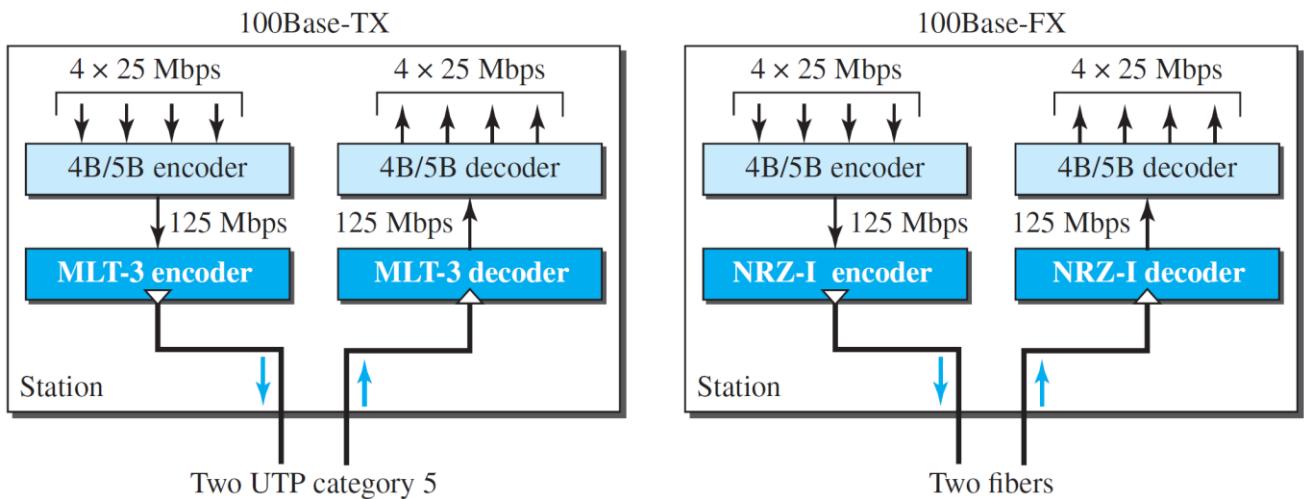
3 paren (max.) per richting ➔  $100 \text{ Mbps} / 3 = 33\text{Mbps}$ . In manchester codering zou dit een power spectrum geven ( $\times 1,6$ ) tot 50MHz en dit is > max. bandbreedte voor CAT3 – UTP (16MHz)

Daarom wordt een 3 level code gebruikt : **8B6T** . 8 bits worden voorgesteld door 6 ternaire levels ➔ baudreducerend. 'signaling rate' is dan  $33 * 6 / 8 = 25 \text{ Mbaud}$  ➔ +/- binnen limiet.

6T levels ➔  $3^6 = 729$  mogelijke codes waarvan we er  $8B = 2^8 = 256$  nodig hebben. Er worden enkel codes gebruikt die minstens 2 transities hebben voor klok-synchronisatie en bovendien zorgen voor DC balance.

## 100BASE X

100base X is ontworpen zowel op UTP als op Fiber → resp 100base TX en 100base FX.



Figuur 112 100 base TX en FX aansluiting en codering.

koper → 100baseTX maar dan alleen op de hogere kwaliteits- **CAT5** UTP bekabeling (DG data grade))

Er worden nog altijd slechts **2** paren/fibers gebruikt, dezelfde als bij 10baseT.

Manchester code lukt niet dus wordt een **4B5B** code wordt gebruikt : 4 bits voorgesteld door een 5bit **binair** symbol

- **4B**: 16 mogelijke codes uit ...
- **5B**: 32 symbolen

→ zorg voor **kloksynchronisatie** door minstens om de 2 bits een transitie te voorzien.

Let wel op : deze 4B/5B code vereist 125Mbaud signaling rate =  $100 \times 5 / 4$ . Het is eigenlijk een baudinducierende code.

In een **NRZI** lijn code, zoals die op **fiber** gebruikt wordt (**FX**) zal er een transitie plaatsvinden bij elke 1. Een 0 behoudt het vorige niveau. Er komen nooit meer dan 2 nullen na elkaar voor synchronisatie.

Data Input (4 bits)	Code Group (5 bits)	NRZI pattern	Interpretation
0000	11110		Data 0
0001	01001		Data 1
0010	10100		Data 2
0011	10101		Data 3
0100	01010		Data 4
0101	01011		Data 5
0110	01110		Data 6
0111	01111		Data 7
1000	10010		Data 8
1001	10011		Data 9
1010	10110		Data A
1011	10111		Data B
1100	11010		Data C
1101	11011		Data D
1110	11100		Data E
1111	11101		Data F

Tabel 1 4B5B code met NRZI signaal.

Van de **niet - data** symbolen stellen er er 2, J en K, de start van een frame voor. (J en K = resp. part 1 en 2).

2 andere, R en T, stellen de stop delimiters voor.

De andere ‘specials’ worden gebruikt voor ‘link control’ functies : idle en error.

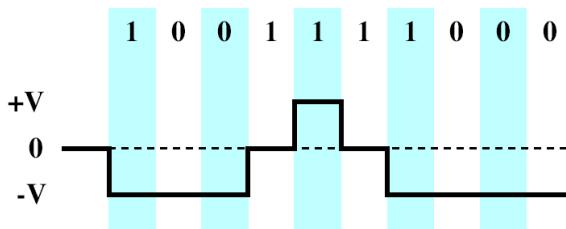
	11111		Idle
	11000		Start of stream delimiter, part 1
	10001		Start of stream delimiter, part 2
	01101		End of stream delimiter, part 1
	00111		End of stream delimiter, part 2
	00100		Transmit error
	other		invalid codes

Tabel 2 4B5B code link control

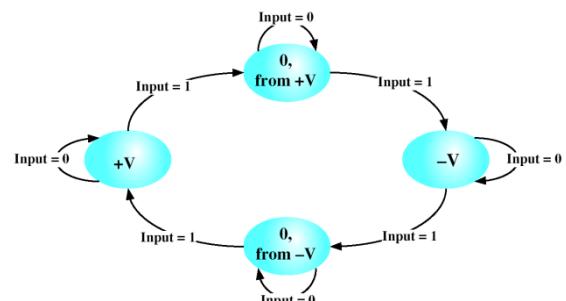
## 100BASE TX

Een 4B5B-NRZI code is perfect om op een **fiber** te sturen, maar voor **koper** is het energiespectrum te groot (tegen 125Mbaud). De 4B5B code wordt op de lijn gecodeerd in MLT3.

MLT3 (Multi-Level Transmit) is een lijncode met 3 signaalniveau's +V, 0 en -V. Net zoals NRZI gaat het signaal naar ‘het andere niveau’ bij een 1 en blijft het hetzelfde niveau houden bij een 0. Aangezien er nu **3 niveau's** zijn is dit iets (maar niet veel) complexer.



Figuur 113 100 base TX MLT3 lijncodering.



Het gevolg van deze lijncode is dat het merendeel van het powerspectrum zit rond 30MHZ. M.a.w. veel gunstiger tegen uitstraling en dus ook interferentie van en naar andere paren.

Vraag. In bovenstaande tekening krijgen we problemen met bitsynchronisatie als hierna, op het einde nog meer 0'en verschijnen. Hoe wordt dit opgelost?

### 1.4.3 GIGABIT ETHERNET

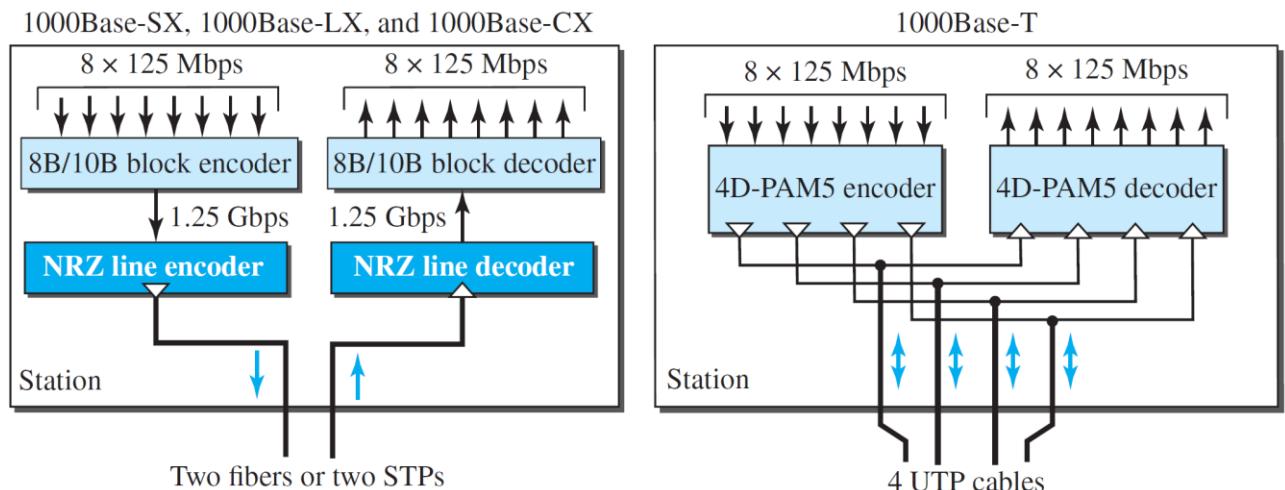
De volledige implementatie van gigabit ethernet op koper (CAT5e – CAT6) is te vinden op een aantal tutorials op het net : <http://www.iol.unh.edu/services/testing/ge/training>.

Check vooral ‘The Ethernet Evolution From 10 Meg to 10 Gig How it all Works!’

Gigabit ethernet gebruikt dezelfde CSMA/CD techniek, zelfde frame grootte en structuur als zijn voorgangers. (Praktisch meestal P2P verbinding op een switch, niet shared medium zoals bij een hub, alhoewel het kan). Check ook hoofdstuk 2 voor CSMA/CD.

#### IMPLEMENTATIES / FYSIEKE UITVOERINGEN :

Gigabit ethernet kan bestaan in een ‘2-wire’ – of een ‘4-wire’ uitvoering.



Figuur 114 1000base fysieke laag

De ‘2-wire’ uitvoeringen gebruiken een NRZ schema, terwijl de ‘4-wire’ versie een 4D-PAM5 code gebruikt.

- 1000 base-SX : ‘S’ staat voor ‘Short’ wavelength → van 770 tot 860 nm, die een duplex link (2 fibers) supporteert, tot **275 m** over **62.5µm** multimode fiber of tot **550m** over **50µm** MM fiber
- 1000 base-LX : ‘L’ staat voor ‘Long’ wavelength → van 1270 tot 1355 nm, supporteert een duplex link, normaal tot **550m** over **50µm** MMF fiber en zelfs tot **5 km** over **8 à 10µm** ‘singlemode’ fiber (SMF).
- (1000 base-CX) : dit werd gebruikt in datacenters om op korte afstanden in koper gigabit links te leggen. De kabels zijn speciale **shielded** twisted pairs, STP. Nu praktisch niet meer voorkomend en vervangen door 1000baseT.
- 1000 base-T : veel voorkomend, over 100m CAT 5e bekabeling (100MHz) wordt eveneens gigabit getransporteerd, een technologische uitdaging.

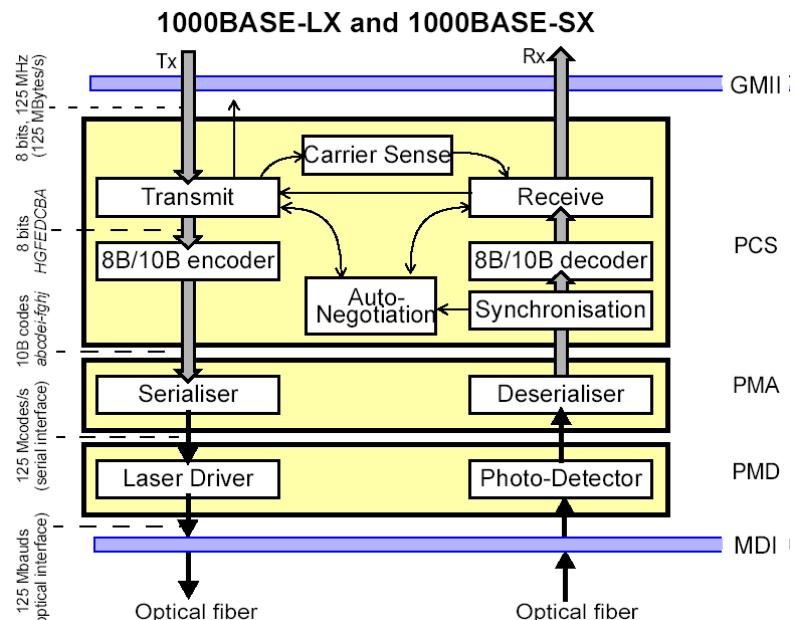
## 1000BASE X

De signaalencodering voor de eerste 3 gigabit ethernet uitvoeringen (1000 base-...X) is steeds 8B/10B, een gelijkaardig schema als 4B5B.

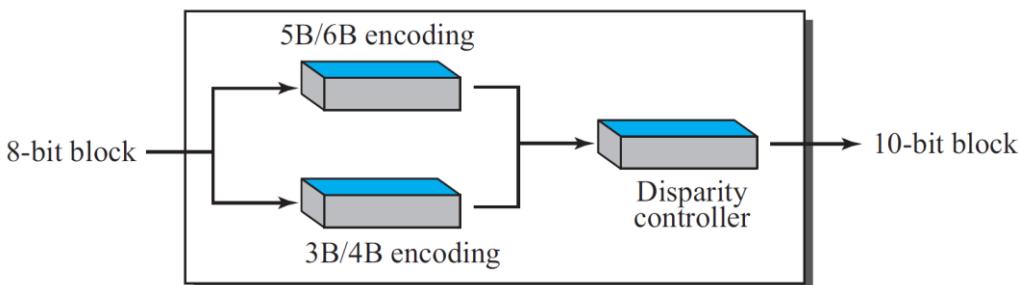
Figuur 115 1000baseX fysieke laag

8B/10B.

Dit is een block-code die te vergelijken is met 4B/5B. Ze geeft meer mogelijkheden op foutdetectie (=meer 'illegale' codes).



Om de substitutietabellen wat te beperken is het uitgevoerd in 2 delen: een 5B/6B en een 3B/4B encoding.



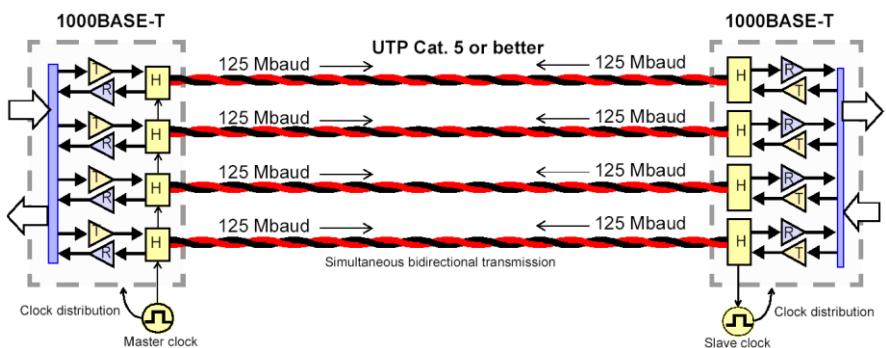
Figuur 116 1000baseX fysieke laag

Net zoals bij 4B/5B is het ook een baudINducerende code waarbij de resulterende (10bit-) stroom een snelheid heeft van 1,25Gbps (+25%)!, telkens op 1 draad: fiber of STP. Dat principe zou zelfs op 4 paren verdeeld bij UTP CAT5 (met een bandbreedte van 100MHz) niet lukken, laat staan op slechts 2 paren, 1 per richting zoals in de 1000base ...X versie steeds het geval is. Daarom is er een andere lijncode voor 1000baseT (= niet X!) wat net bedoeld is om op die bestaande CAT5 bekabeling gigabit ethernet te ondersteunen.

## 1000 BASE-T

Er worden nu **alle 4** de paren van een CAT5-UTP kabel gebruikt, **elk paar** is full duplex gemoduleerd  
→ 8 transmitters én receivers.

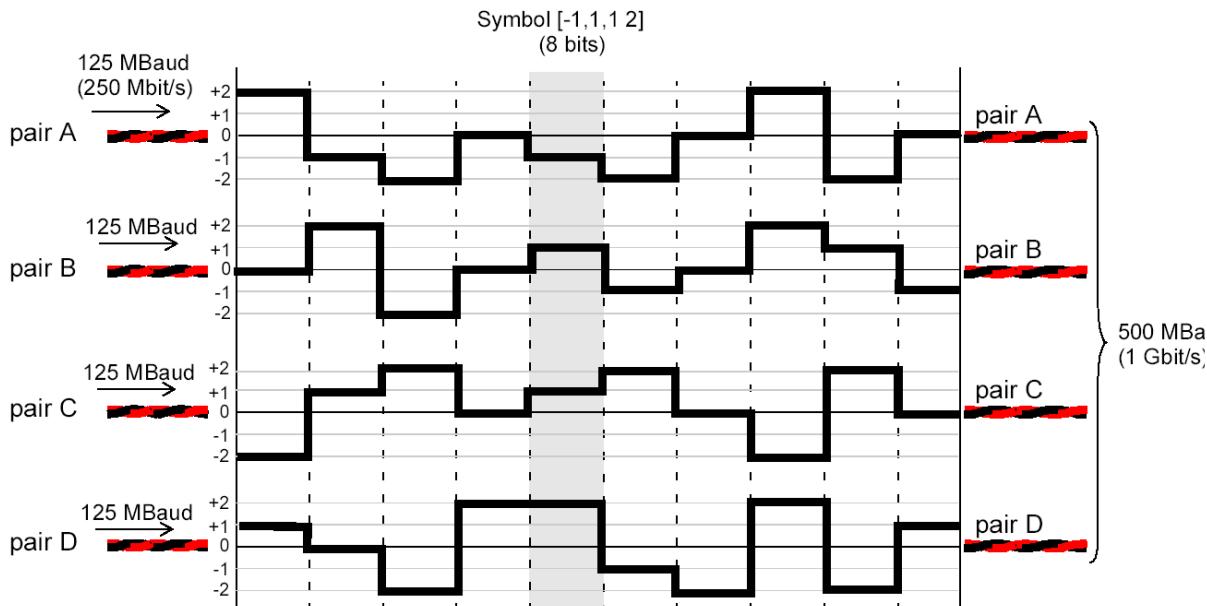
Er is slechts 1 master clock. De slave genereert zijn clock uit het ontvangen signaal.



Figuur 117 1000baseT op CAT5e

Wie de master-clock zal leveren wordt beslist door autonegotiatie.

De signaalcodering voor 1000 base-T is een 4D-PAM5 code. 4D (Dimensional) betekent dat elk paar UTP een dimensie van de lijncode voorstelt.

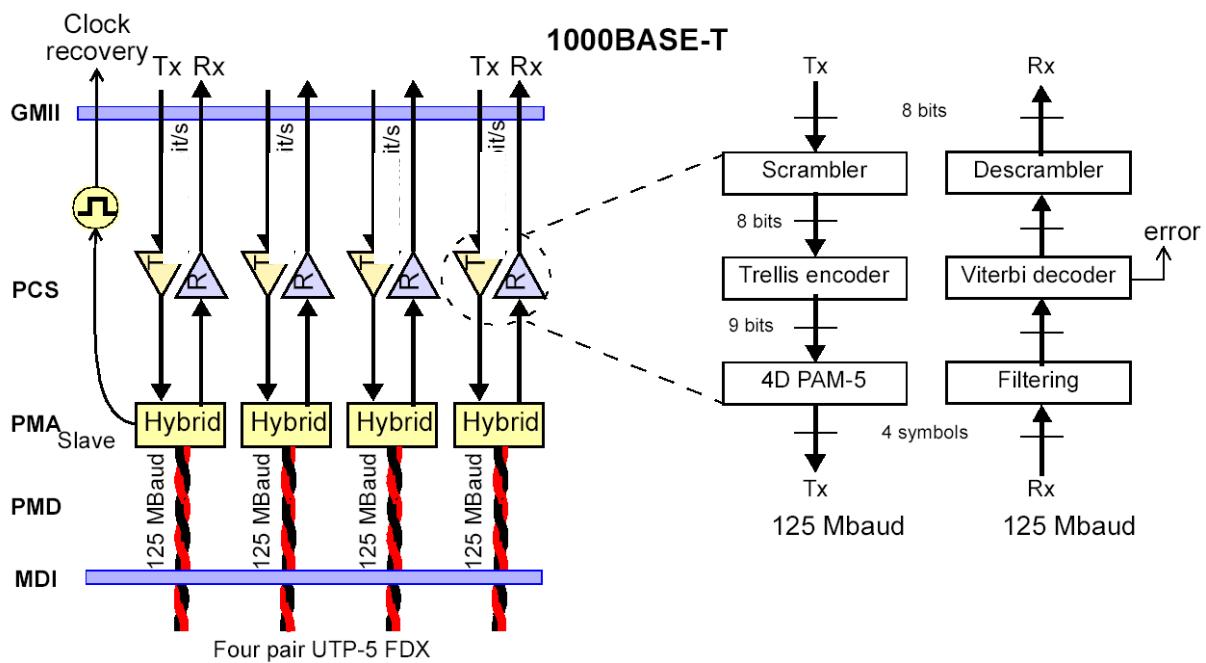


Figuur 118 4D (dimensional) 5 level basisband PAM code

Een 5 level PAM (pulse amplitude modulatie) code kent 5 niveaus waarin 2 bits worden voorgesteld. Op de volledige kabel, 4 twisted pairs, betekent dit dus dat 1 signaalelement, in de 4 dimensies, **8bits = 1byte voorstelt**. Men moet dit dus doen tegen een signaling rate van 125Mbaud. x 8 bits per symbol = 1000Mbps. (125Mbaud, is idem als 100baseTX!).

Dit gebeurt tegelijk in de 2 richtingen via een hybride schakeling → Full duplex.

Nu kan je met 4 paren waarop telkens 5 ‘levels’ worden gestuurd  $5 \cdot 5 \cdot 5 \cdot 5 = 5^4 = 625$  mogelijke ‘constellatiepunten’ voorzien. 1 byte kent slechts **256 #** mogelijkheden → **code redundancy** voor een trellis codering en Viterbi decodering. (= zelfcorrigerende code).



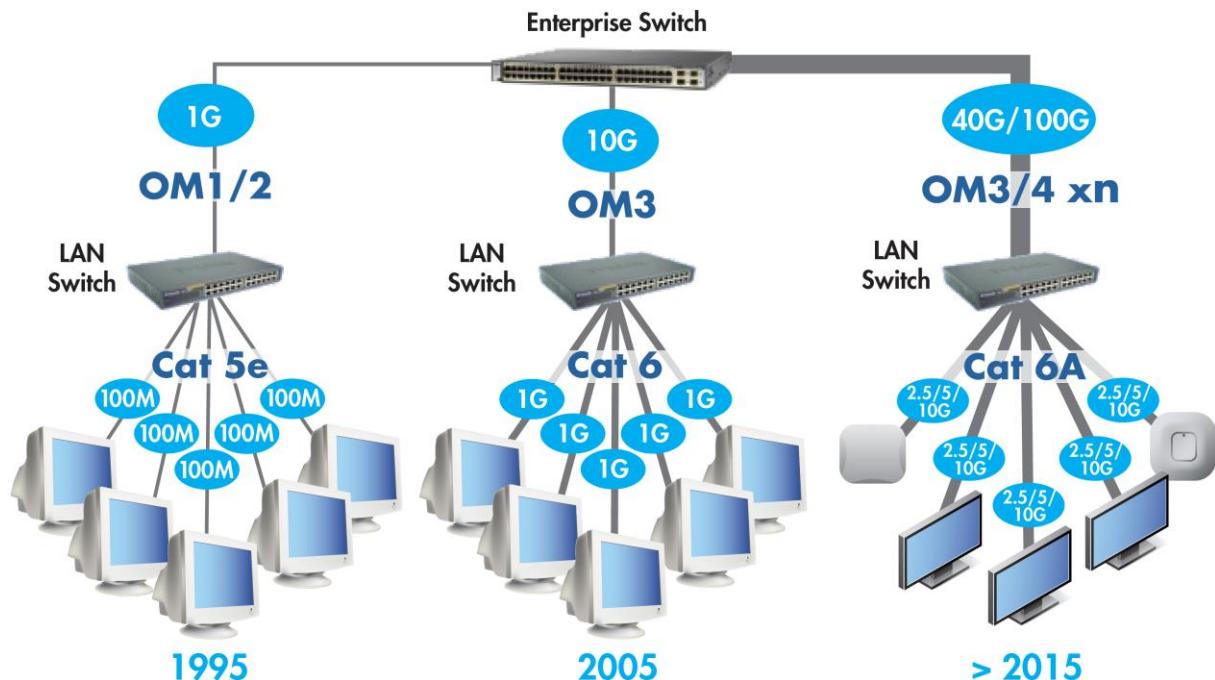
Figuur 119 1000baseT fysieke laag

Gigabit ethernet implementaties:

Implementatie	Medium	Medium Lengte	#'wires'	codering
1000Base-SX	Fiber S-W	550 m	2	8B/10B + NRZ
1000Base-LX	FiberL-W	5000 m	2	8B/10B + NRZ
1000Base-CX	STP	25 m	2	8B/10B + NRZ
1000Base-T	UTP	100 m	4	4D-PAM5

#### 1.4.4 10 GIGABIT ETHERNET, IEEE 802.3ae

ISP's en andere hoge eisen stellende backbones kunnen voorzien worden van deze interface die nogal veel versies kent. Vooral omdat de NICs in desktopmachines reeds gigabit ethernet voorzien, is het nu verplicht om de (LAN-)backbones zwaarder uit te rusten. Servers zijn tegenwoordig zelf reeds voorzien van een 10Gbps interface! (➔ backbones naar 40 en 100Gbps!).



Figuur 120 10 Gigabit ethernet evolutie.

Bovenstaande figuur geeft de LAN evolutie weer (vanuit het standpunt van een fiber fabrikant). Enkele opmerkingen:

- CAT5e heeft het iets langer uitgezonden als horizontale bekabeling... tot ong 2010
- Op CAT5e kan wel degelijk ook 1Gbps!
- De backbone/verticale bekabeling is niet noodzakelijk fiber... eerder CAT6a indien binnenshuis en tot 10Gbps

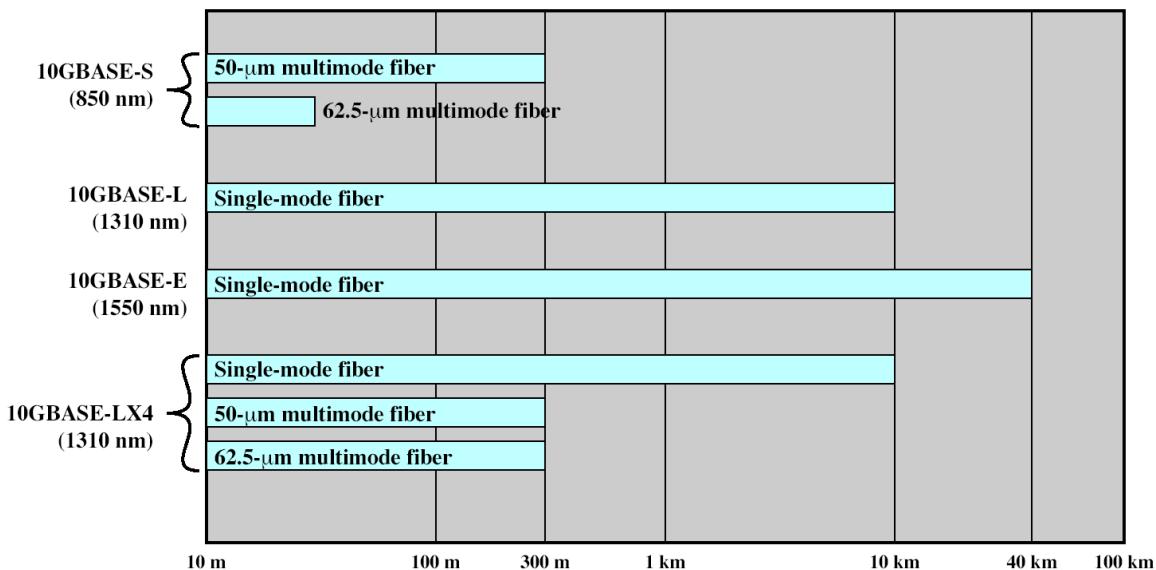
Men gebruikt nog steeds dezelfde frame sizes en frame formaten (maar nu enkel ge'switched', geen multiple access meer, geen CSMA/CD, geen 'hubs'). Voor het eerst stapt de ethernet standaard nu uit zijn LAN omgeving en biedt het ook connectiemogelijkheden naar WAN's. (SONET/SDH).

Er werden in eerste instantie 4 fysieke lagen gedefinieerd, voorlopig allen op fiber. De eerste 3 hiervan hebben een 'suboptie' : 'R' of 'W'. R versies worden gebruikt in LAN omgevingen en dienen aangesloten te worden op 'dark fiber', dit betekent dat er geen ander signaal of zendschema op de fiber aanwezig is. De signaalcodering is 64B/66B.

De W-versies of **WAN** versies zijn bedoeld voor SDH/SONET connecties, en volgen deze synchrone standaard, zie ook SONET/SDH p.122. De 4 fysieke lagen zijn :

- 10GBASE-S (Short) : gebruikt 850 nm licht op MMF fiber. Hiermee haalt men afstanden tot 300m, weliswaar enkel op de beste MMF fibers (26m...300m). versies zijn 10GBASE-SR en 10GBASE-SW.

- 10GBASE-L (Long) : ontwikkeld op **1310 nm** licht in SMF fiber. De afstanden zijn nu een pak groter : tot 10km. versies zijn 10GBASE-LR en 10GBASE-LW.
- 10GBASE-E (Extended) : gebaseerd op **1550 nm** licht in SMF fiber. Zoals de naam al laat vermoeden worden de afstanden nog groter : tot 40km (en zelfs verder volgens sommige fabrikanten). versies zijn 10GBASE-ER en 10GBASE-EW.
- 10GBASE-LX4 : ontwikkeld ook op 1310 nm licht in SMF of MMF fiber. De afstanden zijn nu eveneens tot 10km voor SMF, maar nu gebruikt men WDM (Wavelength Division Multiplexing) om de bitstroom te verdelen over 4 lichtbundels.



Figuur 121 10 Gigabit ethernet opties.

De 7 fiber versies van 10GBASE zijn dus : 10GBASE –SR / -LR / ER en 10GBASE-LX4 voor LAN's, en 10GBASE –SW / -LW / EW voor de WAN omgeving. De max. afstanden die hiermee kunnen bereikt worden vind je in bovenstaande figuur.

Op koper zijn er 2 versies:

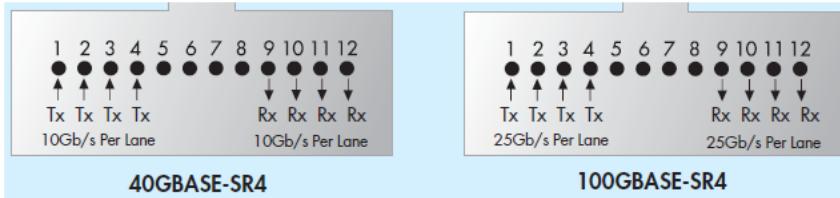
- 10GBASE-CR : op Twinax, 2 paren, 15m lang. Goedkoop. ± minder gebruikt
- 10GBASE-T: op U/STP CAT6 = 55m, CAT6a of CAT7 = 100m

10GBASE-T of IEEE 802.3an gebruikt een PAM-16 lijn code met een zelfcorrigerende reed-solomon code. Dezelfde code wordt ook gebruikt in recentere 2,5 en 5Gbps connecties over bestaande CAT5e en CAT6 bekabeling.

### 1.4.5 40/100 GIGABIT ETHERNET

Als lokale backbones anno '18 upgraden tot 10Gbps, dan moeten ISP backbones en WAN verbindingen eveneens evolueren. De standaard IEEE 802.3ba zag het licht in 2010.

Men maakt gebruik van 'multilane distribution', wat eigenlijk neerkomt op meerdere parallel geschakelde fysieke lijnen om aan de vereiste snelheid te komen. Dit kan gaan over effectief 4 UTP lijnen van bv. 10Gbps, 4 aparte fibers, of 4 verschillende golflengtes die met WDM over 1 fysieke fiber worden gestuurd.



Figuur 122 40- en 100 Gbase SR4.

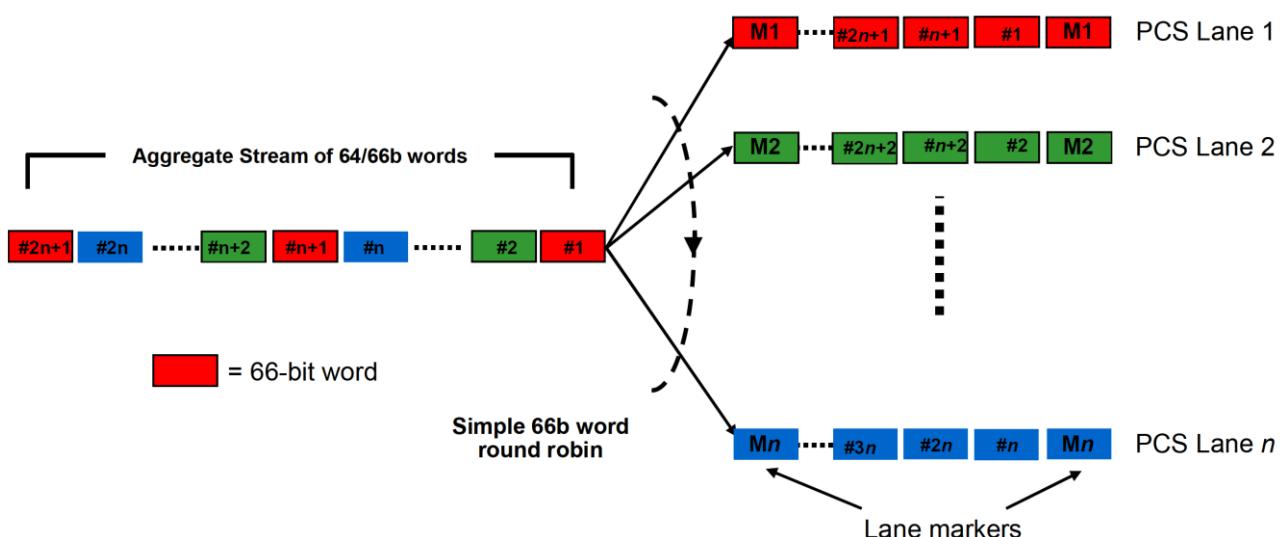
Elk apart kanaal wordt ook wel een 'virtual lane' genoemd, later PCS. In een eerste generatie kon 1 lane 10Gbps transporteren, in de 2<sup>e</sup> generatie 25Gbps. (Er bestaan dus ook versies van 10 lanes x 10Gbps om 100Gbps te transporteren, bv. 100GBASE-SR10.)

De bits op 1 PCS (Physical Coding Sublayer) lane worden 64B/66B gecodeerd, net zoals bv. 10Gbase,



Figuur 123 40- en 100 Gbase lijncodering 10Gbps per lane.

De 66-bit woorden worden met een simpel round robin schema verdeeld over de lanes, met periodisch een 'lane marker' om de lanes met elkaar te synchroniseren (of te 'aligneren').



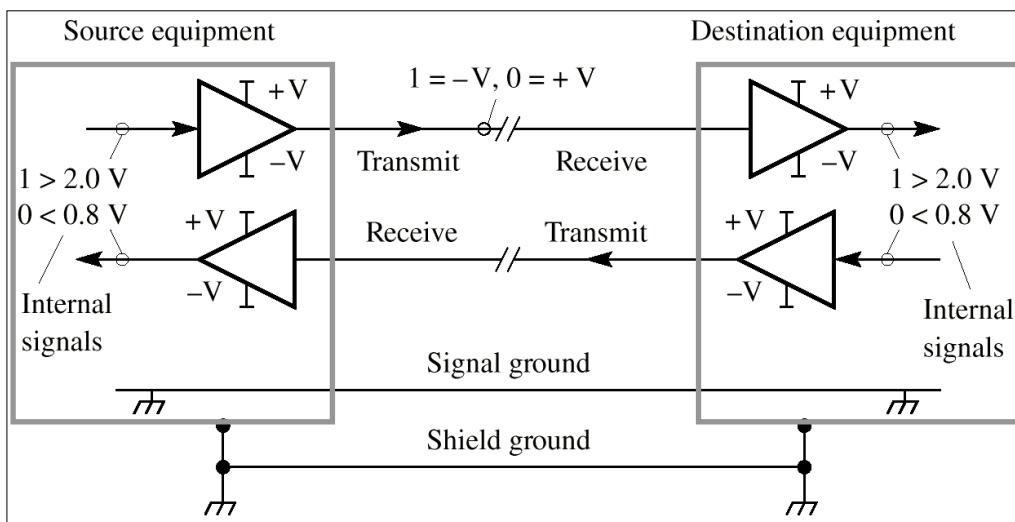
Figuur 124 40- en 100 Gbase lane concept.

Praktische uitvoeringen :

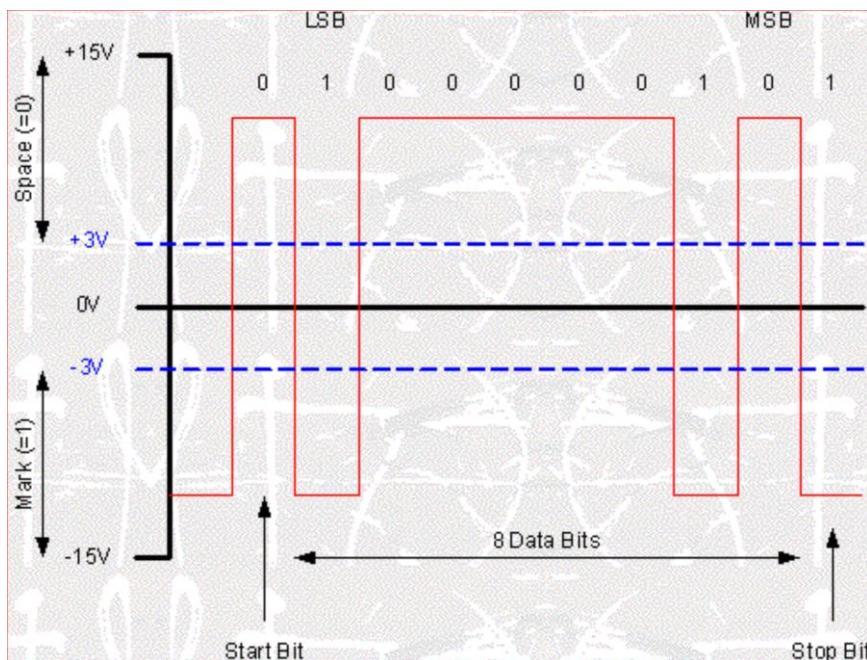
	40 Gigabit Ethernet	100 Gigabit Ethernet
<b>1m backplane</b>	40GBASE-KR4	
<b>10m copper cable</b>	40GBASE-CR4	100GBASE-CR10
<b>100m OM3 MMF</b>	40GBASE-SR4	100GBASE-SR10
<b>10km SMF</b>	40GBASE-LR4	100GBASE-LR4
<b>40km SMF</b>		100GBASE-ER4

### 1.4.6 RS 232

Een veel oudere standaard, het prille begin van communicatie. Hij definiert het aantal en de functies van de elektrische signalen, fysische afmetingen van de connector en pinconfiguratie. EIA noemde dit EIA-232 (A,B,C en D), ITU-T : V24, de signaal **niveaus** noemde men V28.



Figuur 125 RS232, V28 “unbalanced” signalen



Figuur 126 RS232 bitpatroon

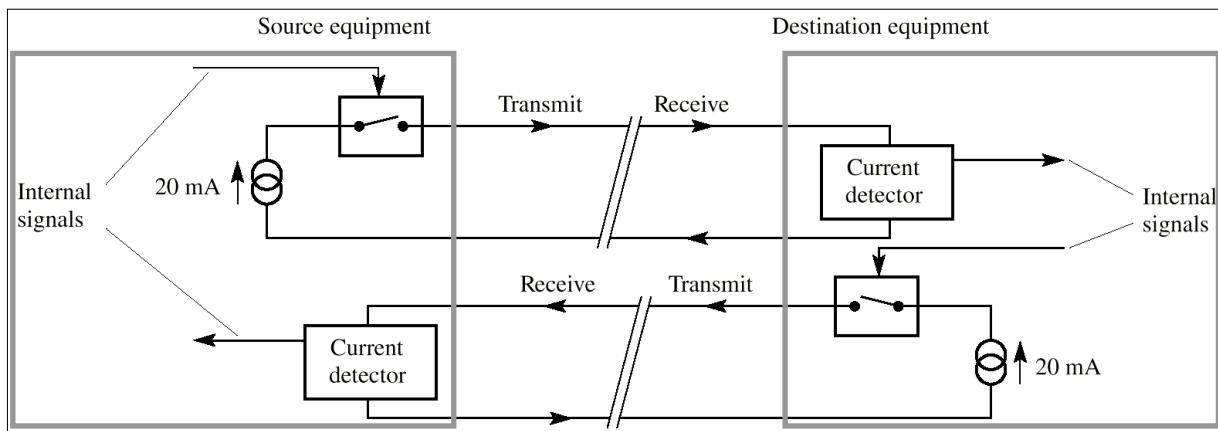
Elke byte (soms 7 bit) wordt voorafgegaan door een startbit en gevolgd door (evt. een pariteitsbit en) een stopbit. Het is een NRZ-L signaal, dus **zonder** bitsynchronisatie per bit → **A-synchroon**, enkel synchronisatie per byte bij de overgang van stop- naar start-bit. Zie ook Asynchrone transmissie op p.75.

Het gevolg is dat geen hoge snelheden kunnen behaald worden: max. 19200 bps volgens de standaard (sommige uarts tot 115200 bps). Ook de afstanden waren beperkt: max 15m. De signalen worden gerefereerd t.o.v. een ‘common ground’, = een gezamenlijk ‘teruglooppad’ voor de stromen van beide datalijnen (en controlelijnen), zie ook Figuur 125.

## 20MA CURRENT LOOP

In het bedrijfsleven wilde men deze standaard gebruiken voor sensor- en actuatorbussen, maar kon men niet leven met de ruisgevoeligheid: men creeerde een stroomkring waarbij stromen (in 1 paar) in tegengestelde richting liepen en bijgevolg veel minder gevoelig waren voor ruis. (Merkop, geen gezamenlijke ground meer!).

Men gebruikte dezelfde UARTS (= zelfde bitpatroon), maar andere line drivers → langere afstanden. Dit systeem is weliswaar niet gestandaardiseerd, vandaar een beetje raar in deze cursus.



Figuur 127 20mA current loop signalen

Om een en ander elektrisch te scheiden worden vaak nog optocouplers gebruikt als ontvanger en zender → 20mA stroom stuurt led rechtstreeks.

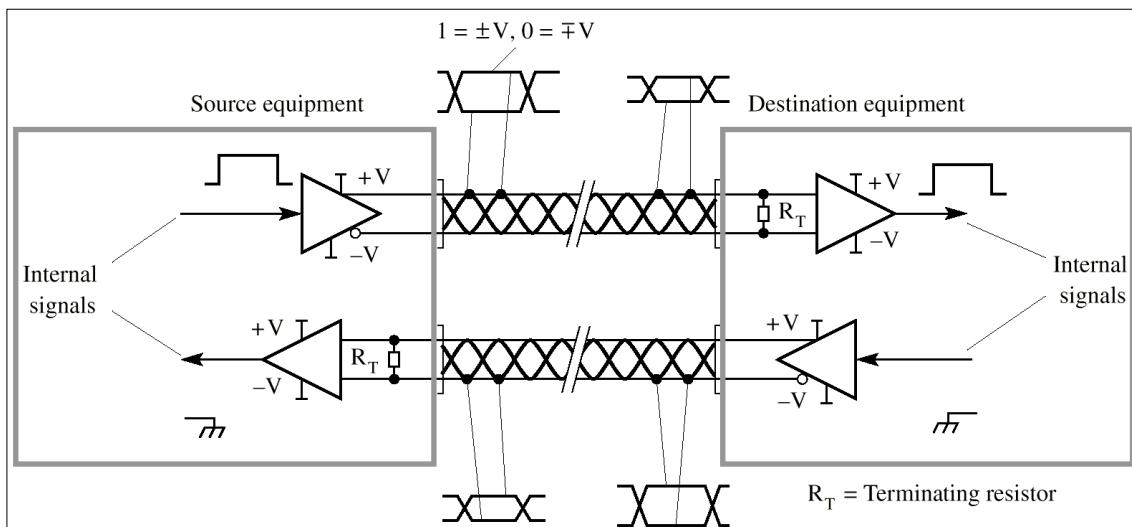
### 1.4.7 RS422 A / V11 EN RS485

Een gestandardiseerde verbetering van RS232 is RS422 die per definitie computers **P2P** verbond, net zoals RS232. De bedoeling is de **afstand** tussen de computers en de **datasnelheid** te verhogen. Men stapt ook nu af van het ‘common ground’ principe maar stuurt het signaal **differential** (ook wel balanced genoemd) op een UTP kabel.

Een differential transmitter wekt signalen op met tegengestelde polariteit.

Vb. een  $1^L$  wordt voorgesteld door  $+5V$  t.o.v.  $-5V$

. een  $0^L$  wordt voorgesteld door  $-5V$  t.o.v.  $+5V$



Figuur 128 RS 422 / V11 “balanced” signalen

Dit zijn nog steeds NRZ-L signalen, te vergelijken met RS232, Figuur 126 op p.107. A-synchroon met start- en stopbits.

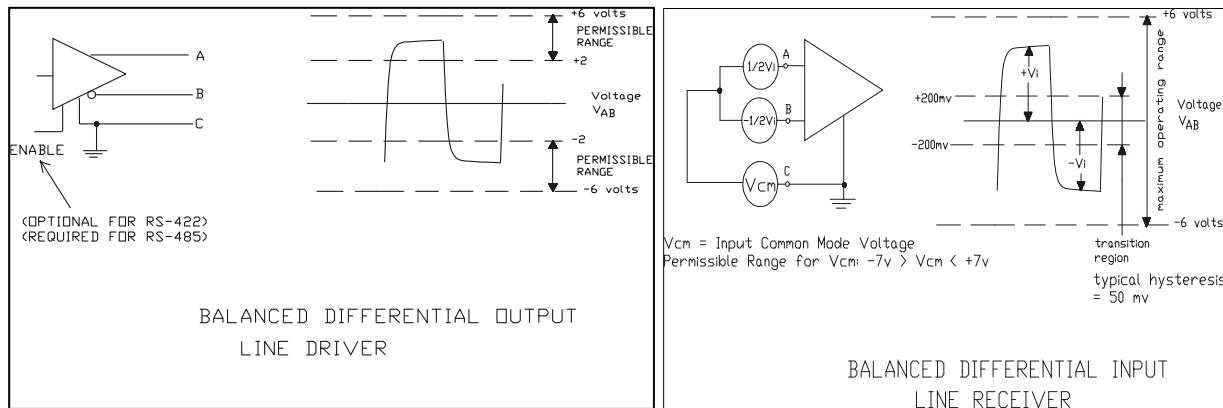
De ‘differential receiver’ is enkel gevoelig voor verschilspanningen, en ‘triggert’ niet tov. een vast liggend referentieniveau gebaseerd op een ‘common ground’. Dit maakt hem veel ongevoeliger voor ruis opgewekt in **beide** draden → goede “common mode rejection”.

Zo worden afstanden en bitrates aanzienlijk verhoogd (vgl. met RS232 : 15m, 19200 bps) :

Signaal type	Lengte	Max. transmissie (bps)
RS –422A / V11 Niet afgesloten / unterminated	10	1 000 000
	100	100 000
	1000	10 000
RS –422A / V11 afgesloten / terminated	10	10 000 000
	100	1 000 000
	1000	100 000

Figuur 129 maximum afstand/bitrate voor RS422.

Belangrijk is ook de afsluiting op karakteristieke impedantie door  $R_T$ . (bv.  $100 \Omega$ ). We zien in bovenstaande tabel dat afstanden of bitrates met een factor **10** verhogen bij afsluiting.

**Spanningsvormen :**

Figuur 130 Spanningen op een RS422/485 lijn.

De **zender** stuurt :

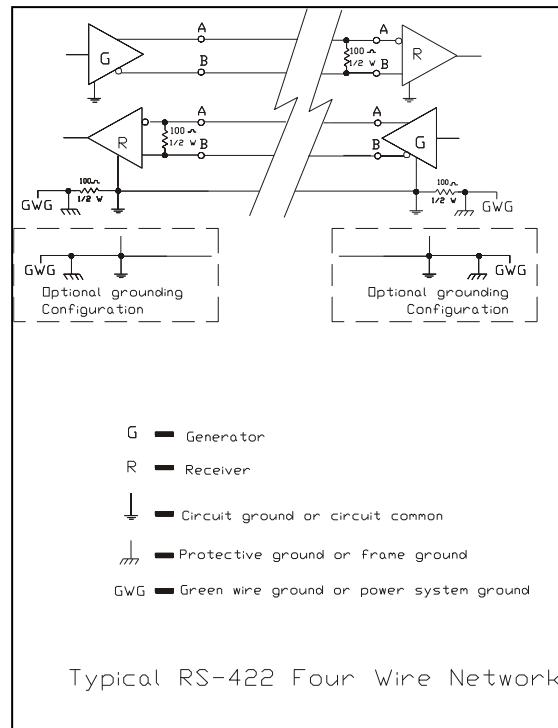
$$T_x \quad V_{AB} = +/- 2.. 6V,$$

terwijl de **ontvanger** zijn signaal verzwakt mag zijn :

$$R_x : \quad V_{AB} = +/- 0,2.. 6V$$

Verzwakking door de transmissielijn is dus toegestaan.

Common mode spanningen tot +/- 7V worden toegestaan.

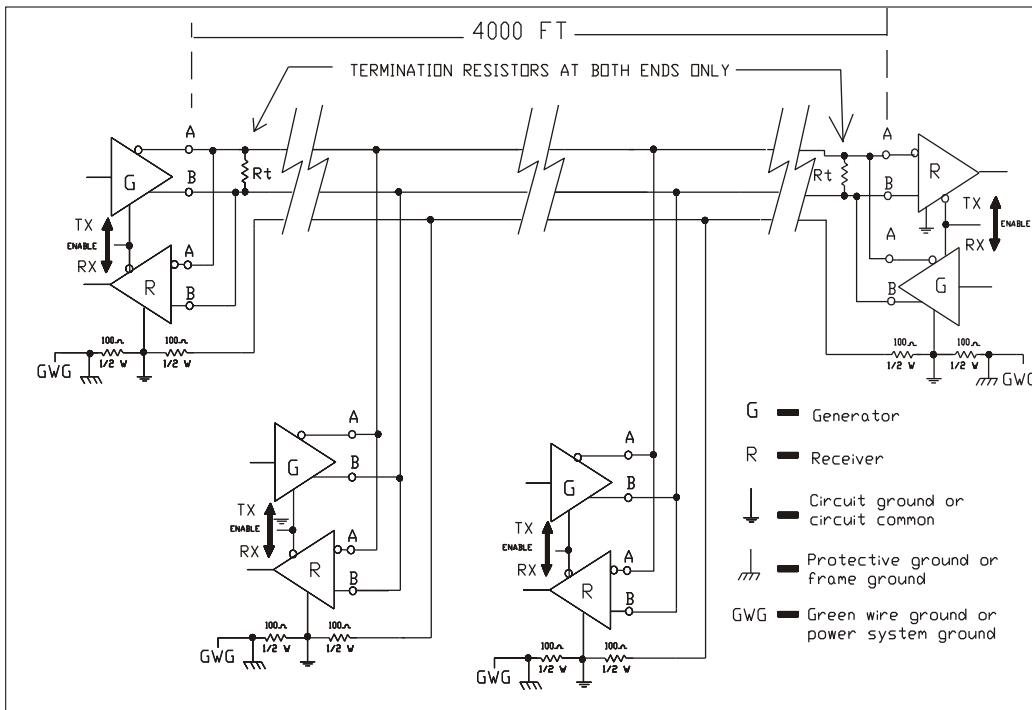
Een **full duplex** communicatie wordt dan :

Figuur 131 RS422 – four-wire Network

## RS485

RS 485 is in feite een **bus-versie** van RS422 (RS422= P2P) en wordt veel gebruikt in de automatisering wereld voor sensor en actuator bussen. (Voor hogere snelheden heeft ethernet ook daar reeds lang zijn intrede gedaan: 'industrial ethernet').

In het geval van RS485 "multidrop" ('multiple access') moeten natuurlijk de verschillende zenders kunnen afgekoppeld worden door een '**enable**' controle zie Figuur 130. Één zender moet tot 32 ontvangers kunnen sturen (sommige referenties spreken over 64 of 128 nodes).

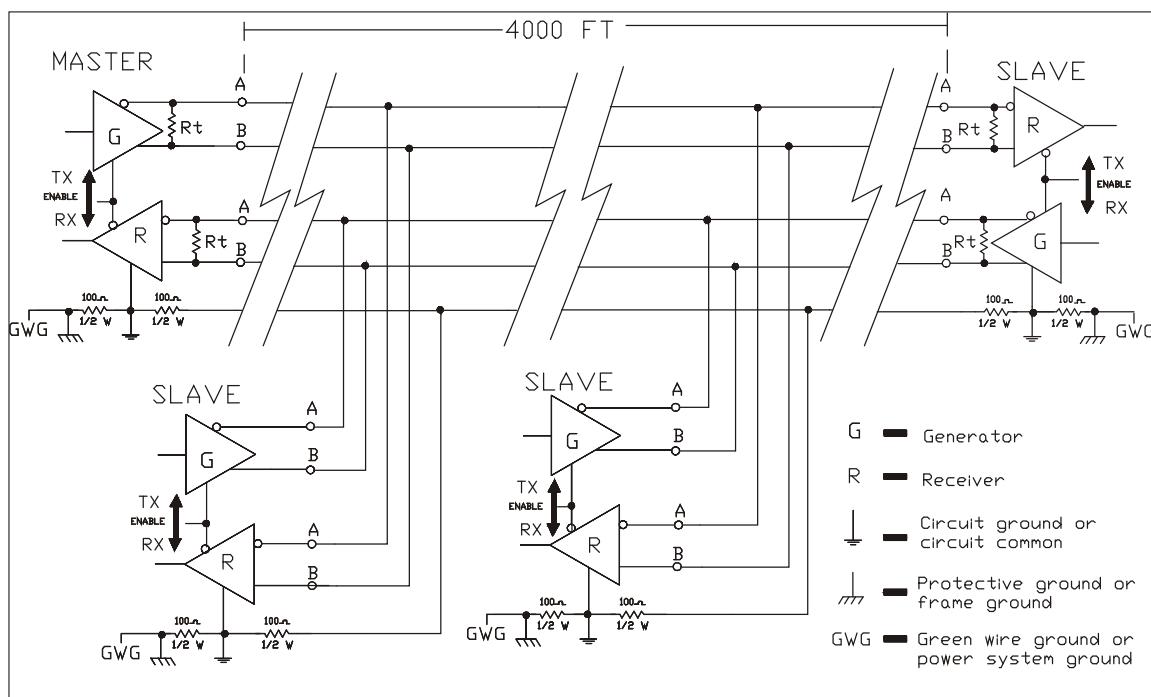


Figuur 132 RS485 – two-wire multidrop Network

Let ook op de afsluitweerstanden op de **uiterste nodes** van de lijn en niet bij elk 'drop'punt in het midden. De signaal-grondlijn dient enkel om de common-mode spanning binnen de +/- 7v te houden.

De software besturing heeft tot taak om het 'token' door te geven aan de node die data mag verzenden op de bus. Er kunnen wel meerdere 'listeners' zijn.

De software wordt sterk vereenvoudigd als men overgaat naar een **4-draads** configuratie. Hier heeft men steeds **1 master** en **verschillende slaves** die kunnen antwoorden op het andere paar. Er is geen communicatie mogelijk tussen slaves onderling.



Figuur 133 RS485 – four-wire multidrop Network

In een RS232 naar RS485 omzetter wordt de RTS lijn (RS232) gebruikt om de ‘transmitter-enable’ van de RS485 te sturen.

### AFSLUITING

Wanneer transmissielijnen niet goed worden afgesloten wordt er een deel van de golf teruggekaatst op de lijn. De afsluitweerstanden zelf vormen echter ook een **belasting** voor het netwerk. Een vuistregel voor al of niet afsluiten is :

“als de voortplantingssnelheid op de lijn veel kleiner is dan 1 bittijd, is er geen afsluiting nodig”

De reflecties sterven uit na enkele ‘round trip’s’ op de lijn. Aangezien de UART de lijn ‘sampled’ in het midden van de bit moeten reflecties uitgestorven zijn.

Bv. een lijn van 4000ft = 1200m. (1ft = 30,48cm).

De voortplantingssnelheid over koperbedrading is  $\approx 2/3 C$  (tss 2/3 en ¼ van de lichtsnelheid)

$$\rightarrow 1 \text{ trip} = 1200 \text{m} / (0,66 * 3 * 10^8) = 6 \mu\text{s} \rightarrow 3 \text{ trips} = 18 \mu\text{s}$$

$$\rightarrow 9600 \text{ baud} : 1 \text{ bittijd} = 1/9600 = 104 \mu\text{s} \rightarrow \frac{1}{2} \text{ bittijd} = 52 \mu\text{s} >> 18 \mu\text{s} \rightarrow \text{uitgedeind}$$

↳ niet noodzakelijk afsluiten.

$$\rightarrow 115200 \text{ baud}: 1 \text{ bittijd} = 1/115200 = 8,6 \mu\text{s} \rightarrow \frac{1}{2} \text{ bittijd} = 4,3 \mu\text{s} << 18 \mu\text{s} \rightarrow \text{niet uitgedeind}$$

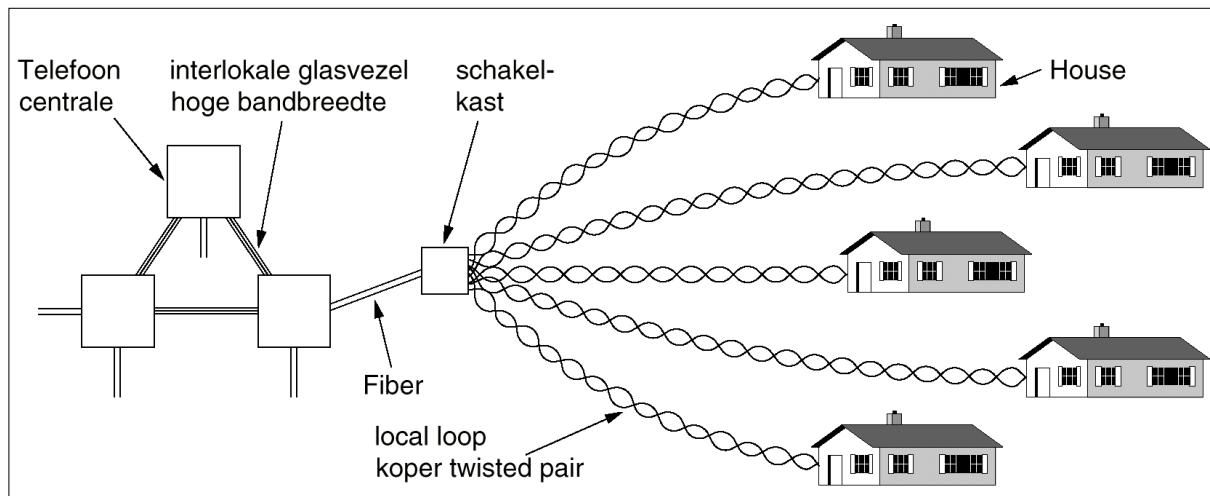
↳ afsluiten !!

## 1.5 HET TELEFOONSSTEEEM

Aangezien de totale cursus de nadruk legt op computernetwerken, telefonienetwerken en de combinatie van de 2 besteden we een aparte paragraaf uitsluitend over telefonie. Bovendien is het zo dat computers die wereldwijd met elkaar in verbinding wensen te komen gebruik kunnen maken van de infrastructuur van publieke telefoniebedrijven, zowel in de vorm van gehuurde lijnen als van geschakelde circuits. Het is dus van belang hiervan de karakteristieke eigenschappen te kennen.

Bovendien is het zo dat IP-telephony en VoIP (Voice over IP) steeds meer dominant wordt als alternatief voor dit systeem, maar er wel mee moet koppelen. Het is dus ook hier belangrijk om weten hoe deze wereld werkt.

In praktisch alle landen is het telefoonnet volledig **gedigitaliseerd** op de uiteinden, de 'local loops' na. De analoge spraaksignalen worden in de schakelkasten **omgezet** naar **digitaal** en over de 'trunks' gemultiplext in TDM (Time Division Multiplexing).

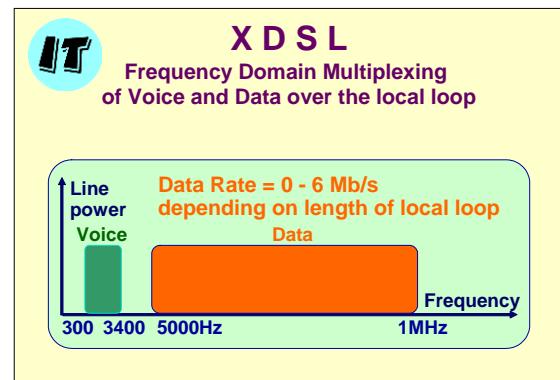


Figuur 134 Telefonië aansluiting local loop.

Bovendien zijn deze laatste km van de local loop ook **digitaal** te verkrijgen → **ISDN**. In dat geval is de local loop ook **digitaal** (2B1Q!) en wordt er verbonden met een **digitale** schakelkast. De spraak moet nu omgezet worden in digitale vorm, ofwel in het (ISDN) handtoestel, ofwel in de binnenuitscentrale. (De manier waarop is bepalend voor de PTT infrastructuur.)

Spraak transporteren gebeurt reeds eeuwen volgens hetzelfde principe : de allerhoogste en –laagste tonen zijn onnoodig om een verstaanbaar gesprek te voeren en worden bijgevolg weggefilterd om storingen te vermijden. Een bruikbaar spraakkanaal heeft een spectrum van **300 tot 3400Hz**, zie ook volgende figuur. (afgerond **4KHz**, wat een sample frekwentie voor digitalisatie oplevert van **8KHz**, theorema van Shannon.)

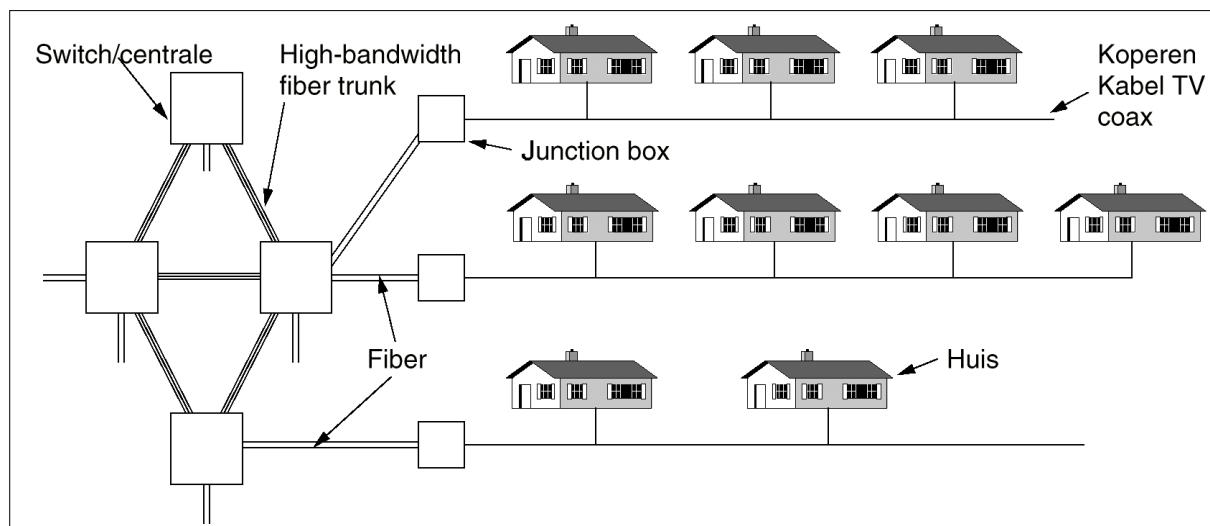
Telefonie - UTP kabels worden dikwijls uitgevoerd met 25 tot 100 paren in 1 bundel, met bovendien twistlengtes van enkele cm, over afstanden tot enkele km. Dit heeft ernstige repercussies op de bruikbare bandbreedte : ong 1MHz! Maar geen erg : een telefoongesprek vraagt slechts 4KHz!



Figuur 135 Frequentieband voor xDSL.

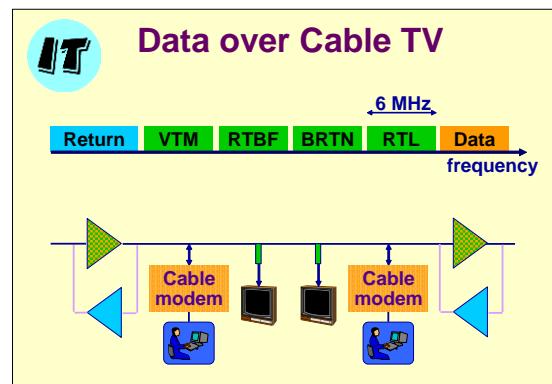
Deze grote onbenutte bandbreedte heeft xDSL ontwerpers (alcatel- Antwerpen!) ertoe aangezet om dit voor andere doeleinden te gebruiken en hierop tot 2 ... 8 ... tot 15 Mbps in ADSL en zelfs 100 Mbps in VDSL2 te vervoeren! Merk ook op dat vanaf de schakelkast fysisch er een **STER**-bekabeling ligt tot aan de gebruiker, een groot voordeel tov. kabeldistributie. Als je ziet dat deze schakelkasten toegeleverd worden met fiber zijn de mogelijkheden natuurlijk onbegrensd.

TV-distributie-maatschappijen (=CATV) passen i.t.t. hun concurrent hierboven, de PTT's, een **bus**-bekabeling toe. In dit geval moet de beschikbare bandbreedte gedeeld worden onder de gebruikers van de bus.



Figuur 136 het kabel TV netwerk, HFC Hybrid Fiber Coax bekabeling.

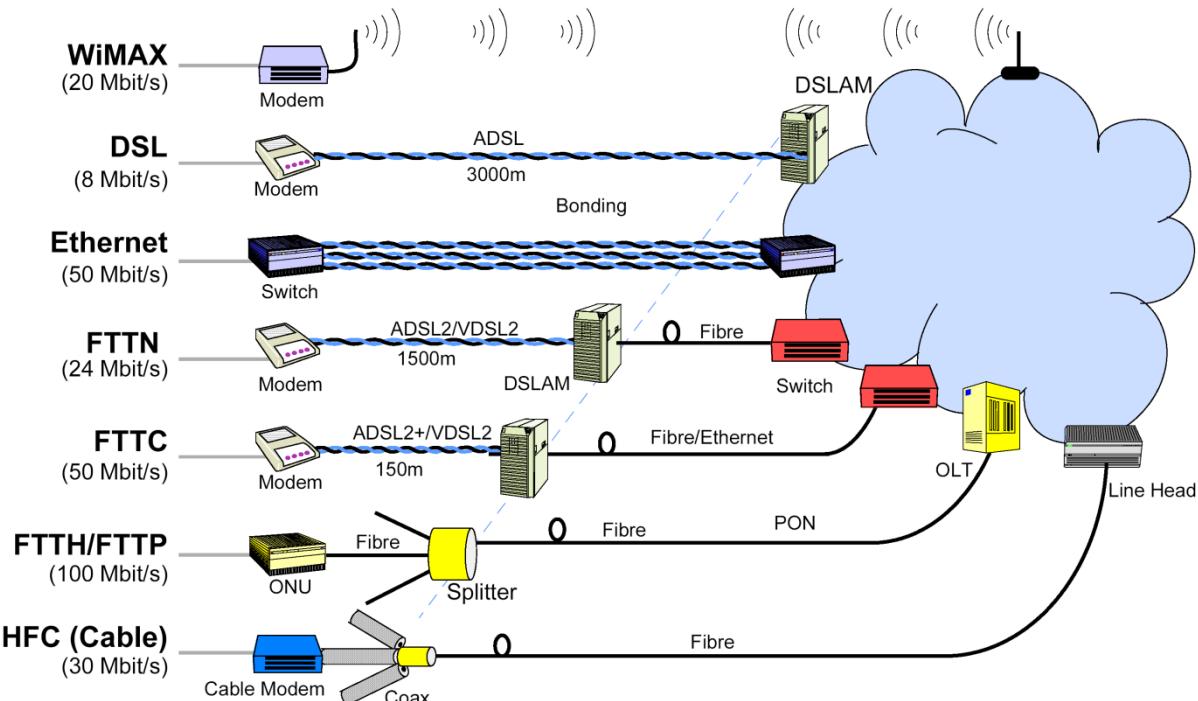
De CATV-maatschappijen hebben een hoog-kwaliteits-coax-netwerk als LocalLoop, dat eveneens tot bij de gebruiker reikt. Aangezien de gebruikte (dikke) coax een bruikbare bandbreedte van 88MHz tot ong. 800MHz heeft, wint men hier de bandbreedte-slag. Zie ook Figuur 63 op p.62



Figuur 137 Bandbreedte gebruik bij HFC.

Ze moduleren hierop verschillende kanalen die bij de gebruiker terug uitgefilterd moeten worden → breedband. Hierbij voorzien ze een stuk bandbreedte van hun medium om **data** over te verzenden. Op te merken valt dat 'downstream' de bandbreedte veel hoger is dan 'upstream'. Een en ander wordt gestandaardiseerd in IEEE 802.14, men spreekt van **HFC** of '**Hybrid Fiber Coax**' structuren.

De meest luxueuze oplossing zou zijn **FTTH**, Fiber To The Home, maar dat zou enorme investeringen vragen voor noden die (nog?) niet aanwezig zijn en die bijgevolg niet worden terugbetaald. Alternatieven op deze 2 koplopers zijn WiMAX : een draadloze local loop en andere, zie onder.



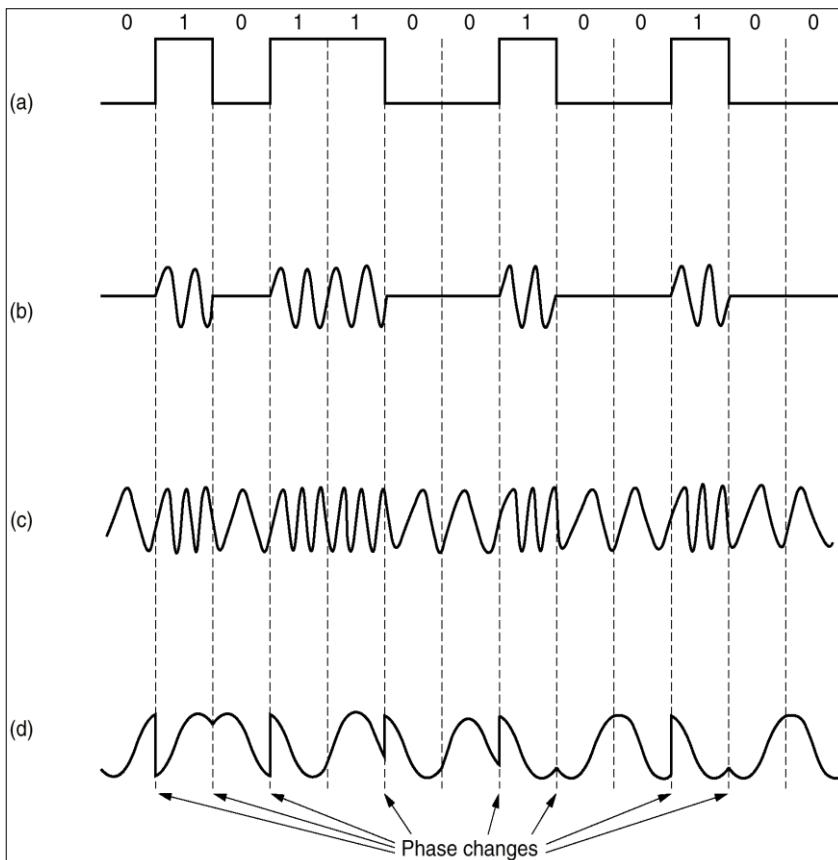
Figuur 138 toekomstbeeld voor de 'last mile'.

FTTN = 'Fiber To The Node/Neighborhood'

PON = "Passive Optical Network"

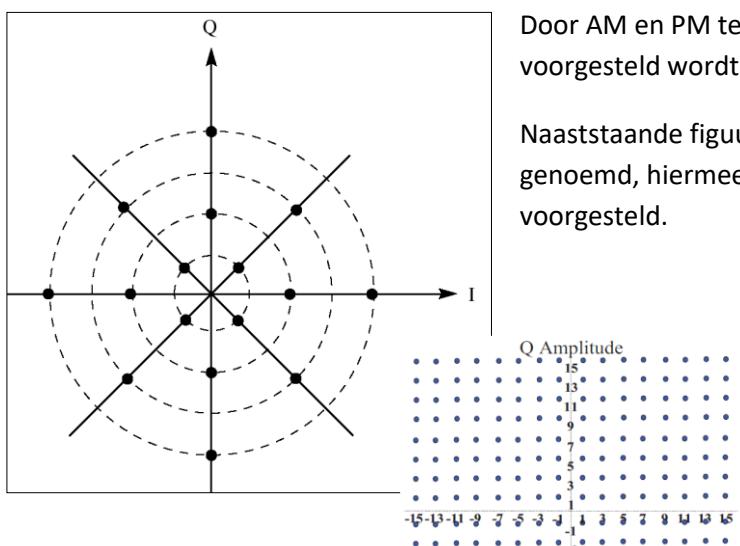
ONU = "Optical Network Unit"

### 1.5.1 ANALOGE TELEFONIELIJNEN



De bandbreedte van een analoog telefoniekanaal is beperkt : van 400Hz tot 3400Hz, m.a.w.  
 $B=4000\text{Hz}$  met guard band. Om hierover digitale bits te sturen moet men bijgevolg moduleren aangezien continue reeksen 1-en of 0-en niet worden doorgezonden. Men kent 3 soorten modulatie : b) **AM**, c) **FM** en d) **PM**.

Figuur 139 Modulatievormen : b)AM c) FM en d) PM.



Door AM en PM te combineren komt men **QAM**, wat voorgesteld wordt door een fasediagram.

Naaststaande figuur is een 16 punts-constellatie ook 16-QAM genoemd, hiermee kunnen 4 bits in 1 signaalelement worden voorgesteld.

Figuur 140 16 en 256- QAM constellatie.

Veel gebruikte coderingen zijn constellatiepunten, 8 bits info

ook QAM 64 en zelfs QAM256 met ... 256 per punt..

Zo zal een (oude) V32<sup>bis</sup> 28,8kbps modem een **128-QAM** codering kennen waarbij de 7 bitscodering 6 databits en 1 trellisbit voorstellen.

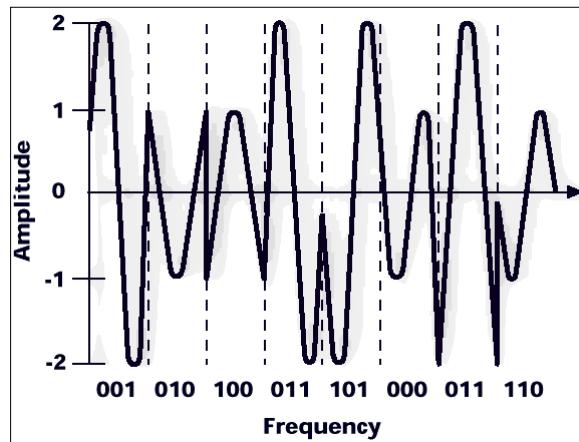
Er zijn degelijke DSP's nodig om deze convolutiecode te decoderen.

Om dit praktisch voor te stellen bekijken we een 8-QAM : een 3 bitscodering. Het komt overeen met een 16-QAM constellatie waar de diagonale assen zijn verwijderd, enkel de orthogonale blijven over.

Stel dat we de sequentie 0 0 1 0 1 0 1 0 0 0 1 1 1 0 1 0 0 0 0 1 1 1 1 0 moeten transporteren, dan wordt dit verdeeld in 3-bit groepen die elk 1 punt in de constellatie voorstellen. Volgens de tabel kan men dan de amplitude en fasedraaiing vinden die met die punt overeenkomt.

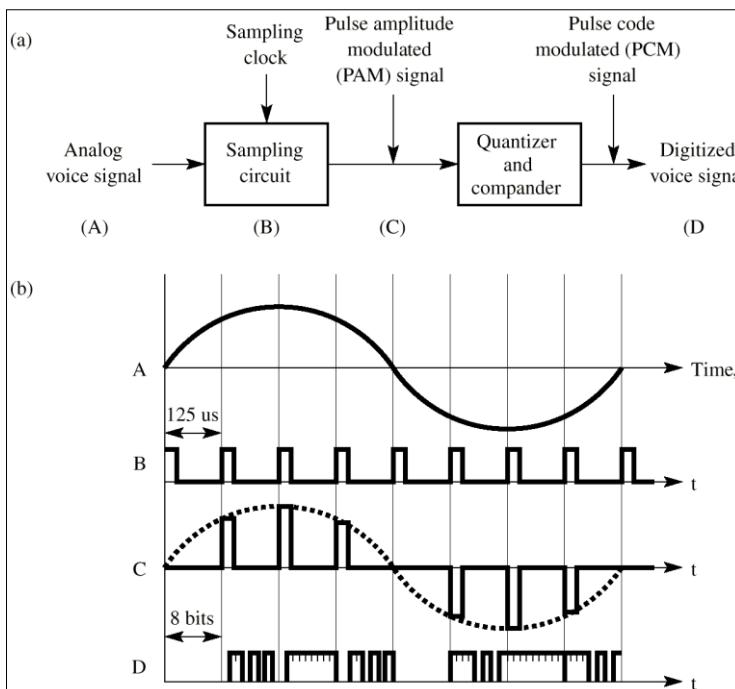
Figuur 141 Een 8-QAM golfvorm

Bit Value	Amplitude	Phase Shift
000	1	None
001	2	None
010	1	1/4
011	2	1/4
100	1	1/2
101	2	1/2
110	1	3/4
111	2	3/4



## 1.5.2 DIGITALE TELEFONIELIJNEN

### DIGITALISATIE VAN GELUID.



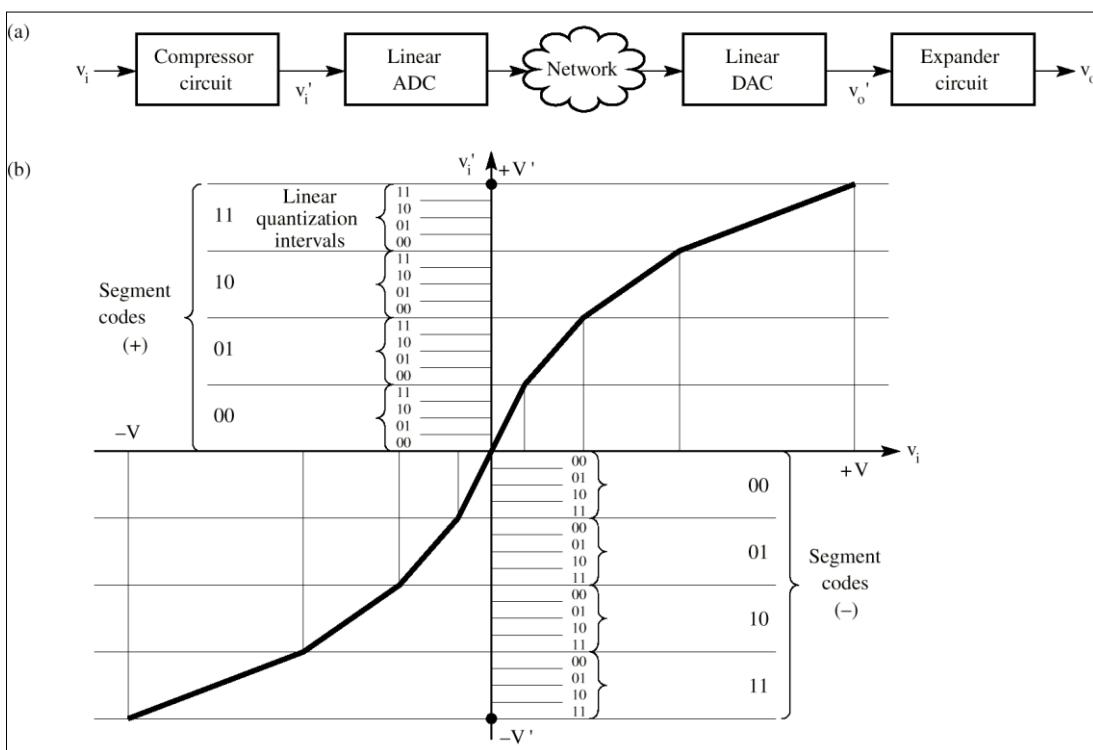
Als basiskwaliteit is nog steeds geluid met een bandbreedte van **4KHz** genomen. Het Nyquist theorema leert ons dan dat er moet gesampled worden met min **8KHz** → elke **125 µs**.

We verkrijgen een **PAM** (Pulse Amplitude Modulatie) signaal zoals te zien is in golfvorm **C**.

Dit PAM signaal wordt **gequantiseerd** naar 8 bits (1 tekenbit) → **PCM**, wat meteen de eenheid van transmissie-capaciteit oplevert : **64kbps** = 8bit x 8000/s.

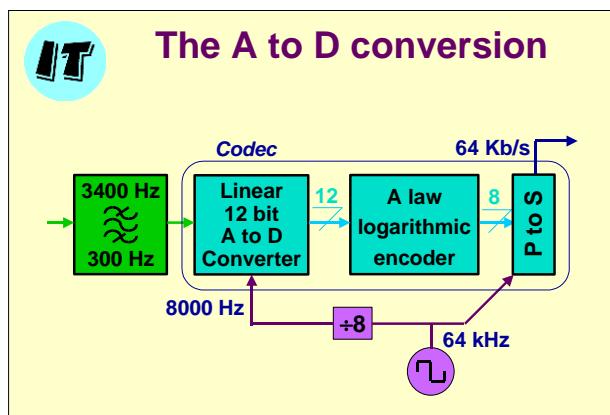
Figuur 142 Digitalisatie van geluid.

Natuurlijk gevolg van het quantiseren is het opwekken van **quantiseringsruis** : de discrete niveaus verliezen nauwkeurigheid binnenin 1 interval, en dit is niet meer te herstellen. Deze ruis is nu des te erger naarmate het signaal **kleiner** wordt. Het menselijk oor is echter gevoeliger voor zwakkere geluiden dan voor lawaai. Daarom wordt de quantisering **niet lineair** gemaakt en zal men zwakkere amplitudes nauwkeuriger omzetten dan hoge amplitudes.



Figuur 143 Companding van geluidssignalen.

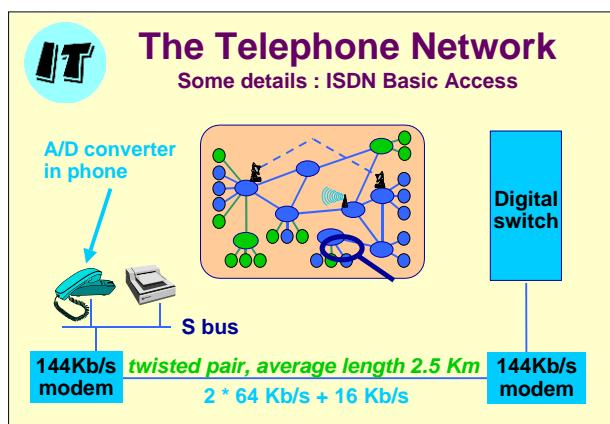
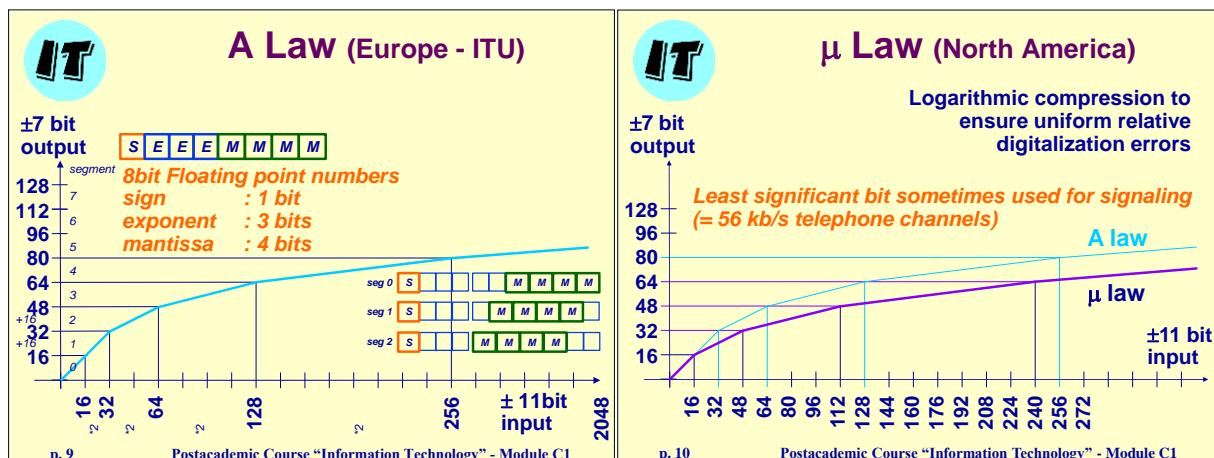
Zoals te zien op Figuur 143 wordt het signaal eerst door een **compressor** gestuurd voor er lineair wordt geconverteerd. Aan de andere kant volgt er natuurlijk een **expander**. De gezamenlijke bewerking wordt **companding** genoemd. Om het principe aan te geven wordt er gewerkt met een 5 bit code → 1 tekenbit, 2 bit voor de **segmenten** (4) en 2 bit voor de **intervallen**.



Praktisch werd dit in de eerste PCM codecs (coder/decoder) analoog uitgevoerd, nu gebeurt de companding techniek echter **digitaal**: er wordt effectief geconverteerd met 12 bit, waarvan na compressie 8 bit overblijft. Er zijn 2 compressiekarakteristieken gestandardiseerd door ITU-T in de standaard G.711 :  $\mu$ -*law* (gebruikt in Noord-Amerika en Japan) en **A-law** in Europa. Beide vragen een bitrate van 64Kbps.

Figuur 144 A/D conversie van een spraakkanaal.

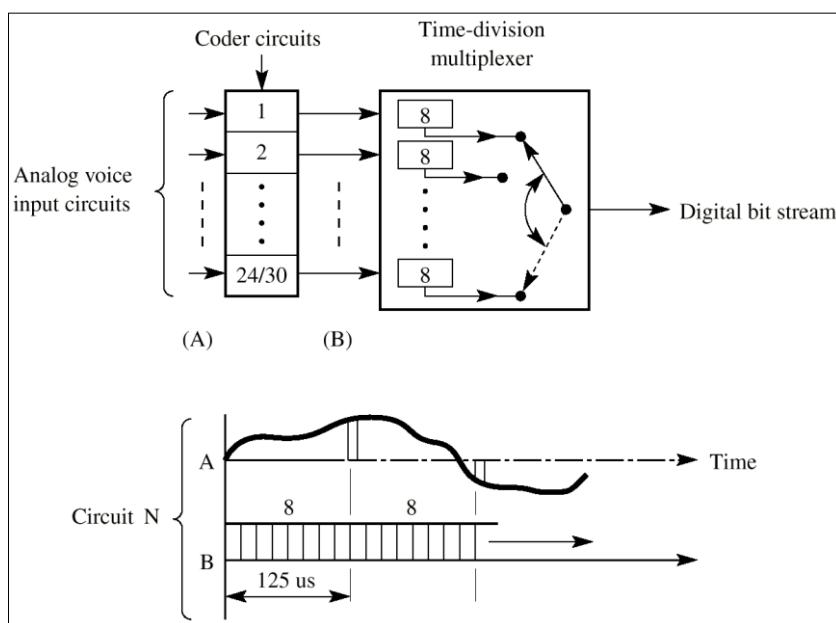
Figuur 145 A law en  $\mu$  law companding.



Een uiteindelijke ISDN aansluiting voorziet **2** mogelijke spraakkanalen van 64Kbps + 1 signaleringskanaal van 16Kbps tot aan de schakelkast. (Belgacom : Twin)

Figuur 146 ISDN basic access aansluiting.

## TDM - MULTIPLEXING



Op de 'trunks' worden de gesprekken (64Kbps) gemultiplext volgens TDM.

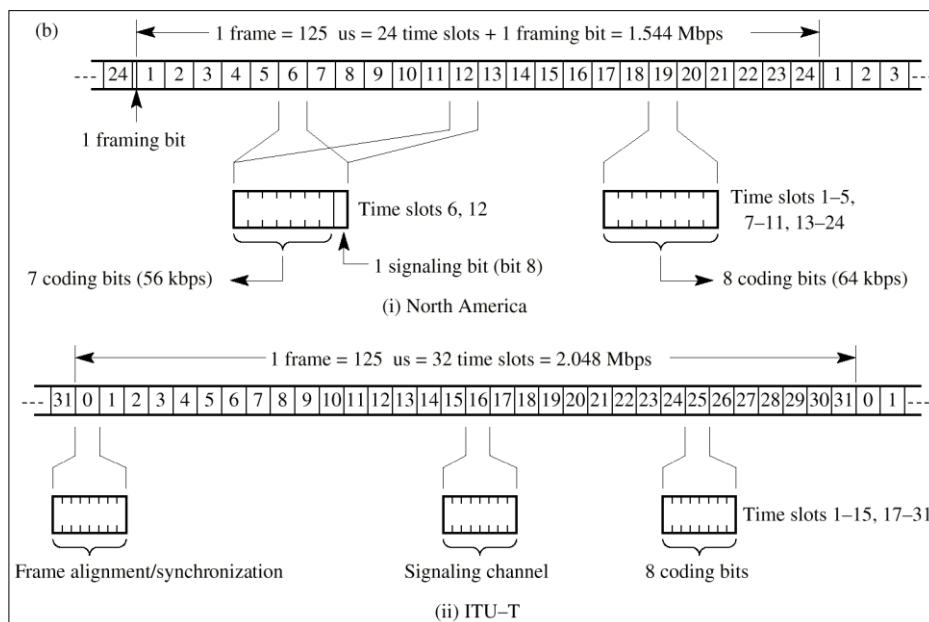
In USA en Japan zijn dat er **24**, in Europa → ITU-T : **30**.

(Als het analoge lijnen zijn die binnenkomen moeten ze eerst nog worden omgezet.)

Elke 125 μs ontstaat op een kanaal een sample van 8 bit → bijgevolg is ook een frame 125 μs.

Figuur 147 TDM mux van spraakkanalen.

De bytes worden door een multiplexer 'verzameld' en achter elkaar op een trunklijn geplaatst. Een MUX is rondgedraaid in 125μs, want dan staat de volgende byte reeds klaar in elk kanaal.



**USA** :  $24 \times 64\text{kbs} = 1,544\text{Mbps} = \text{T1 or DS1 link genoemd.}$

In het begin van het frame is er 1 framing bit voor frame synchronisatie, en op slot 6 en 12 is er 1 signaleringsbit.

**EUR** : ITU-T voor-ziet **32** tijdsloten, 1 voor synchronisatie, 1 voor signalering.

Figuur 148 T1 en E1 frames.

Een **E1** link heeft bijgevolg een bit rate van  $32 \times 64\text{Kbps} = \mathbf{2,048 \text{ Mbps}}$ , goed voor **30** gesprekken, geschikt voor een PRI interface van ISDN, zie ook de cursus CO – PDN (Connection Oriented Public Data Networks).

Hogere bit rates voor de digitale carriers worden bekomen door meerdere van deze frames nogmaals te multiplexen.

	<i>Link</i>	<i>Bit rate (Mbps)</i>	<i># spraak kanalen</i>
<b>USA</b>	<b>T1 / DS1</b>	1,544	24
	<b>T2 / DS2</b>	6,312	96
	<b>T3 / DS3</b>	44,736	672
	<b>DS4E</b>	139,264	1920
	<b>T4 / DS4</b>	274,176	4032
<b>EUR</b>	<b>E1</b>	2,048	30
	<b>E2</b>	8,448	120
	<b>E3</b>	34,368	480
	<b>E4</b>	139,264	1920
	<b>E5</b>	565,148	7680

Dit gaf echter complicaties aangezien de verschillende klokken van de lagere orde MUXen niet 100% perfect synchroon liepen. Men moest een aantal ‘justification bits’ invoegen om dit op te vangen → bijna synchroon of **‘Plesiochronous’ Digital Hierarchy (PDH)**

Zoek de standaarden G703/G704/G705 = PDH.

Een bijkomend probleem was de moeilijke ‘Add en Drop Multiplexer’, wat uiteindelijk uitmondde in een nieuwe evolutie :SDH.

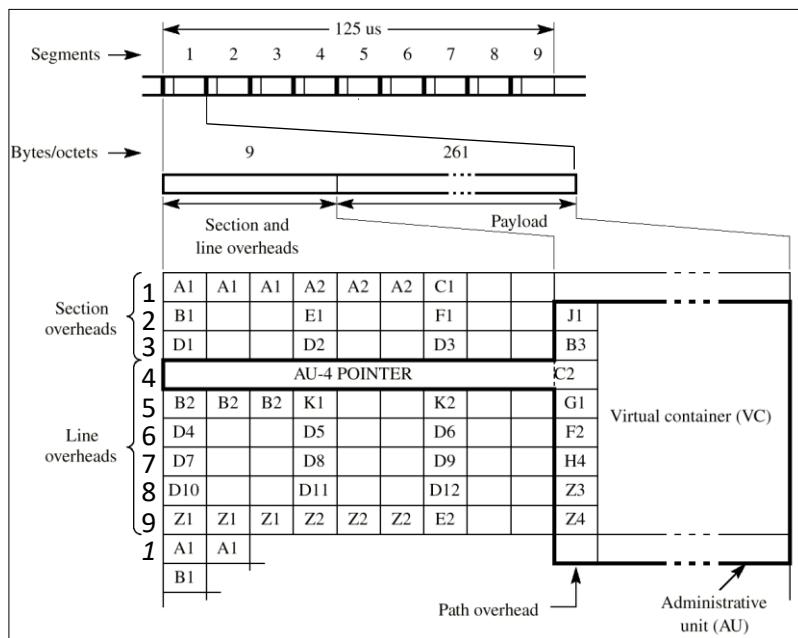
Tabel 3 USA en internationale TDM carrier standaarden

## SONET/SDH

In 1988 ontstond dan het idee van **B-ISDN** wat voor elke gebruiker een toegang voorzag van **155Mbps** tot 622Mbps. Dit lijkt moeilijk op te lossen met bovenstaande trunks. Om de problemen van PDH op te lossen, en tegelijk te voldoen aan de stijgende vraag naar bandbreedte ontwikkelde BellCore in de VS een **optische transmissie-interface** : SONET : SYNchronous Optical Network, gestandardiseerd door... uiteraard ANSI. Een compatibele versie, **SDH** : **Synchronous digital hierarchy**, wordt iets later gestandardiseerd door ITU-T in recommendation G.707.

Het transmissiemedium is enkel **fiber** en alle toestellen (switchen) worden gesynchroniseerd door 1 'masterclock' met een nauwkeurigheid tot 1 op  $10^9$ . De bits worden op een SDH lijn op uiterst exacte tijdstippen verstuurd. Er worden zelfs 'dummy' frames verstuurd bij gebrek aan bruikbare informatie → **'synchrone'** transmissie : **'Synchronous** Optical NETwork / Digital Hierarchy .

De basis transport eenheid noemt men in SDH nu STM-*n*, Synchronous Transport Module, met *n* als hierarchie aanduiding. Zo'n module wordt, i.t.t. een frame, 2 dimensioneel voorgesteld.

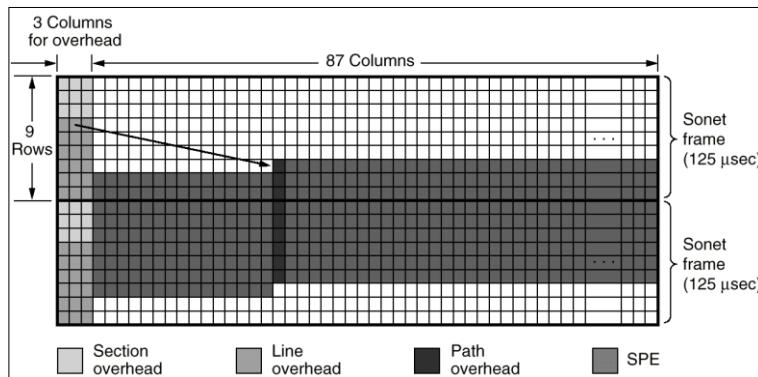


Een STM-1 module wordt elke **125μs** (= sample frekwentie codecs) opgestuurd en bevat een matrix van  $270 \times 9 = 2430$  bytes → bit rate = **155,520 Mbps**.

Een frame bevat negen 270-byte segmenten die elk een 9 byte header en 261 byte payload bevatten.

De eerste 9 (byte) kolommen vormen de header, wat er 261 overlaat voor de 'payload'.

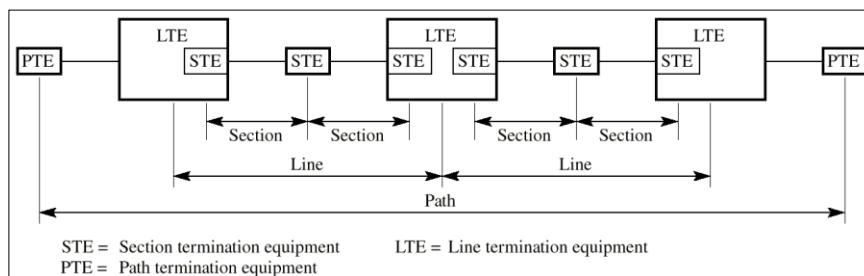
Figuur 149 Een SDH STM-1 module.



We zien dat de SPE, Synchronous Payload Envelope, overal mag beginnen, zodanig dat bvb. tijdens de opbouw van een dummy frame, een payload die bij de bron aankomt, in het frame kan worden geplaatst zodat het niet moet bewaard worden. (Een OC-1 frame van sonet = 1/3 van OC3 = STM-1).

Figuur 150 Een SONET STS-1/OC-1 frame

De overhead-bytes hebben een specifieke functie per **sectie**, **lijn** of **pad** van een totaal SDH-transmissiesysteem.



Een **sectie** is de lengte van 1 enkele fiber, afgesloten met een **STE**. Zo'n STE is bvb. een 'repeater' die de optische signalen regenerereert.

Figuur 151 Een SDH controle systeem.

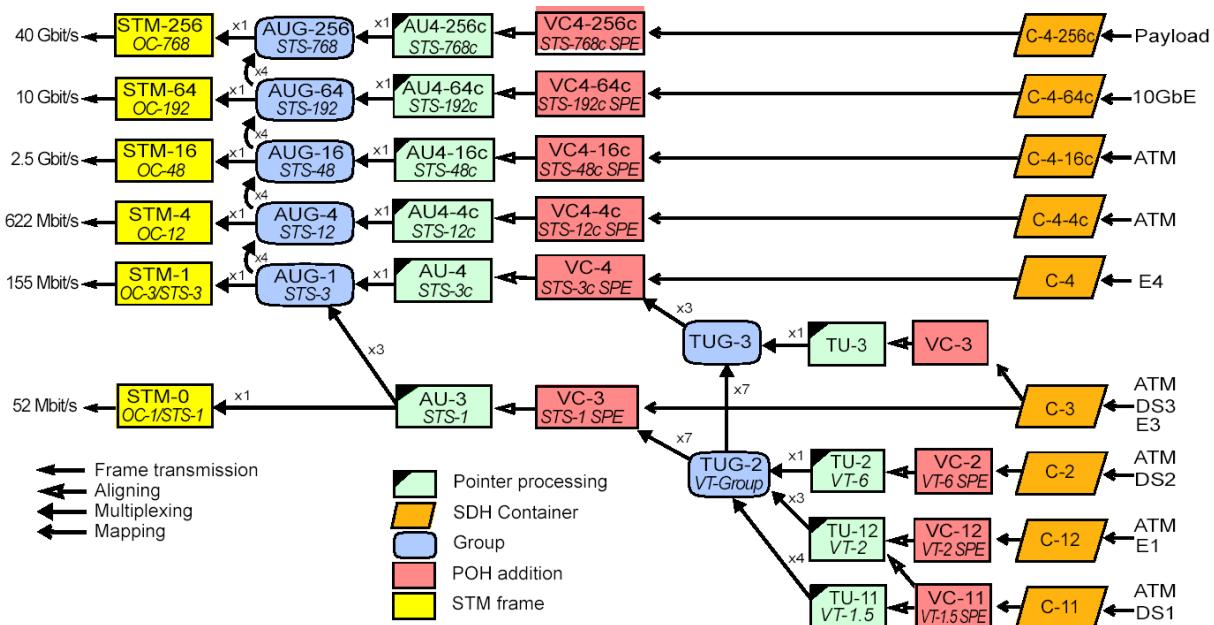
De 'lijn'-eenheden overspannen meerdere secties, afgesloten met **LTE's**. Een LTE is bvb. een **multiplexer** of **switch**. Een 'path' is een 'end to end' transmissiepad doorheen het gehele transmissiesysteem, afgesloten door een **PTE**.

De **section-overhead** bytes uit Figuur 149 hebben betrekking tot de controle van een sectie. Ze bevatten framing-bytes (A1-A2), pariteitscontrole (B1), frame-identificatie (C1), datakanalen voor netwerkmanagement (D1-D3), spraakkanalen voor het onderhoudspersoneel (E1) en kanalen voor het management van de 'customer equipment' (F1).

De **line-overhead** bytes controleren een volledige lijn : bvb. D4-D12 is een 576 Kbps datacommunicatiekanaal voor alarmen, onderhoud en administratie van een volledige lijn.

De kolommen in het 'payload' veld kunnen op verschillende manieren ingedeeld worden om lagere orde bitstromen te vervoeren. Bvb. De verschillende PDH frames, ook '*tributaries*' genoemd, worden in **containers** getransporteerd over het SDH netwerk. Elke container bevat 'path overhead', en de combinatie van de 2 wordt '**virtuele container**', **VC**, genoemd, zoals o.a. te zien is in Figuur 149.

Een STM-1 module kan nu **verschillende** VC's bevatten, zelfs containers van verschillende grootte.

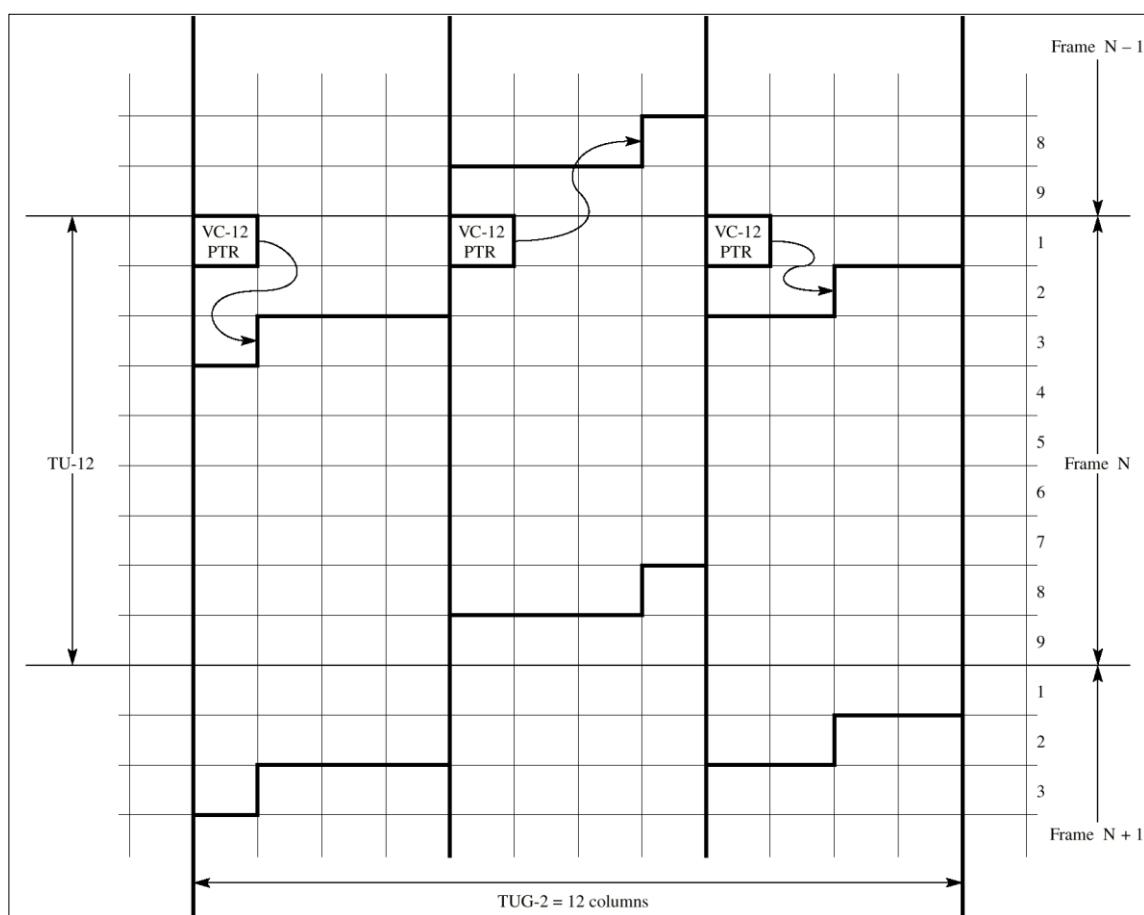


Figuur 152 SONET / SDH multiplex hierarchie.

Merk op dat de laagste orde container, **VC-1**, in 2 vormen voorkomt : enerzijds VC-11 om 1,5 Mbps **T1** frames in de VS te vervoeren, anderzijds VC-12 voor 2 Mbps **E1** frames. Elke VC wordt voorafgegaan door een pointer die aanduidt waar de VC begint, relatief t.o.v. een frame (zie verder).

De VC en zijn pointer worden '**tributary unit's, TU**', genoemd. Zo kunnen bvb. 3 TU-12 ge'Groepeerd' worden in een **TUG-2** (=TU-Groep) van 6Mbps. Deze laatste wordt op zijn beurt gegroepeerd (x7) in een **TUG-3** van 45 Mbps. Drie van dergelijke TUG-3's passen netjes in een VC-4 van 140MBps, de grootste VC die kan vervoerd worden in een STM-1 frame, en (daarom) ook wel **AU-4** of '*Administrative Unit*' genoemd. De pointer die aanduidt waar deze AU-4 begint wordt aangegeven in de 1<sup>e</sup> byte van de 'line overhead' → 'AU-4 pointer', zie ook Figuur 149 en Figuur 150.

Bekijken we zo'n TUG-2 , met 3 VC12's in. Merk op dat de TUG-2 slechts 12 kolommen beslaat van de 261 in een STM-1 frame, m.a.w. dat we 'sterk **inzoomen**'.



Figuur 153 Inplanting van een TUG-2 in een STM-1 frame.

Elk van de 3 VC12's beslaan 4 kolommen. Elke VC12 heeft een pointer staan in de eerste byte-positie die aanduidt waar de VC12 begint. Dit is dus niet altijd netjes in het eerste segment van een STM-1 aangezien er timingverschillen kunnen zijn tussen de PDH en de SDH stroom, ze zijn niet gesynchroniseerd. De VC12 met zijn pointer vormen samen een TU-12.

Een VC12 beslaat 4 rijen →  $4 \times 9 = 36$  bytes : 32 van het E1 frame, 1 van de pointer en 3 bytes die worden ge'stuffed'.

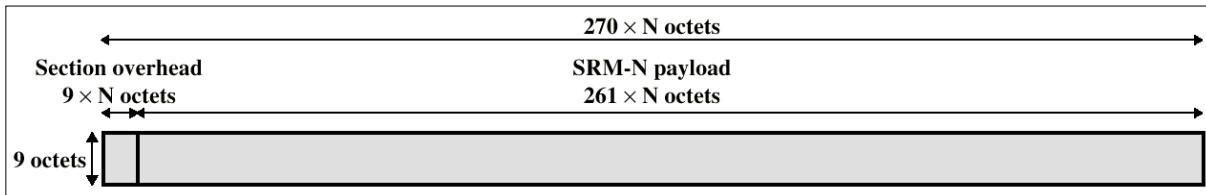
Hogere orde multiplexers bereiken, zoals te zien in Figuur 152 op p. 123 bit-rates van :

- OC3 / STM-1 = 155Mbps
- OC12 / STM-4 = 622Mbps
- OC48 / STM-16 = 2,488Gbps (2,5Gbps)
- OC192 / STM-64 = 9,953Gbps (10Gbps)
- OC768 / STM-256 = 40Gbps !

Het is bv. OC192 / STM64 waarop 10Gb ethernet in de 'W' versie kan aansluiten, zie ook pt. 1.4.4 op p.103.

De laatste, STM-256, is een uitdaging voor de ethernetwereld, mogelijk zal de volgende standard hierop aansluiten.

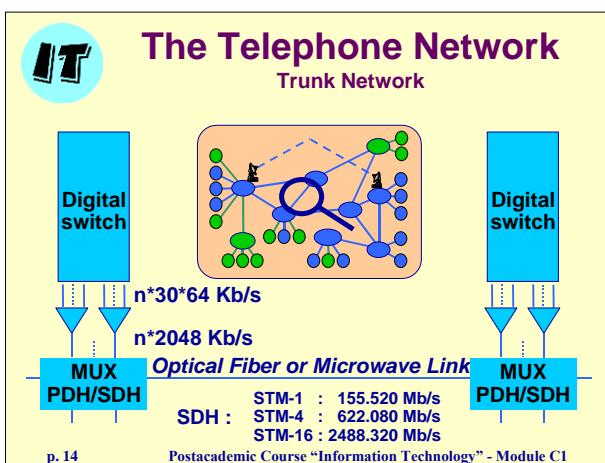
De frames zien er dan uit als :



Figuur 154 Een SDH frame formaat van STM-n.

Vergelijk ook het STS-1/OC-1 frame van SONET met dat van STM-1 van SDH.

Voor meer informatie : zoek de standaarden G707/708/709 = SDH.



Met deze achtergrond lijkt het duidelijk dat gehuurde lijnen stijgen in kostprijs naarmate de bitrate maar ook de afstand en bijgevolg het aantal MUXen, stijgt.

Figuur 155 Een SDH trunk lijn.

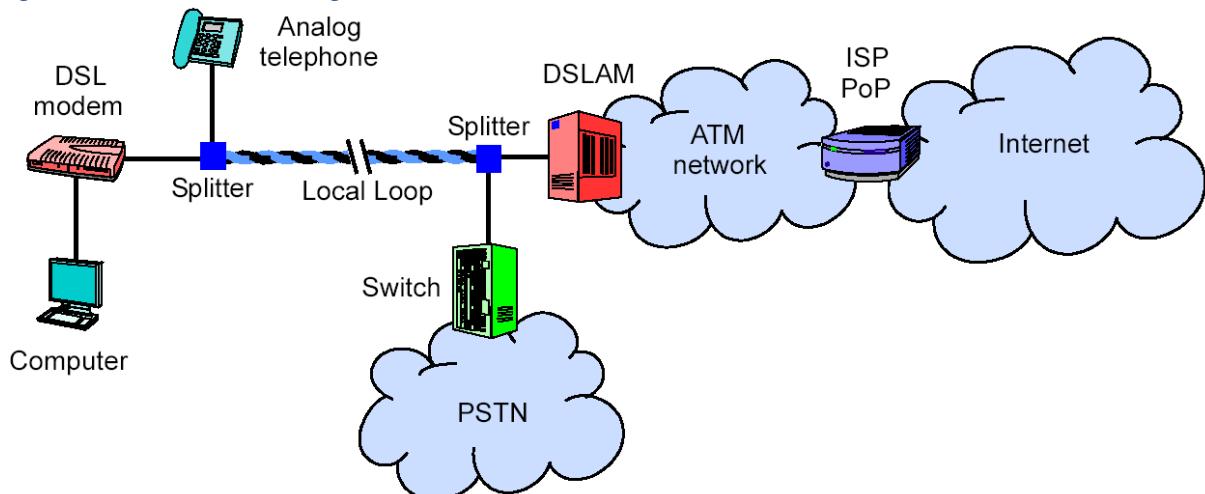
Bvb. Nemen we als vertrekpunt Brussel :

Brussel	Antwerpen	Parijs	New York
64Kbps	16 675,-	64 000,-	106 000,-
GBE	201.612	277	4.500.000

Figuur 156 Gehuurde lijnen : kostprijs.

### 1.5.3 XDSL / ADSL, (ASYMMETRIC) DIGITAL SUBSCRIBER LINE

Figuur 157 Klassieke ADSL aansluiting.

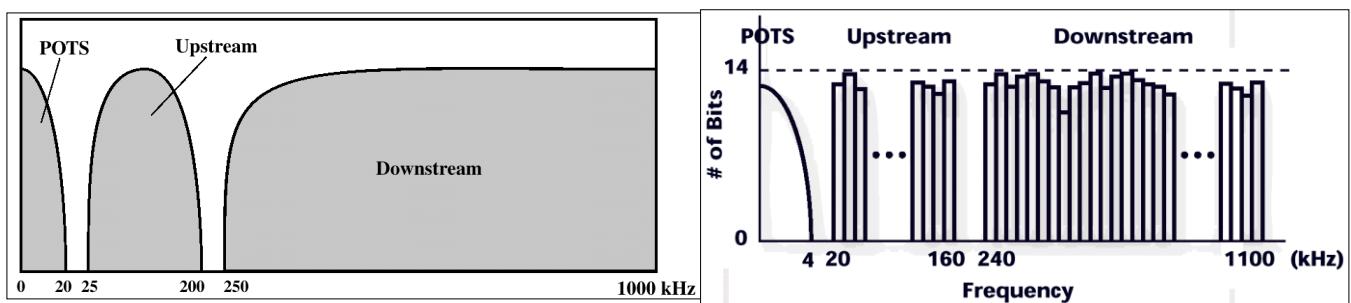


Om een hogesnelheidsnetwerk mogelijk te maken stonden de publieke operators voor de keuze : ofwel fiber doortrekken tot aan de ‘*subscribers*’, ofwel de beschikbare twisted pair bekabeling in de local loop gebruiken tot op zijn grenzen. Tot 1,1MHz of meer, gebruik makend van geavanceerde modem technologie DMT, Discrete MultiTone.

We zien in de figuur dat naast het klassieke (analoge) telefonie systeem (dat aansluit op het PSTN, Public Switched Telephone Network) op dezelfde UTP kabel het DSL systeem word gesuperponeerd.

‘Asymmetric’ duidt er op dat er meer capaciteit ‘downstream’ wordt voorzien dan ‘upstream’, met in het achterhoofd ‘video on demand’ wat voorlopig toch niet echt doorbreekt. Desalniettemin stijgt de vraag naar hoge snelheidsaccess naar het internet behoorlijk, zeker met multimedia toepassingen en de 2<sup>e</sup> generatie E-business in het vooruitzicht, maar telkens met een asymmetrische capaciteitsverdeling.

ADSL gebruikt FDM om de 1MHz bandbreedte van de twisted-pair volledig(er) te benutten.

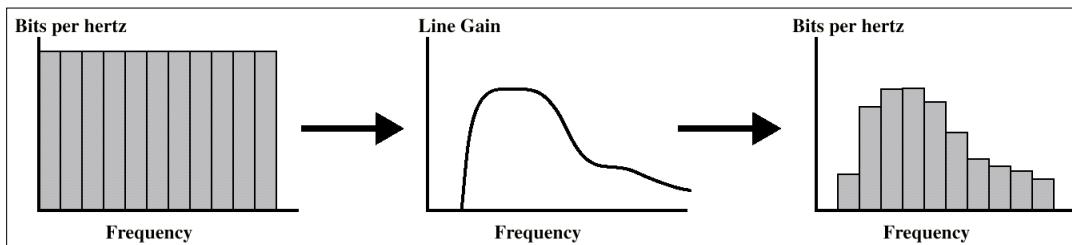


Figuur 158 ADSL frekwentiebereikindeling.

Er zijn 3 elementen in de ADSL strategie :

- **POTS**, Plain Old Telephone System, waarin de gewone, **analoge**, spraak van **0-4Khz** wordt vervoerd. De rest is voorzien als ‘guard-band’ om crosstalk met de digitale kanalen te vermijden.
- Gebruik FDM om 2 banden te voorzien : 1 smallere upstream en 1 brede downstream.
- Gebruik FDM binnenin deze 2 banden, zodat een bitstream wordt gesplitst in meerdere parallelle stromen die elk in een frekwentieband worden vervoerd.

DMT gebruikt dus meerdere carrier-signalen op verschillende frequenties. De band is ingedeeld in 4KHz subkanalen. Bij de initialisatie zendt de DMT modem testsignalen uit om de SNR (Signal to Noise Ratio) te testen op elk subkanaal. De modem kent dan meer bits toe aan de kanalen met de beste transmissie eigenschappen met een maximum van 15 bits per kanaal.

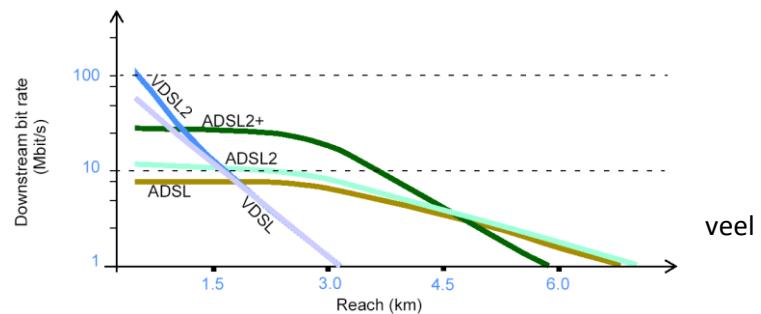


Figuur 159 DMT bits per kanaal allocatie.

Elk kanaal kan een datarate ondersteunen van 0 tot 60Kbps door QAModulatie. (15 bits → 32768 constellatiepunten). De figuur toont hoe typisch bij hogere frequenties er teveel verzwakking optreedt en er dus minder bits worden getransporteerd. De huidige modems gebruiken 256 'downstream' kanalen wat, in theorie :  $x$  60 Kbps, neerkomt op max 15,36 Mbps. In praktijk gaat dit van 1,5 tot 9 Mbps downstream en 16 tot 640 Kbps upstream.

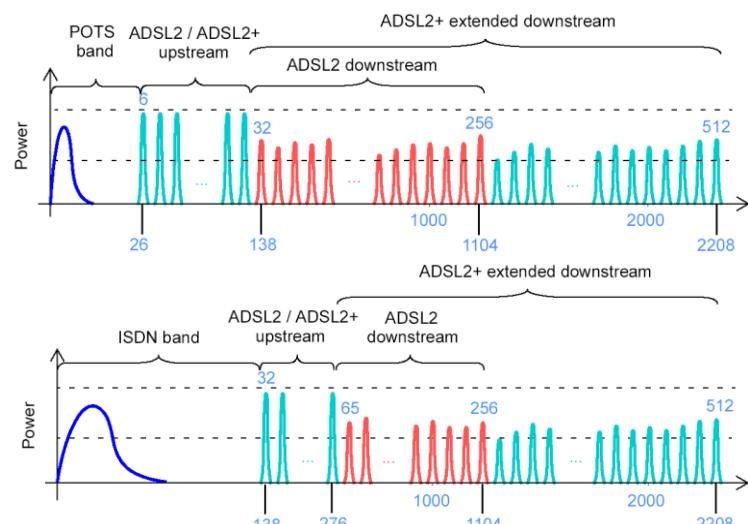
### XDSL

Bij dit schrijven zijn de nieuwste xDSL technieken in ontwikkeling, waarbij verschillende specificaties nog moeten worden uitgewerkt. VDSL heeft een vergelijkbaar schema als ADSL met een hogere datarate door te mikken op veel kleinere afstanden. We zien dat ADSL een bereik heeft tot 3km waarna de capaciteit sterk afneemt tot 5 (6?) km. VDSL heeft enkel nut tot 1,4km. Het heeft dan wel datarates van 13 tot 52 (100?)Mbps downstream en 1,5 tot 2,3 Mbps upstream.



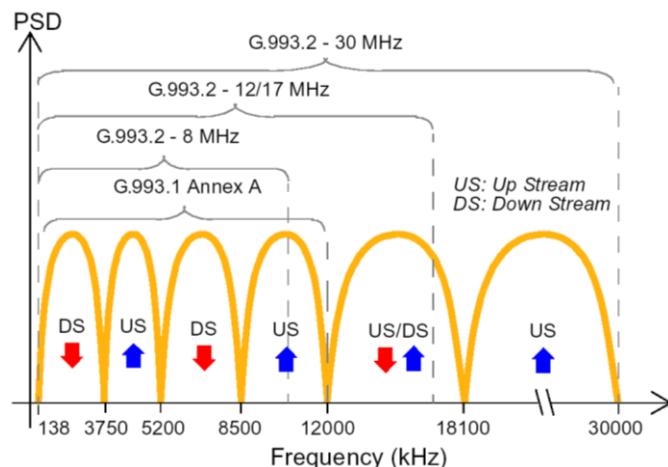
We zien ook dat ADSL2+ rekent op een veel hogere (dubbele) bandbreedte van de UTP kabel, tot 2,2MHz, wat mogelijk is tot ong 3km lengte.

In het onderste deel van de figuur zien we dat als xDSL wordt gesuperponeerd op een ISDN installatie de xDSL tones een stuk minder spectrum bezitten!

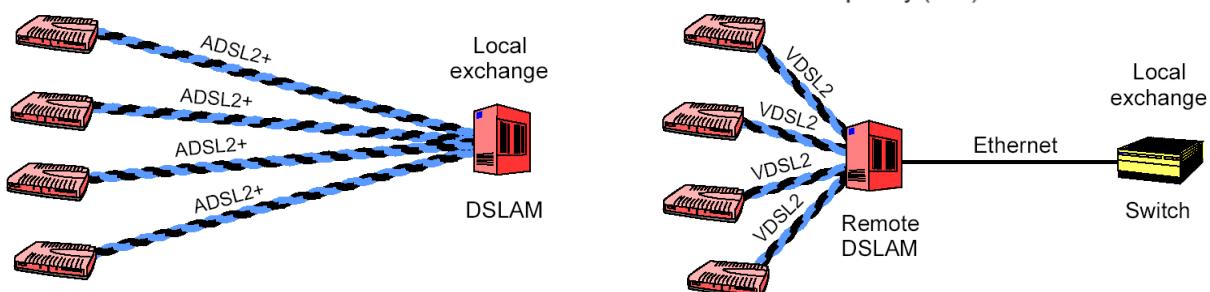


Figuur 160 ADSL2+ frekentieband.

VDSL2 zal een nog grotere frekventieband innemen, afh. van de afstand. Men realiseert dit ook door de DSLAM (DSL Access Multiplexer) steeds dichter bij de eindgebruikers te plaatsen door bv. vooruitgeschoven DSLAM's.

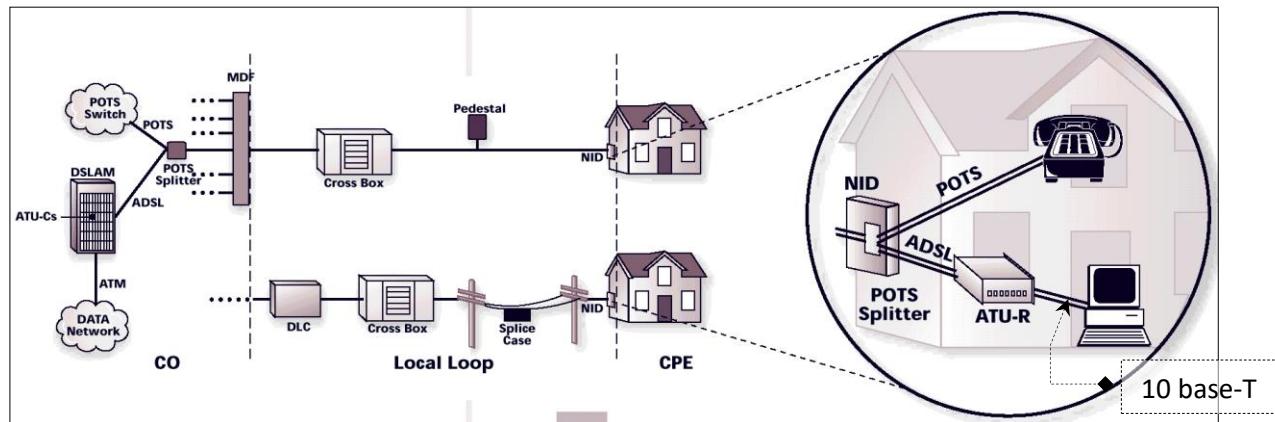


Figuur 161 VDSL2 frekentieband.



Figuur 162 ADSL2+ tov. VDSL2 afstanden.

### XDSL CONFIGURATIE



Figuur 163 Een ADSL loop architectuur.

De gehele installatie bestaat uit :

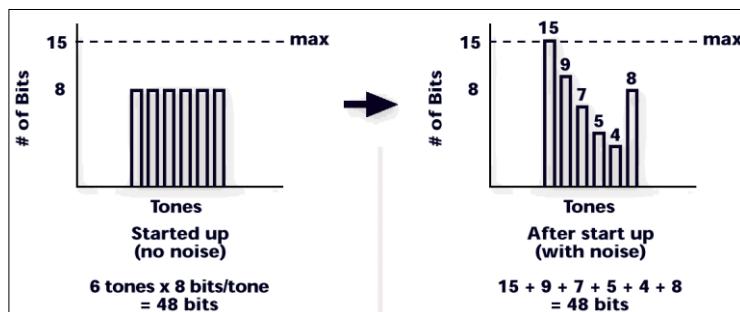
- Een ADSL 'local exchange' of Adsl Transceiver Unit Central office (**ATU-C**)
- Een ADSL modem of Adsl Transceiver Unit Remote (**ATU-R**)
- Een bandfilter om de POTS te filteren, **POTS splitter**
- Een ADSL mux die verschillende ADSL lijnen, die na de POTS splitter binnenkomen op een ATU-C, multiplexed naar een ATM fiber, een 'Digital Subscriber Line Access Multiplexer' of **DSLAM**. Hij bevat dus verschillende ATU-C's.

In de Network Interface Device, NID, zit een simpele passieve laagdoorlaat filter die de POTS altijd doorlaat, zelfs bij het falen van de ADSL lijn. De ATU-R bevat een hoogdoorlaat filter en converteert het datasignaal naar een standaard 10 base T interface (of ATM.25~).

Beiden, ATU-C en ATU-R worden gecontroleerd door een NMS, Network Management System, die de maximum datarates kan instellen, afh. van het afgesloten contract. Deze datarates kunnen evt. adaptief worden bijgesteld om problemen met de lijnkwaliteit te minimaliseren :

### BIT SWAPPING.

Dit treedt op in al de 3 bovenstaande modes. Het ADSL systeem past zich steeds aan om de lijnproblemen teniet te doen. Elk 4KHz subkanaal wordt constant ge'monitor'd en de verdeling van bits per kanaal wordt aangepast naargelang er performantieproblemen zijn op een kanaal.

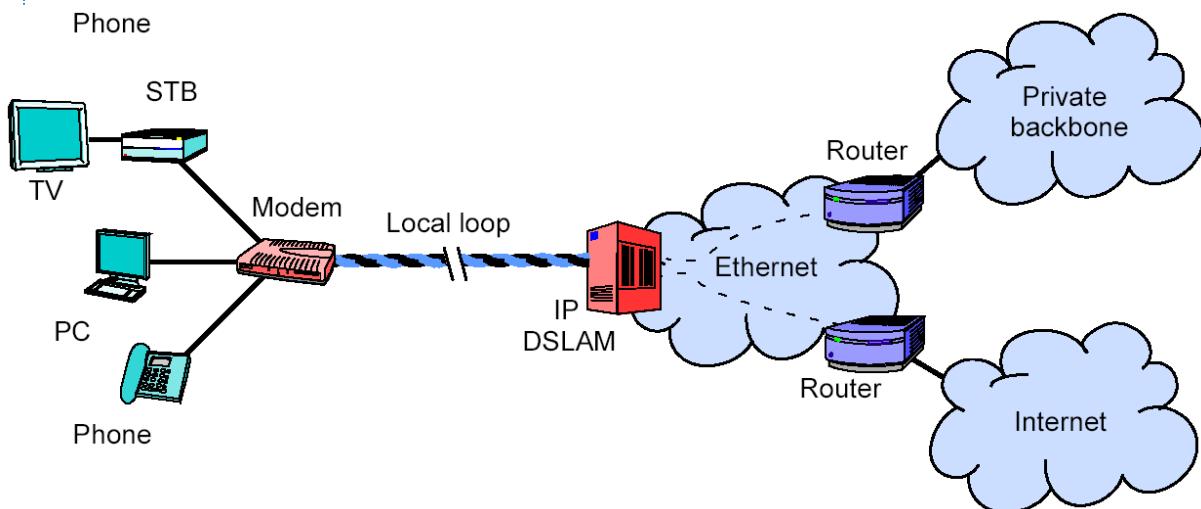


Figuur 164 Bit swapping op een ADSL lijn.

In de meeste gevallen zullen vooral op de lagere frequenties de meeste bits worden ge'alloceerd'. Zeker bij lange afstanden.

Merk op dat de ADSL connectie van de klant tot de ISP (Internet Service Provider) steeds 'op' is als zijn ADSL-modem aan staat, en er dus geen 'dial up' moet gebeuren zoals bij een analoge modem. (Ook bij een ISDN modem gebeurt dit maar dit duurt enkele ms waardoor het transparant lijkt.) Bij het opzetten van de ATU's synchroniseren ze ogenblikkelijk.

### XDSL IN EEN TRIPLE PLAY ARCHITECTUUR



Figuur 165 . xDSL in een triple play architectuur

Als we bovenstaande figuur vergelijken met Figuur 157 van p. 126 zien we dat de POTS compleet verdwenen is. ALLE verkeer, voice – video – data, ('triple' play) verloopt nu langs de DSL kanalen in digitale vorm, verpakt in ethernet/IP pakketten. Eenmaal in het netwerk worden ze aan de gepaste services gekoppeld.

Dit is de architectuur van de toekomst waar operators nu volop in investeren ...en ingenieurs aan ontwikkelen.

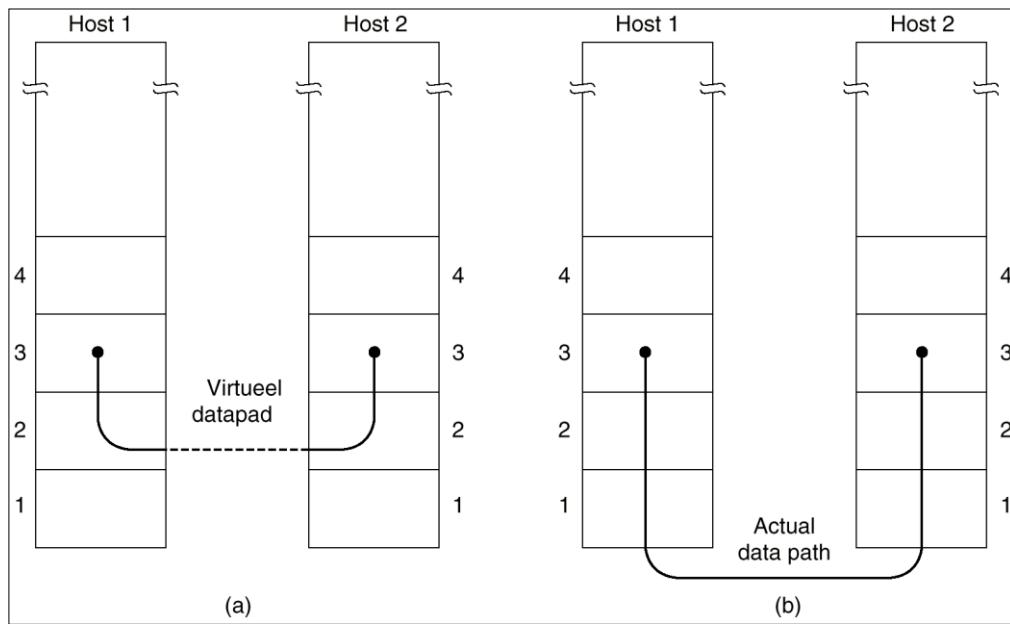
## 2 DE DATALINK-LAAG

Hetgeen tot nu toe behandeld werd situeert zich in de zogenoemde fysische laag, laag 1 van het OSI-model. Op het eerste zicht lijkt dit voldoende voor communicatie, spijtig genoeg maken deze fysische verbindingen ook **fouten**, aangegeven door hun BER, Bit Error Rate. Bovendien hebben ze een eindige datasnelheid en veroorzaken ze een vertraging.

In laag 2, de datalink laag, proberen we deze fouten te ontdekken en te verbeteren. Daarvoor heeft ze een aantal specifieke functies :

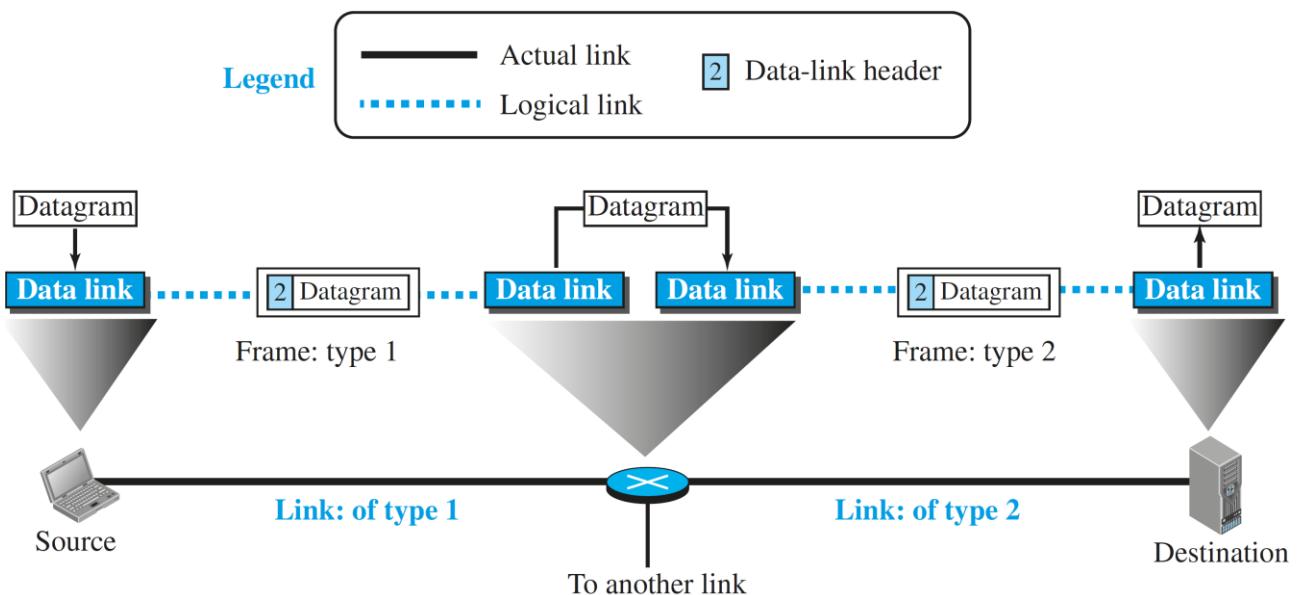
- het aanbieden van een goed gedefinieerde **interface** naar de **netwerklaag**,
- het bepalen hoeveel bits er worden samengenomen in een **frame** dat, opgestuurd over het fysieke medium, een goede kans maakt om foutloos aan te komen,
- het afhandelen van toch optredende transmissiefouten → **error control**.
- het reguleren van de framestromen zodat langzame ontvangers niet overspoeld worden door te snelle zenders → **flow control**
- het **opzetten** en afbreken van een verbinding, ...

We weten dat dit in realiteit over de fysische laag gaat, toch gaan we **virtueel** een verbinding opzetten met de datalinklaag in de ‘remote’ computer...



Figuur 166 De datalink : a) virtuele ; b) feitelijke communicatie.

... Of de verbinding loopt over verschillende netwerken, waarbij een router (L3) de frames zal uitpakken tot op laag 3. Laag 2 dient enkel om binnenin **1 netwerk** de ‘next hop’ te bereiken. Voor de volgende link (die de router kiest op basis van zijn L3 routing tabel) wordt **opnieuw** een frame samengesteld dat aangepast is voor deze link.



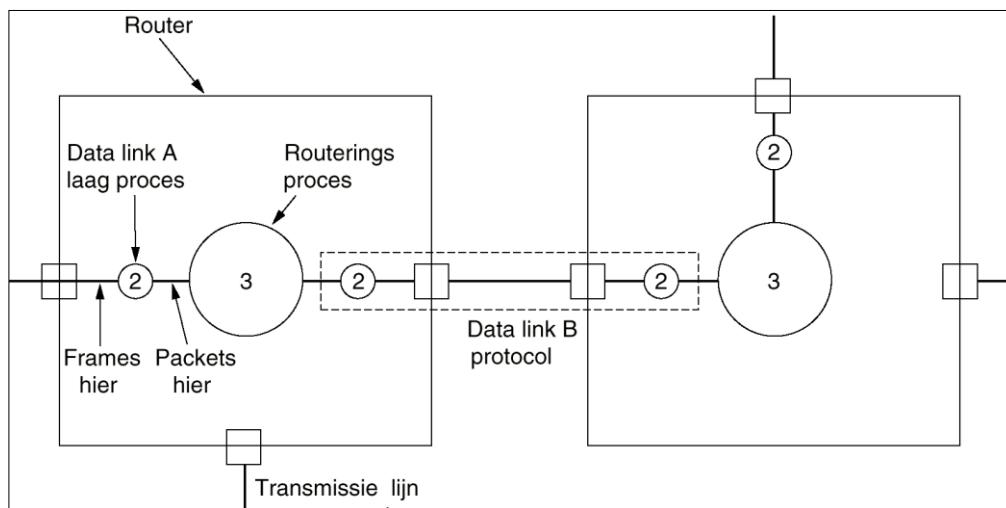
Figuur 167 Communicatie L2 – L3 over een router.

Deze L2 verbinding kan afh. v.h. soort data, maar vooral ook afh.v. het fysische medium uit verschillende diensten bestaan. We onderscheiden 3 hoofdtypen :

1. onbevestigd verbindingsloos of ‘unacknowledged - Connection Less’ (**unACK - CL**)
  2. bevestigd verbindingsloos of ‘acknowledged - Connection Less’ (**ACK - CL**)
  3. bevestigd verbindingsgericht of ‘acknowledged - Connection Oriented’ (**ACK - CO**)
1. Unack-CL zal frames oversturen zonder dat de aankomst hiervan wordt bevestigd. Dit type datalink wordt gebruikt als de BER zeer laag is zodat hogere lagen in het protocol de verloren steken moeten oprapen, bv. TCP laag 4. Het wordt ook gebruikt voor isochroon verkeer (zie verder) zoals spraak, waarbij te laat komende data erger zijn dan verkeerde data.
  2. Beter is Ack – CL. Hier wordt elk frame bevestigd. De zender is bijgevolg op de hoogte of de boodschap goed is ontvangen. Meestal is er bij de zender ook een ‘**time out**’ functie die ervoor zorgt dat, als een ACK niet binnen een vastgestelde tijd aangekomen is, het frame wordt herzonden. Deze datalinklaag is veel van toepassing op draadloze- of **onbetrouwbare** systemen.
  3. De meest uitgebreide dienst is ACK-CO. Hierbij wordt er tussen zender en ontvanger **eerst** een **verbinding** tot stand gebracht net zoals het opzetten van een telefoongesprek. Er worden in deze eerste fase variabelen en (frame)-tellers geïnitialiseerd om bij te houden welke frames ontvangen zijn en welke niet. In de **tweede** fase worden de frames zelf overgezonden, en bevestigd terug naar de zender. De **derde** fase breekt de verbinding af.

Op te merken valt dat, alhoewel we deze principes bespreken in de datalink, vele van deze technieken zoals ‘flow control’ en ‘error control’ ook in andere, hogere protocollen voorkomt, bvb. **laag 4** de transportlaag.

De datalink **situert** zich als laagste software-protocol op één (=1!) fysische verbinding. De datalink levert frames af ... ‘**tot aan het einde van de kabel !!!**’.



Figuur 168 Situering van het datalink-protocol

Stel een WAN verbinding waarbij (door de netwerklaag) een weg gezocht wordt tussen verschillende routers. Bij het binnengaan van een frame wordt het door de datalink A ‘uitgepakt’, op fouten gecontroleerd, en evt. samengevoegd met andere frame-inhouden om aan laag 3 te worden aangeboden als een **pakket**. Op laag 3 zal de uitgang gekozen worden langs dewelke het pakket moet worden verder gezonden. Afh. van het soort fysische verbinding van deze uitgang kan de datalink B op deze link, i.e. een **andere** datalink, het pakket terug opsplitsen in **andere** frame-groottes met **andere** foutcontroles enz.

We zullen ook onderscheid maken tussen 2 soorten protocollen net zoals in de fysische laag :

- **byte-** georiënteerd : protocollen gebaseerd op besturingstekens bv. uit de ASCII tabel : STX, ETX, ... . Versturen van tekens met start- en stopbits. Wordt zeldzaam.
- **bit-** georiënteerd : rijen bits worden in blokken verstuurd → bits tellen.

## 2.1 ONTWERPVEREISTEN VAN DATALINK-PROTOCOLLEN

### 2.1.1 METHODES VOOR FOUTENCONTROLE

Indelen in frames : grote data-eenheden worden opgesplitst in kleinere blokken ( $\rightarrow$ framing) om hierop foutcontrole te kunnen uitvoeren. De grootte van de blokken is afh.v. de foutenkans op de datalinklaag.

In grote lijnen zijn er 2 methodes :

1. **Forward** error control := detectie + correctie
2. **Backward** error control := detectie + ... **retransmissie**.

#### FORWARD ERROR CONTROLE

Deze techniek wordt vooral gebruikt om retransmissie van frames te beperken, of waar de mogelijkheid voor retransmissie niet bestaat / te lang zou duren. Bvb. bij compact disk spelers : de CD kan moeilijk 'terug draaien', of communicatie met satellieten : retransmissie duurt te lang.

**Hamming distance** := het minimum aantal bits verschillend tussen 2 geldige woorden.

Bvb. pariteitscontrole : 7 bit data + 1 par. (bvb. = Even)

0000000	0	}	<u>Hamming distance = 2 !</u>
0000001	1		
0000010	1		
0000011	0		

$\rightarrow$  hierbij wordt 1 bit fout gedetecteerd, 2 bitfouten in 1 woord echter NIET! (M.a.w. er moeten 2 bitfouten optreden om van 1 codewoord naar het andere over te gaan.)

Stelling : voor de **detectie** van  $n$  bitfouten  $\rightarrow$  moet de Hamming distance,  $Hd = n+1$   
**correctie** van  $n$  bitfouten  $\rightarrow$   $Hd = 2n+1$

Voorbeeld van error correctie : Hamming 1bit code, een block code. D.w.z. de volledige boodschap wordt behandeld als een 'block' van  $k$  'source'-bits. Hieraan worden een aantal error-correctie bits toegevoegd wat de te verzenden boodschap brengt op  $n$  bits,  $n>k$ .  $\rightarrow$   $(n,k)$  block code.

( $k/n$  = code efficientie,  $1-k/n$  = redundantie).

Bvb. stel een ASCII karakter van 7 databits  $\rightarrow k=7$ .

Alle 1 bit fouten moeten gedetecteerd en gecorrigeerd worden. Deze block vereist 4 controlebits, allen op de bitposities van de machten van 2 : 1,2,4 en 8.  $\rightarrow$  block code (11,7)

Stel het **origineel** woord is 1 0 0 1 1 0 1 (= ‘source bits’), er worden 4 correctiebits =X toegevoegd.

Bitplaats	11	10	9	8	7	6	5	4	3	2	1
Verzonden code	1	0	0	X	1	1	0	X	1	X	X

De 4 controlebits X worden berekend door de (source-)bitpositions = 1<sup>L</sup> op te tellen modulo 2 :

$$11 = 1011$$

$$7 = 0111$$

$$6 = 0110$$

$$3 = 0011$$

$$\underline{X = 1001}$$

Wat het **verzonden** code woord brengt op :

Bitplaats	11	10	9	8	7	6	5	4	3	2	1
Verzonden code	1	0	0	1	1	1	0	0	1	0	1

De ontvanger op zijn beurt telt ook alle bitpositions = 1<sup>L</sup>, nu van de **verzonden** code.

$$11 = 1011$$

$$8 = 1000$$

$$7 = 0111$$

$$6 = 0110$$

$$3 = 0011$$

$$1 = 0001$$

$\underline{0000} \rightarrow$  geen fout.

Stel er is een bitfout op plaats 11  $\rightarrow =0^L \dots \rightarrow$

$$8 = 1000$$

$$7 = 0111$$

$$6 = 0110$$

$$3 = 0011$$

$$1 = 0001$$

$\underline{1011} \rightarrow$  fout op bit 11  $\rightarrow$  inverteer

Als er nu 2 fouten zouden optreden hebben we hiervan wel een detectie, geen correctie.

Nog geavanceerder zijn convolutiecodes zoals trellis/viterbi of reed solomon.

## BACKWARD ERROR CONTROLE

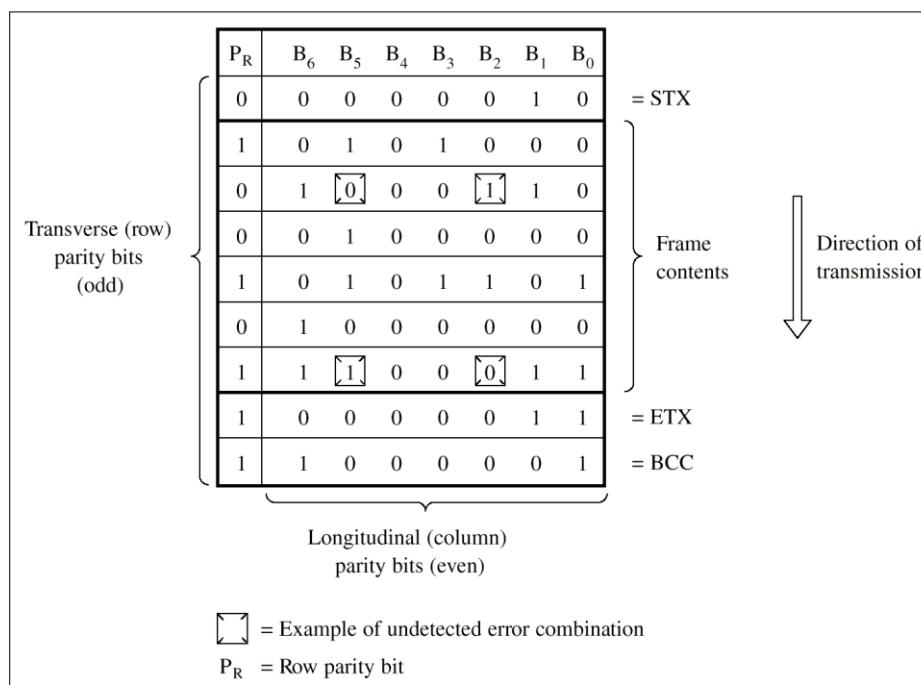
Dit **detecteert** enkele fouten en vraagt om **retransmissie** → ‘ARQ = Automatic Repeat reQuest’.

Het doel is nu om frames te kiezen die een grote kans maken om foutloos verzonden te worden. Bvb. de BER van glasvezel =  $10^{-9}$ , van een telefoonlijn : =  $10^{-5}$ . Als we nu in dit laatste geval een framegrootte zouden kiezen van  $10^5$  worden praktisch alle frames foutief →  $\forall$  herzenden !...?

**Pariteitscontrole** : dwarspariteit of ‘row parity’ en langspariteit of ‘column parity’

Zoals hierboven reeds besproken heeft een gewone dwarspariteit, berekend op 1 karakter, een hamming distance van 2.

Wanneer nu een **blok** karakters wordt doorgestuurd wordt hierop nog een langspariteitscontrole uitgevoerd → er wordt een ‘Block sum Check Character’ (BCC) toegevoegd dat bestaat uit een pariteitscontrole van de bits **per kolom**. (hier bvb. even, terwijl de dwarspariteit oneven is). De BCC is de modulo 2 som van de karakters uit het blok.



Het is duidelijk dat bij het optreden van **1** bitfout de **exacte plaats** hiervan kan aangeduid worden.

**2** bitfouten op 1 karakter worden niet herkend door de rij-pariteit, wel door de kolompariteit. Enkel een herhaling van dezelfde fouten op een volgend karakter glipt door de controle.

**Figuur 169 Rij – en kolom-pariteit.**

Een alternatief schema bestaat erin om als BCC het 1's complement op te sturen van de som (=gewone, niet modulo) van alle karakters. De ontvanger herhaalt deze procedure, inclusief het BCC, en moet dan  $\forall 1^L$  uitkomen.

### Cyclic redundancy check.

De vorige controlesschema's worden veel gebruikt voor het optreden van **enkelvoudige** bitfouten. Voor '**burstfouten**' te ontdekken moet men een beroep doen op 'polynoom'-codes. Voor een volledig frame wordt dan een 'check-sequentie' berekend en toegevoegd achteraan het frame. De ontvanger doet dezelfde berekening op het frame inclusief de **FCS** (Frame Check Sequence) of **CRC** (Cyclic Redundancy Check) en moet een gekend resultaat uitkomen, bvb.  $\forall 0^L$ , zoniet is er een fout en moet het frame herzonden worden.

Bvb. stel een bericht (Message)  
**M(x) = 11100110** moet opgestuurd worden, en de overeengekomen Genera-tor-polynoom **G(x) = 11001**, ook voorgesteld als :

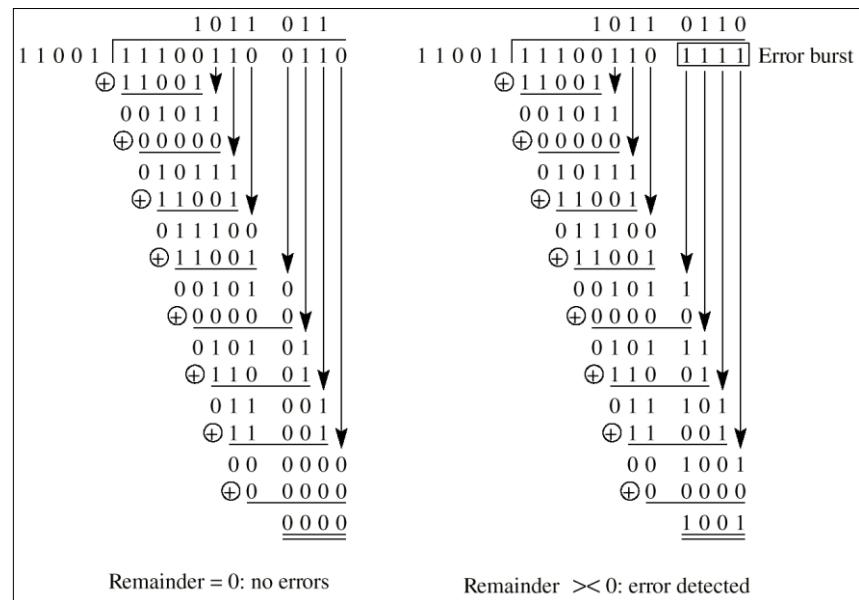
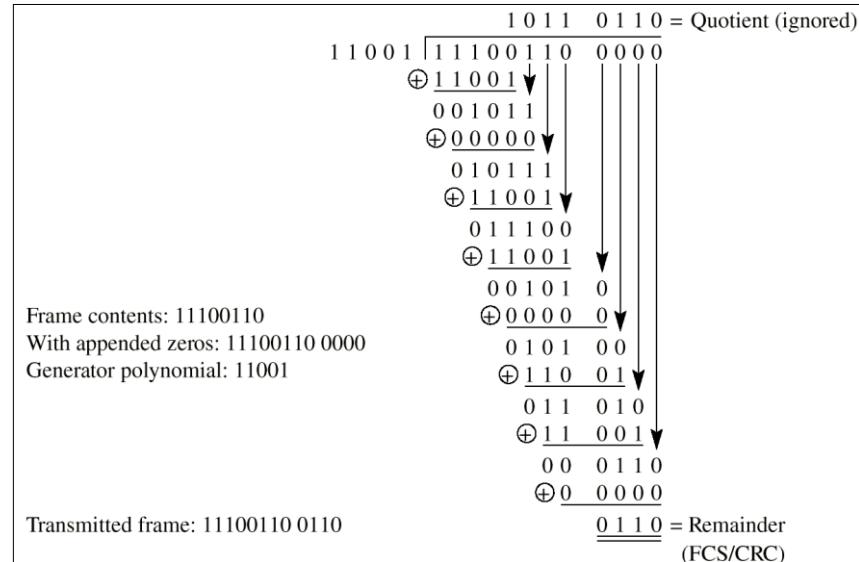
$$G(x) = X^4 + X^3 + 1.$$

De CRC zal bijgevolg **4** bits lang zijn, en daarom wordt M(X) eerst vermenigvuldigd met  $2^4$  door er **4** nullen aan toe te voegen.

Het geheel wordt dan **modulo-2**

gedeeld door bit per bit een EXOR functie uit te voeren. De rest is de FCS of CRC.

Bij de ontvanger wordt het volledige frame, **inclusief CRC**, gedeeld door dezelfde polynoom **G(x)**, (op voorhand afgesproken), waarbij in dit geval een rest =  $\forall 0^L$  een foutvrij frame aanduidt.



Figuur 170 Een CRC berekening.

Praktisch :  $\text{CRC - 16} = X^{16} + X^{15} + X^2 + 1$       ( $\exists G(x) = 1\ 1000\ 0000\ 0000\ 0101$ )

$\text{CRC - CCITT} = X^{16} + X^{12} + X^5 + 1$

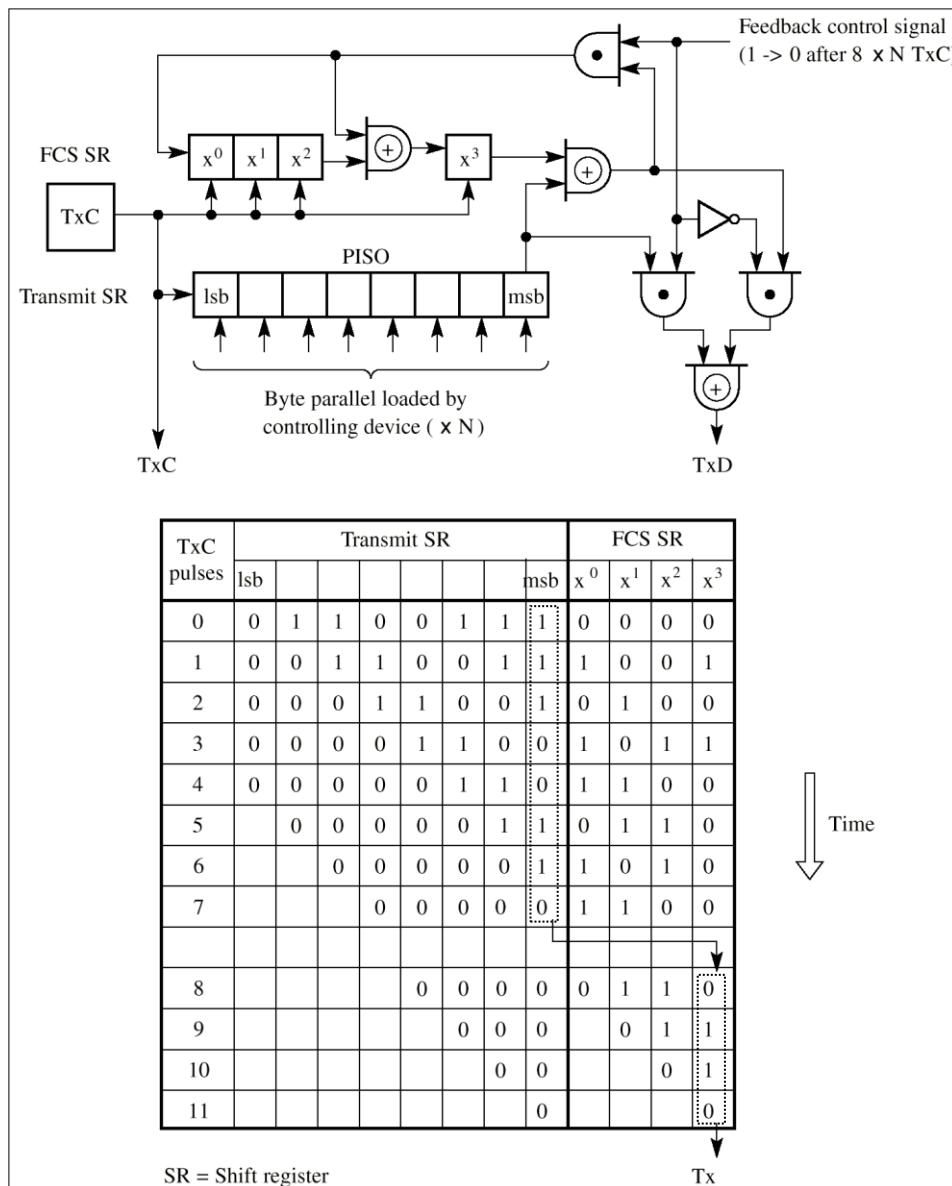
$\text{CRC - 32} = X^{32} + X^{26} + X^{23} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$

CRC –16 en CRC – CCITT worden veelvuldig gebruikt in WAN's, CRC – 32 in LAN's, bvb. ethernet.

Alhoewel de berekeningswijze laat vermoeden dat het opwekken van de CRC behoorlijk ingewikkeld is, kan het nochtans zeer eenvoudig gebeuren zowel in hardware als in software.

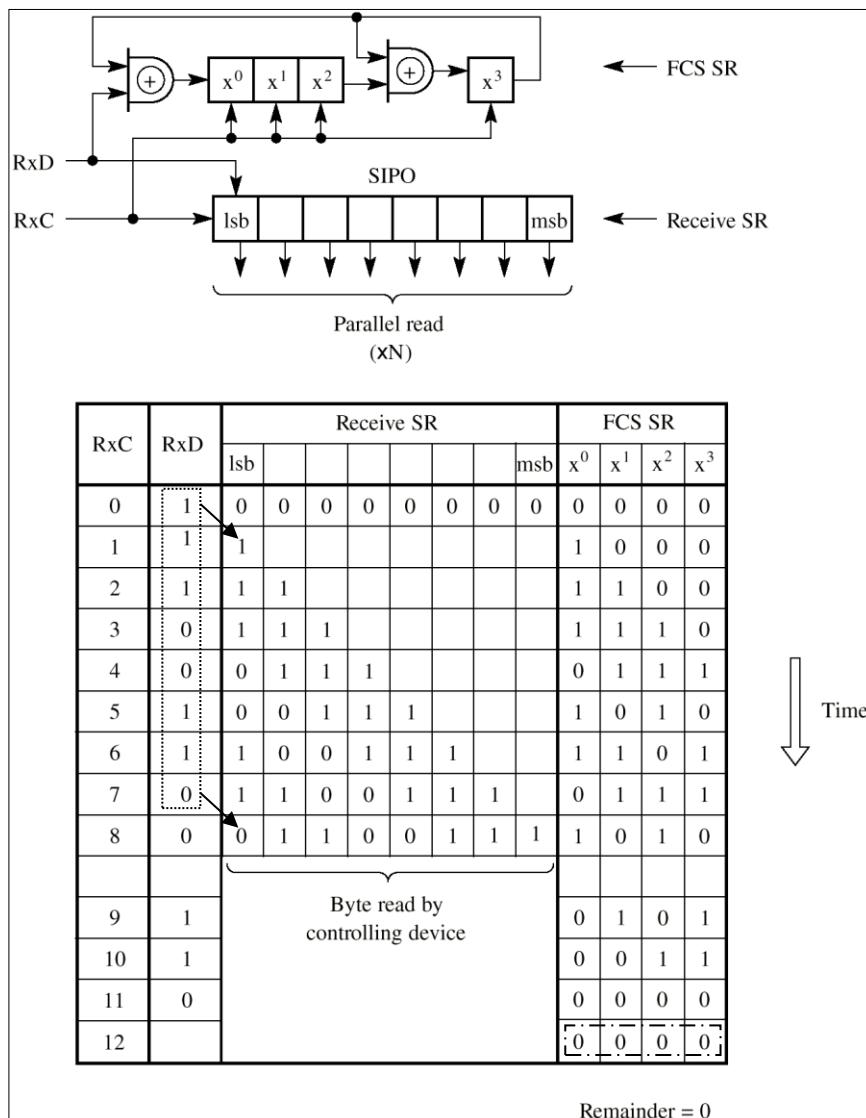
Bvb. de generatie van een 4-bit CRC met  $G(x) = X^4 + X^3 + 1$ , zie ook Figuur 170.

We hebben hiervoor een **4-bit** schuifregister nodig, waarbij de cellen  $X^3$  en  $X^0$ , (de bits = **1<sup>L</sup>** uit **G(x)** behalve de eerste) gecreëerd worden door een XOR functie met het terugkoppelpad en de vorige cel.



Figuur 171 Een hardware CRC generator.

Elke 8-bit byte wordt parallel geladen in het PISO schuifregister en schuift, msbit eerst, uit tot enerzijds TxD en anderzijds, XORed met  $X^3$  naar het feedback signaal. Hiermee wordt de FCS/CRC gecreëerd. Nadat alle data (= N byte) is uitgeschoven via TxD wordt het controlesignaal 0<sup>L</sup> waardoor de CRC wordt overgezonden op TxD. Als er bvb. slechts 1 byte (**11100110**) zou worden overgezonden is TxD = **11100110 0110**, zoals berekend in Figuur 170.



De **ontvanger** hardware is gelijkaardig aan de zender: de ontvangen data schuift in het SIPO register dat byte per byte wordt gelezen.

Maar tegelijk wordt uit RxD de FCS berekend.

Als alle data zijn toegekomen wordt de FCS gecontroleerd of hij  $\forall 0^L$  bevat, zoniet is er een fout opgetreden.

Figuur 172 Een hardware CRC ontvanger.

Onderaan vindt u ook een software programma om CRC's te berekenen.

{ Assume a preformatted frame to be transmitted (including a zero byte at its tail) or a received frame is stored in a byte array buff[1.. count]. Also that the 8 active bits of a 9-bit divisor are stored in the most-significant 8 bits of a 16-bit integer CRCDIV. The following function will compute and return the 8-bit CRC }

```

function CRC : byte;
var
  i, j : integer;
  data : integer

begin  data := buff[1] shr 8;
       for j := 2 to count do
         begin
           data := data + buff[j];
           for i := 1 to 8 do
             if ((data and $8000) = $8000) then
               begin data := data shr 1;
                 data := data xor CRCDIV; end
             else data := data shr 1;
           end;
         end;
       CRC := data shr 8;
end;
```

### 2.1.2 HERSTEL VAN TRANSMISSIEFOUTEN 'ERROR CONTROL'

Door toevoeging van redundantie kunnen nu fouten **gedetecteerd** (→ retransmissie) of zelfs gecorrigeerd worden. De eerste methode is de meest gebruikte, zeker voor full duplex verbindingen. Er kan gewerkt worden aan het ACKnowledge principe.

Er bestaan afspraken over welke CRC of pariteit wordt toegepast en wat de **procedure** is voor hertransmissie → **ARQ** : 'Automatic Repeat reQuest'. Over het algemeen zendt de ontvanger een speciaal **besturingsframe** terug met een positieve- (ACK) of negatieve- (NAK) bevestiging.

Een verdere complicatie is dat frames (en dus ook ACK/NAK) volledig kunnen verdwijnen. Daarom zal de datalink voor elk verzonden frame **timers** initialiseren waarbinnen een ACK moet ontvangen worden. (minstens groter dan 2 x de looptijd van de verbinding...) De normale procedure bij het aflopen van die timer is dan dat de frames herzonden moeten worden, waardoor de mogelijkheid ontstaat dat hetzelfde frame **2-maal** wordt ontvangen. Om dit te voorkomen worden aan de uitgaande frames **volgnummers** toegekend.

We kunnen nu 2 types ARQ-**procedures** onderscheiden :

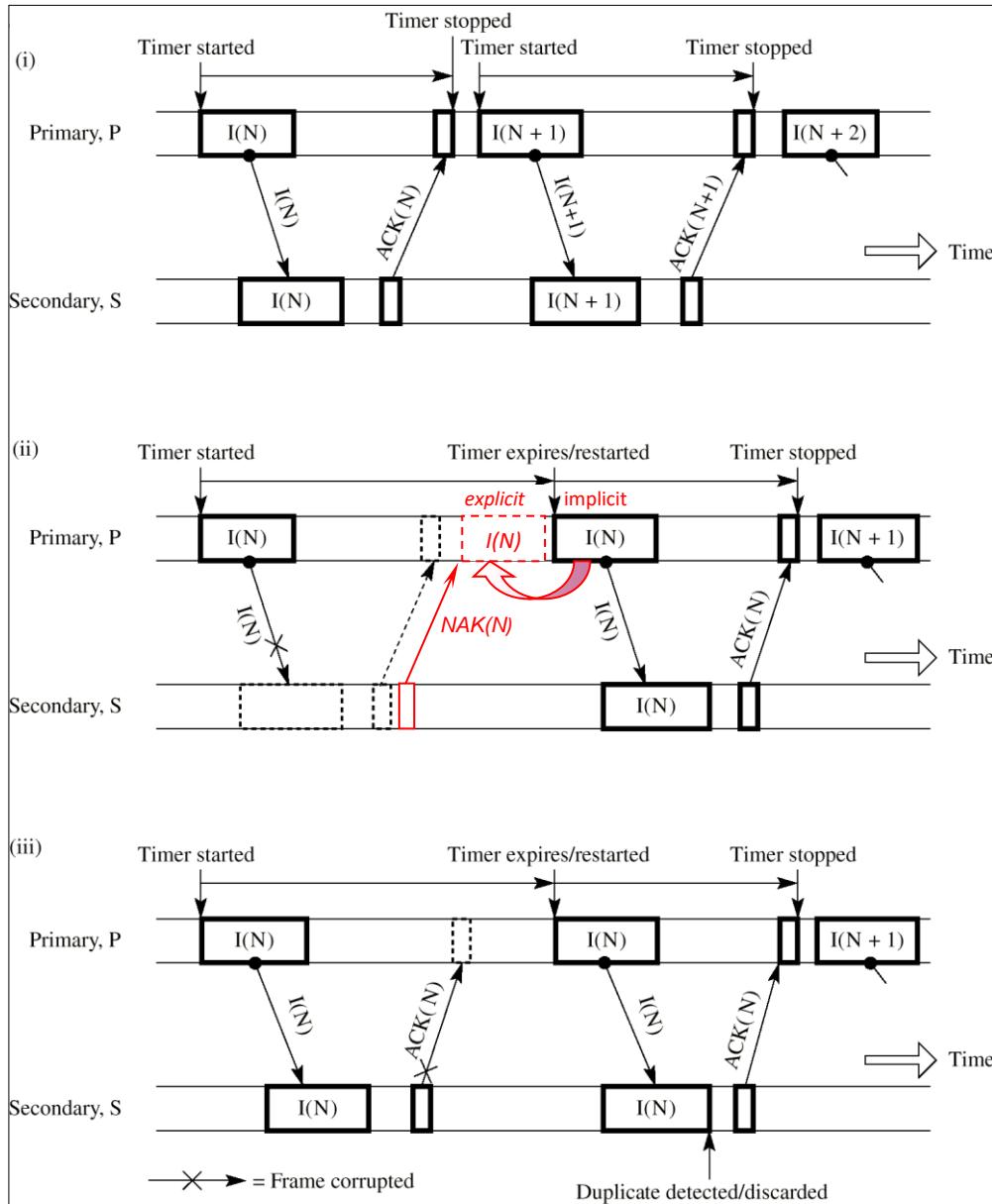
- **idle RQ** (implicit en explicit), meestal gebruikt bij byte-oriented schema's
- **continuous RQ** voor bit-oriented schema's, dat zelf 2 strategieën kent :
  - Go back N
  - Selective repeat (implicit en explicit)

Zoals we later meer specifiek gaan bespreken bij HDLC bestaan er over de link Informatie frames (I) en controle frames bvb. ACK tussen zender (= Primary, P) en ontvanger (= Secondary, S).

### STOP AND WAIT ARQ OF IDLE RQ

Eigenlijk is dit principe bovendien in te delen in : 'implicit retransmission' → elk correct ontvangen bericht wordt bevestigd, het **ontbreken van ACK** (binnen de timer tijd) betekent een **corrupt** frame.

In normale werking wordt elk frame bevestigd binnen de timer-tijd.



Wordt er een frame verstoord of verworpen dan bevestigt de ontvanger dit **niet** (implicit!). De timer verloopt en het frame wordt herzonden.

Ook ACK frames kunnen worden verstoord waar-door het frame wordt herzonden en dus 2 maal ontvangen.

Duplikaten moeten op basis van hun frame-nummer ( $N$ ,  $N+1, \dots$ ) worden herkend en verwijderd.

Figuur 173 Idle RQ, implicit (+ explicit) retransmission

Al een eerste **verbetering** is nu dat corrupte I-frames door de ontvanger

worden herkend, waarbij deze een **NAK** bericht terug stuurt, wat neerkomt op een vraag voor **retransmissie**. Er moet nu niet meer gewacht worden op het aflopen van de timer → explicit request. (Veruit meest gebruikt.)

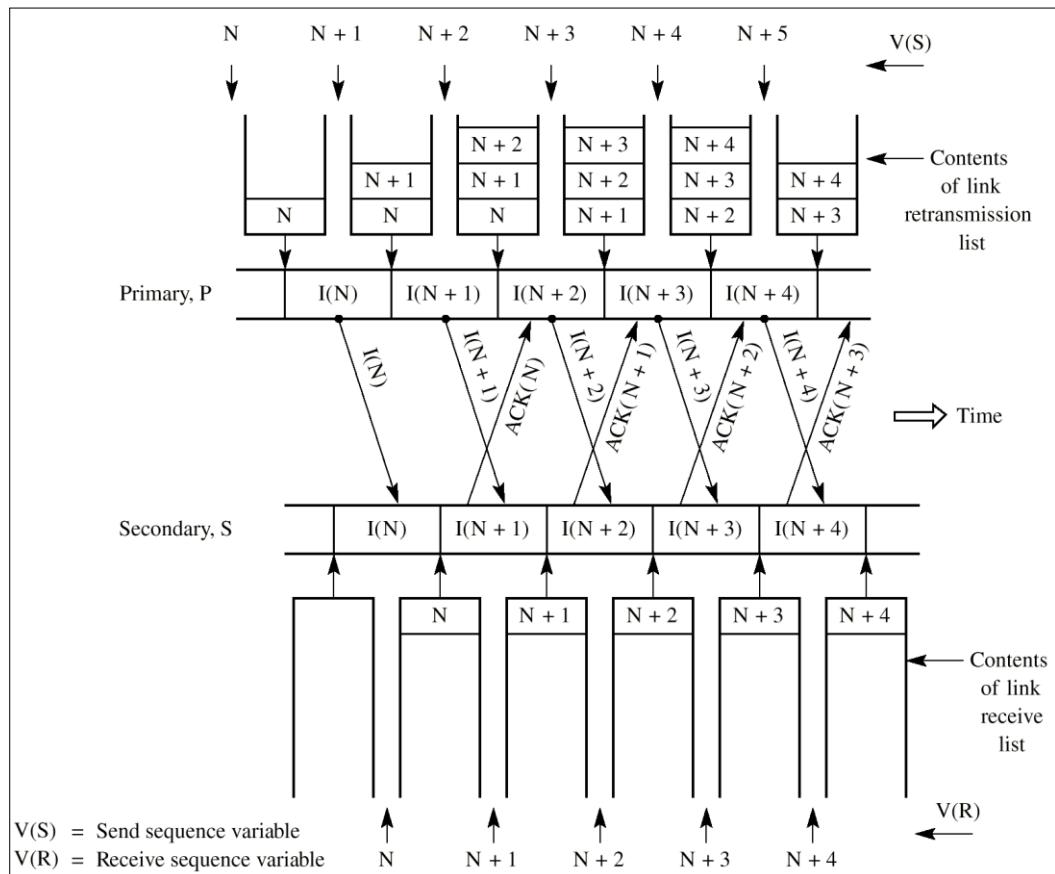
De timer-werking moet uiteraard wel worden behouden.

- Conclusie :**
- de zender (=P, primary) heeft max 1 I-frame uitstaan → er is geen probleem om de **volgorde** van de ontvangen frames te bepalen door de ontvanger (=S, secondary).
  - om '**duplicates**' te ontdekken moet er een '**sequence**'-nummer (**N**) in het pakket zitten.
  - 'link utilization' : de verbinding werkt als een **half duplex** verbinding.

### CONTINUOUS REQUEST .

Aangezien 'Idle RQ' een full duplex verbinding a.h.w. slechts gebruikt in half duplex mode, worden nieuwe schema's voorzien om de 'link utilization' te verhogen. Dit ten koste van, zo zal blijken, geheugenbuffers.

Ook hier wordt elk I-frame bevestigd door een ACK frame, **maar de zender wacht er niet op** om het volgende I-frame te versturen. Foutloze communicatie verloopt als volgt :



Figuur 174 Continuous request foutvrije communicatie.

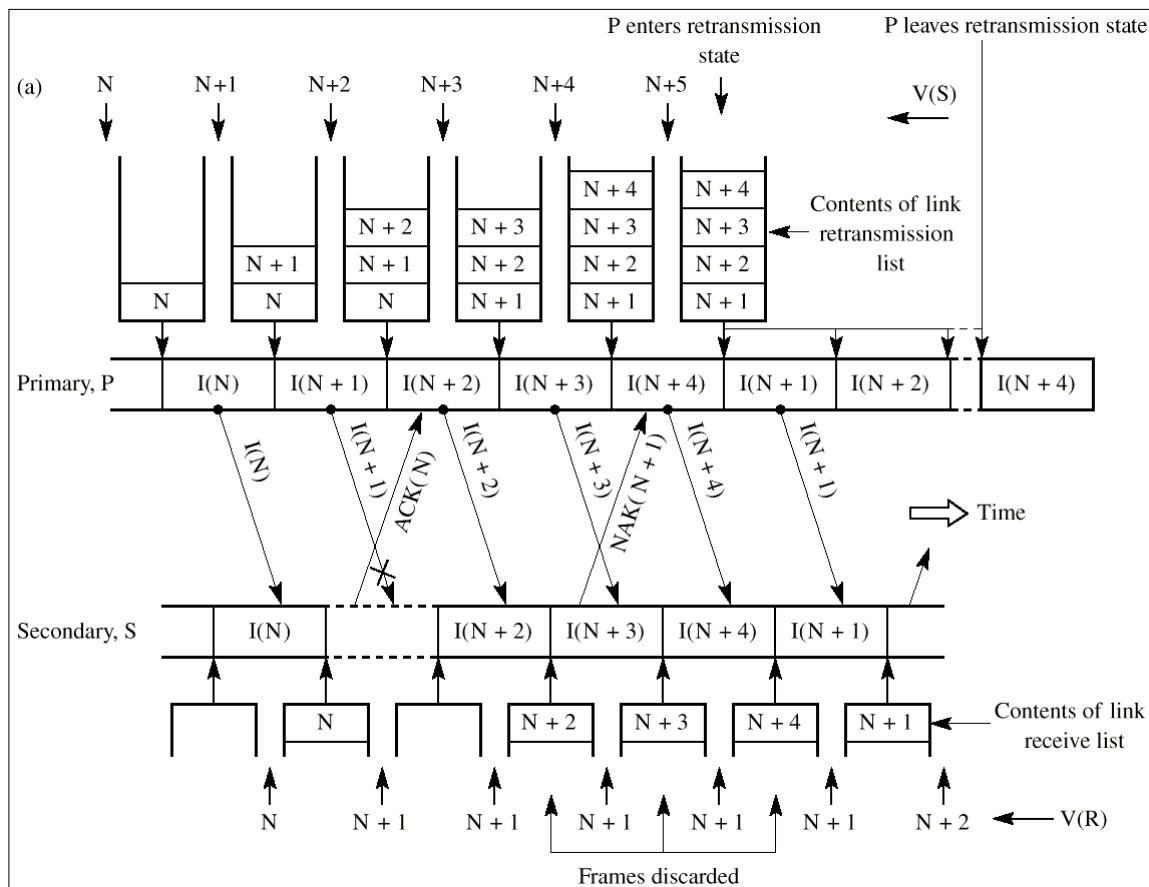
Elk pakket krijgt een nummer ( $N+...$ ) en wordt in de retransmissie lijst bijgehouden tot de resp.  $ACK(N+...)$  wordt ontvangen. Er moeten dus FIFO buffers bijgehouden worden in P en S.

Bovendien houdt P een teller bij,  $V(S)$ , die het **volgende te verzenden frame** aanduidt, m.a.w. het aantal verzonden pakketten is  $V(S)-1$ . S houdt een teller  $V(R)$  bij die het **volgende te verwachten frame** aanduidt, m.a.w. het aantal ontvangen pakketten is  $V(R)-1$

Als er **fouten** optreden zijn er 2 retransmissie-strategieën : **Go back N** en **selective repeat**.

### GO BACK N CONTINUOUS REQUEST :

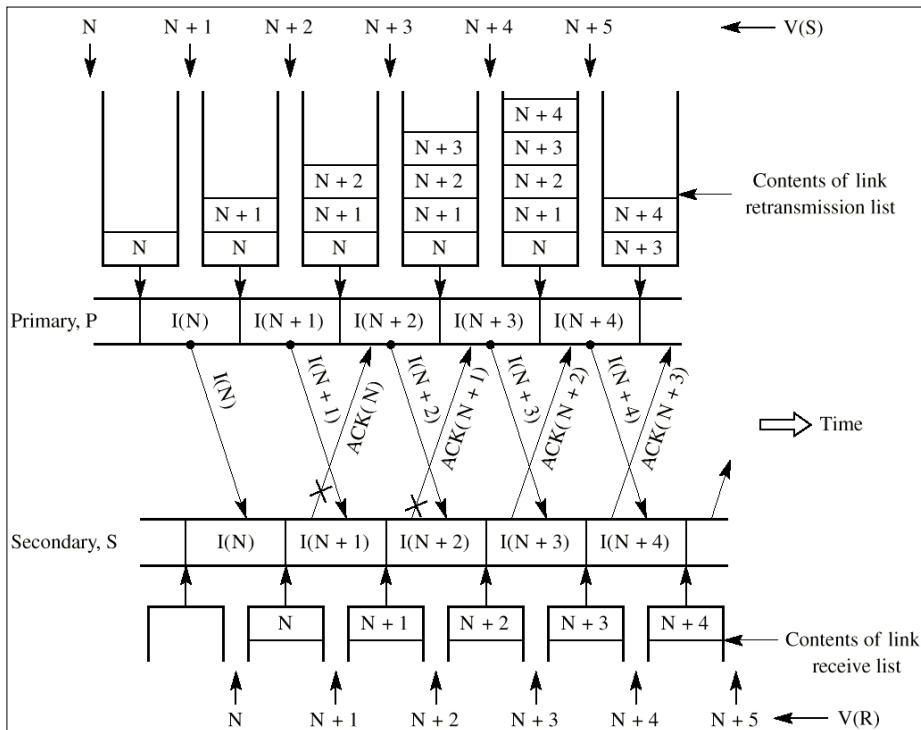
Als S een pakket 'out of sequence' ontvangt brengt hij P hiervan op de hoogte door een NAK frame met het nr. van het niet ontvangen frame (vb. in Figuur 175 frame N+1). De retransmissie start terug **volledig** vanaf dit nummer. S verwerpt alle frames totdat hij het gewenste ontvangt → **Go back N**



Figuur 175 Go back N : een fout I frame

Het voordeel van dit schema (t.o.v. het volgende) is dat S geen problemen kent om de frames **in volgorde** te plaatsen, hij heeft zelf **geen FIFO buffer** nodig. Het nadeel is dat een aantal **correcte** frames moeten worden **herzonden** → 'link utilization' ↘.

Het is bovendien mogelijk dat ACK- frames verloren gaan. In Figuur 176 : als P een **ACK(N+2)** ontvangt gaat hij ervan uit dat ACK(N) en ACK(N+1) fout waren, maar dat, bij gebrek aan NAK(N of N+1), de resp. I-frames wél zijn aangekomen. P gaat gewoon verder met I(N+3).



Figuur 176 Go back N : een fout ACK frame

Het is zelfs mogelijk dat **NAK** frames verloren gaan. S zal na een time out, waarbij het voor retransmissie gevraagde I-frame niet is binnen gekomen, NAK's blijven sturen → ACK betekent dus : geen (onderliggende) NAK's. ACK is ook cumulatief : hier 'tot en met N+2'.

(nota : dit is een 'expliciete Go back N' werking. Er bestaat ook een 'impliciete ~' die we zullen zien in pt. Data transport in NRM mode. op p. 164)

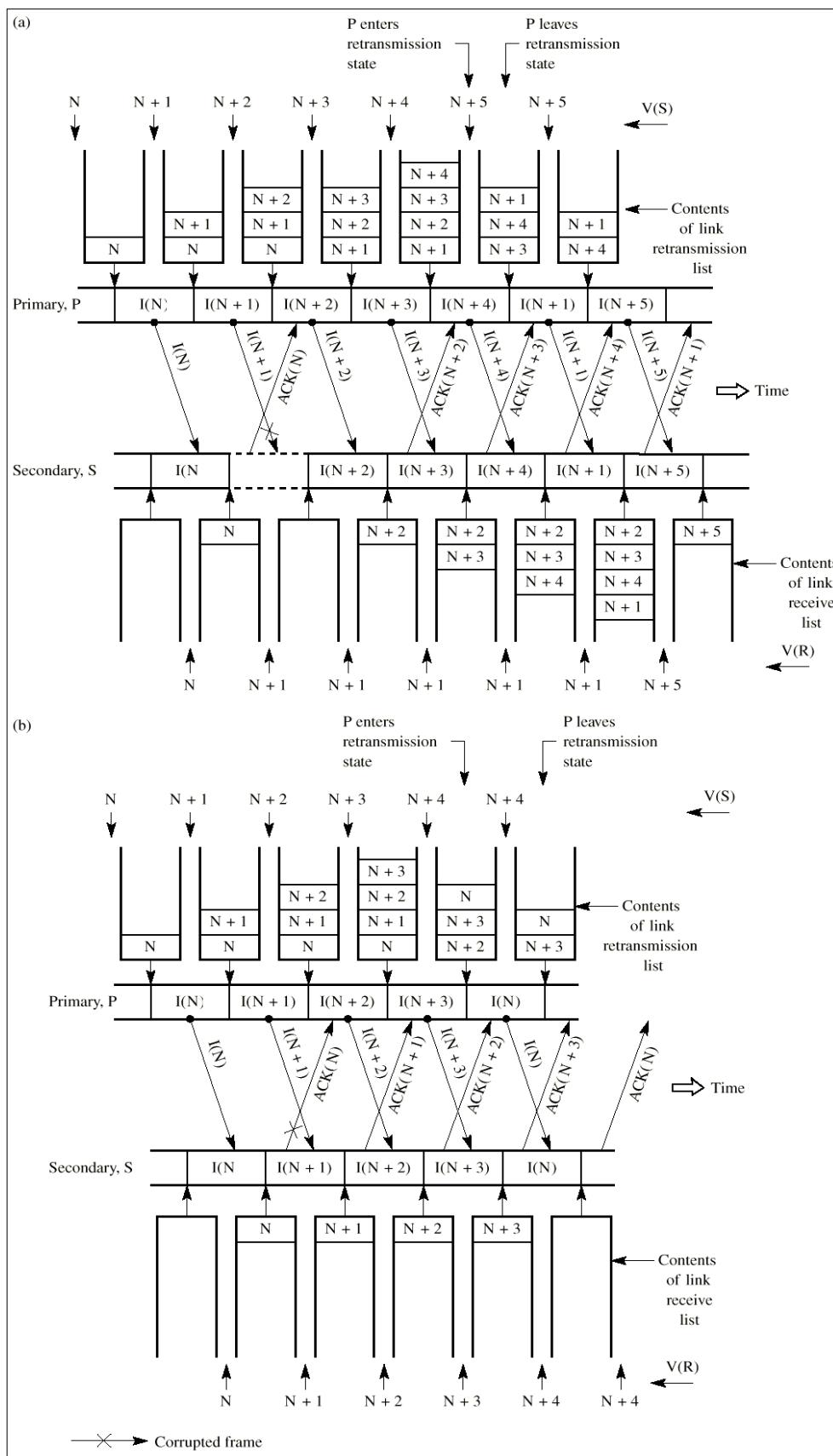
#### SELECTIVE REPEAT CONTINUOUS REQUEST

In dit schema zijn er 2 mogelijkheden, net zoals bij idleRQ :

- Implicit retransmission :** zie Figuur 177a → S bevestigt **elk** I frame, en **P leidt** uit de ontvangen ACK-frames **af** welke I-frames verloren zijn en herzondt ze. (=Go back N)  
Bij ontvangst door P van een ACK 'out of sequence' wordt het gemiste I-frame herzonden (**implicit retransmission**) (Figuur 177b → en evt. verworpen door S als hij dit al heeft) .
- Explicit retransmission :** (Figuur 178) **S beslist** a.h.v. de ontvangen I-frames of er een frame verloren is en stuurt een **NAK**-frame met dit nummer. (Bij foutloze transmissie worden er uiteraard ook ACK-frames gezonden om P toe te laten zijn buffer te wissen).

In beide gevallen worden alle goed ontvangen frames bijgehouden in S en deze laatste moet ze ook 'in volgorde' doorgeven aan de bovenliggende software laag.

## SELECTIVE REPEAT CONTINUOUS REQUEST MET IMPLICIT RETRANSMISSION

a) Verloren I-frame:

S ontvangt geen goed N+1 frame, maar bij **implicit retransmissie** geeft hij hiervoor **geen NAK frame**. Enkel wanneer het **volgende** frame, N+2, wordt ont-vangen bevestigd hij: ACK(N+2).

P moet hier dan uit afleiden dat N+1 moet worden herzonden, wat dan ook spontaan gebeurd (na N+4).

Alle bevestigde frames verdwijnen uit P's retransmissielijst.

b) Verloren ACK-frame:

**Implicit → Alle** I-frames moeten bevestigd worden. Als ACK(N) verloren gaat zal P, bij ontvangst van ACK(N+1), I(N) herzenden en wachten tot ACK(N) terug-komt.

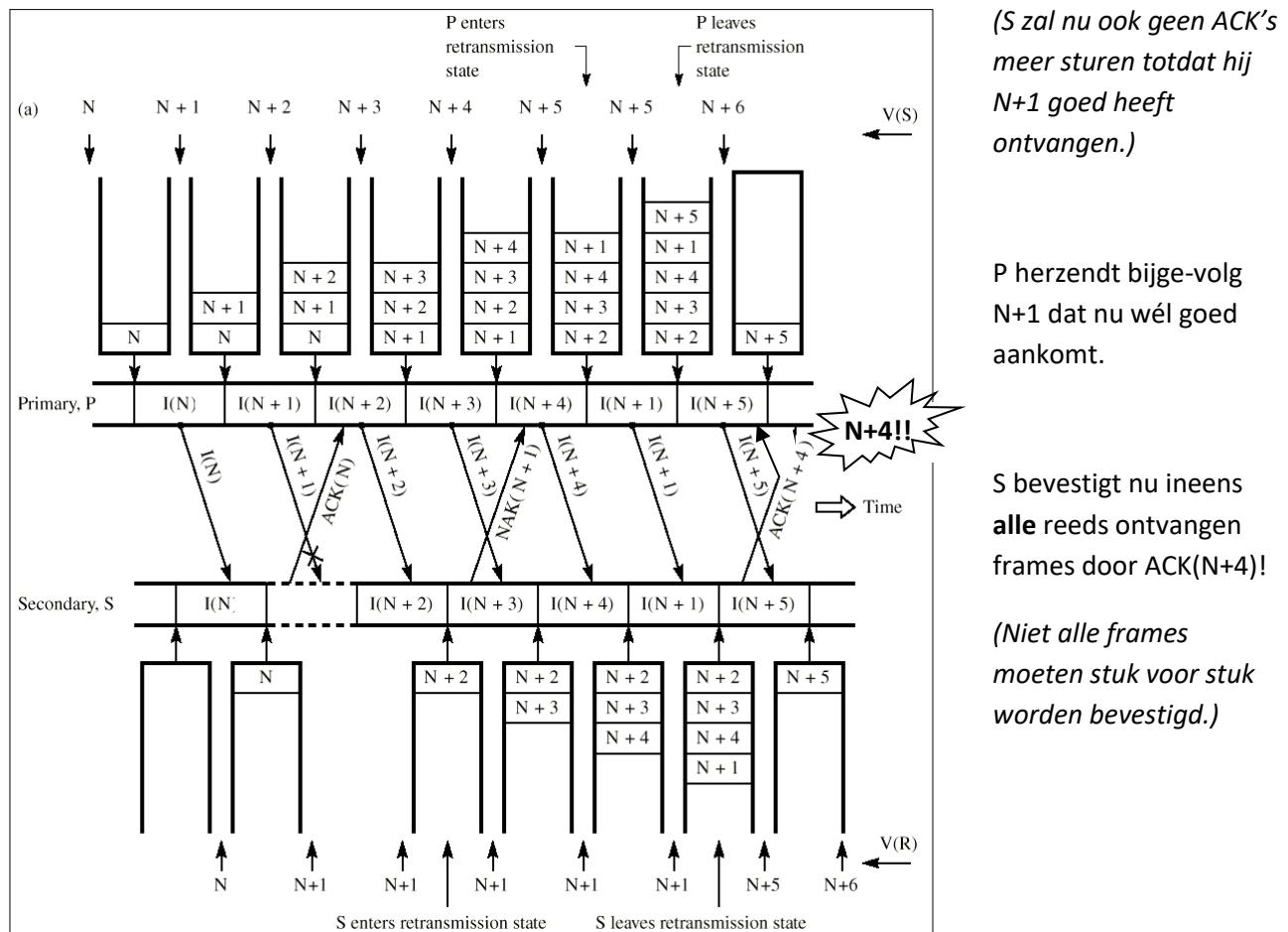
S merkt dat I(N) een dubbel is en verwerpt het.

Figuur 177 selective repeat – implicit retransmission : a) fout I-frame, b) fout ACK frame.

## SELECTIVE REPEAT CONTINUOUS REQUEST MET EXPLICIT RETRANSMISSION

Hierbij moeten **niét alle** frames worden bevestigd, alhoewel er regelmatig een ACK nummer moet binnenkomen om P toe te laten zijn retransmissiebuffer te ledigen. **Slecht** ontvangen frames moeten echter **wéél** worden gemeld door NAK. (soms ook *selective reject* genoemd, SREJ bij HDLC.)

Stel N+1 wordt niet goed ontvangen. S merkt dit pas na ontvangst van N+2 en stuurt NAK(N+1).



Figuur 178 selective repeat – explicit retransmission , fout I-frame

Omwille van de grote buffer nodig in zowel P als S en de noodzaak om het terug **op volgorde** plaatsen van de frames door S wordt in de praktijk veel gebruik gemaakt van **go back N**, ondanks de betere 'link utilisation' van selective repeat.

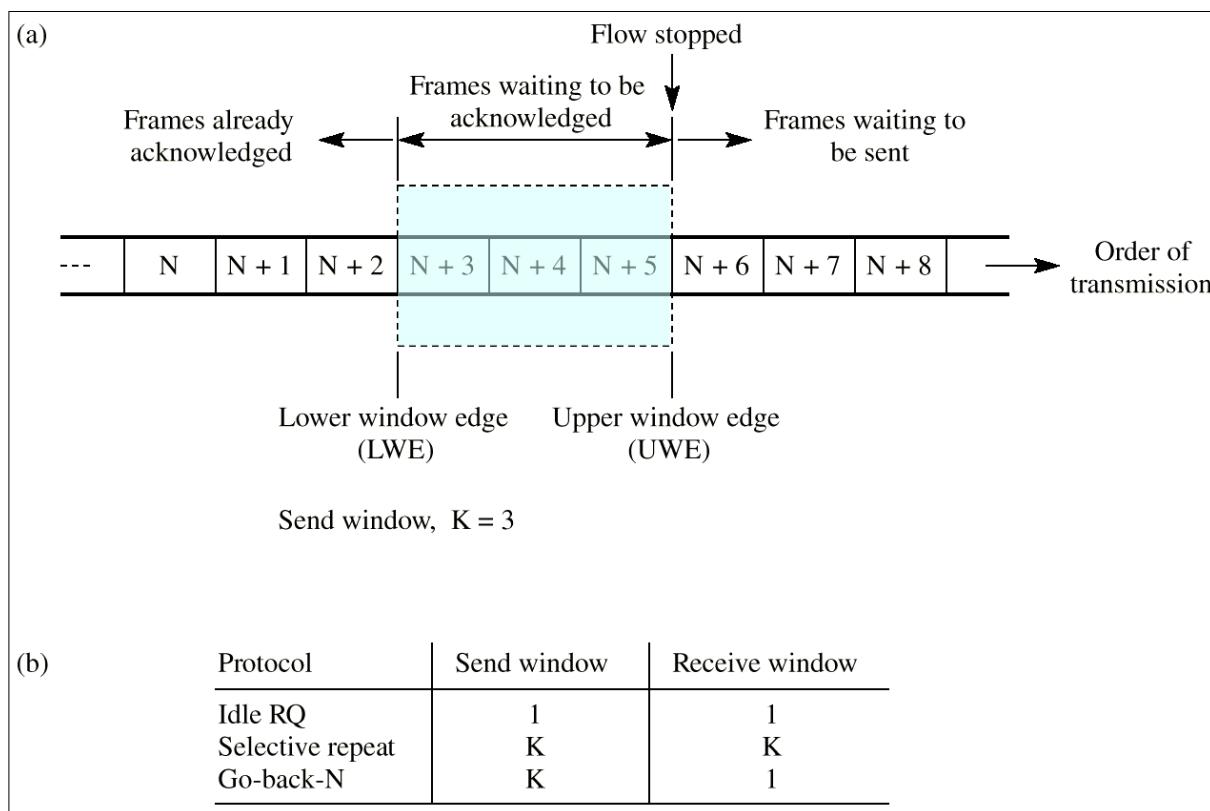
### 2.1.3 FLOW CONTROL

Een volgende belangrijke taak van de datalink-laag is te voorkomen dat een ontvanger overstelt raakt met frames die hij niet kan verwerken. De meest simpele vormen van flow control zijn:

1. Software controle door de bekende Xon-Xoff methode, ook '**in-band flow control**'
2. Hardware controle door RTS –CTS, ... '**out of band flow control**'

Bij voorgaande retransmissieschema's is het echter duidelijk dat intelligentere flow controls moeten toegepast worden. Er moet **afgesproken** worden in het protocol hoe groot de zend- en ontvangst-buffers zijn, m.a.w. hoeveel frames er onbevestigd uit mogen staan → **windowing, of sliding window**.

Er wordt een maximum aantal onbevestigde I-frames toegelaten in de retransmissielist ; **het 'send window'**. K. Is K=1, dan is er geen window en valt men terug op idle RQ met onvermijdelijk verlies van efficiënt gebruik van de datalink. De **window** techniek is m.a.w. alleen van toepassing op **continuous RQ**, waar I-frames kunnen worden gezonden zonder ACK van het vorige frame.



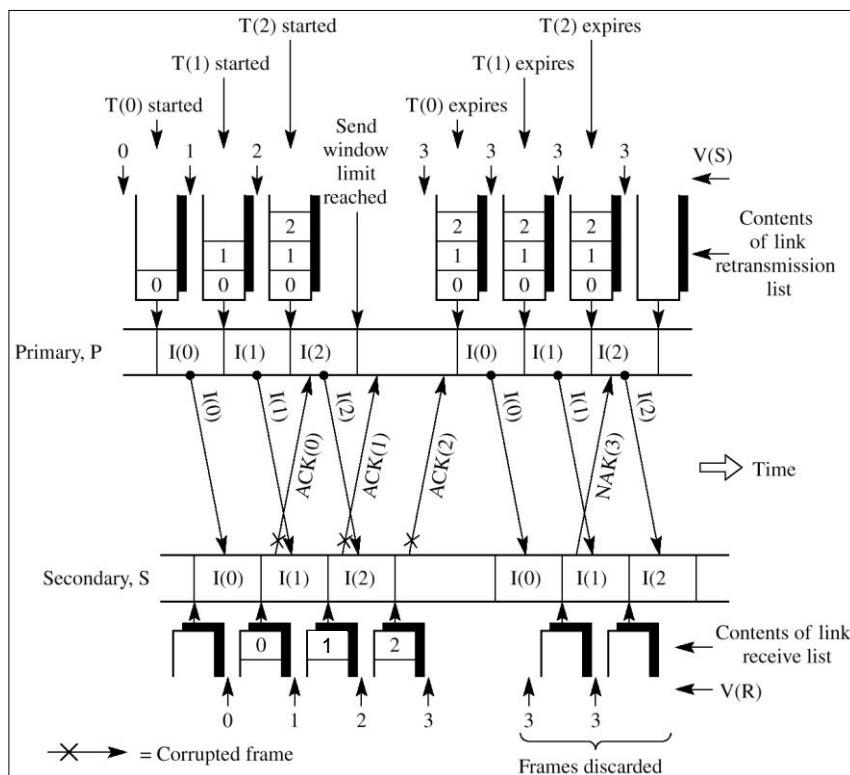
Figuur 179 Sliding window flow control.

Een verzonden pakket vergroot bij de zender de UWE, een ontvangen ACK de LWE. Het opsturen van I-frames stopt als UWE-LWE = K, de max. venstergrootte. Bij foutvrije communicatie is er een venster ( $\leq K$ ) dat over de te zenden frames schuift : '**sliding window**'.

De venstergrootte K (...=V) wordt zo gekozen dat de ontvanger S alle I-frames kan ontvangen zonder dat de flow stopt. Hierbij moeten maximum frame grootte, buffer grootte, vertragingstijden van de link, en bit rate in ogenschouw worden genomen.

Tot nu toe hebben we een volgend frame een nummer meegegeven dat het vorige +1 was, veronderstelend dat er geen grens was. Door de 'window' techniek kunnen we deze nummers nu beperken tot  $K+1$  en een  $(K+1)$ -modulo-telling toepassen : na  $K$  begint terug 0 Bv. modulo 4 ( $M=4 \rightarrow K=3$ ) : 0, 1, 2, 3, 0, 1, 2, 3, 0, ... .

Bv. Stel de venster grootte  $K=3$ . P zend 3 frames, allen goed ontvangen door S. Deze zend ACK frames, die allen fout zijn (of te laat aankomen). De 'time out' tellers van P verplichten hem de frames te herzenden.



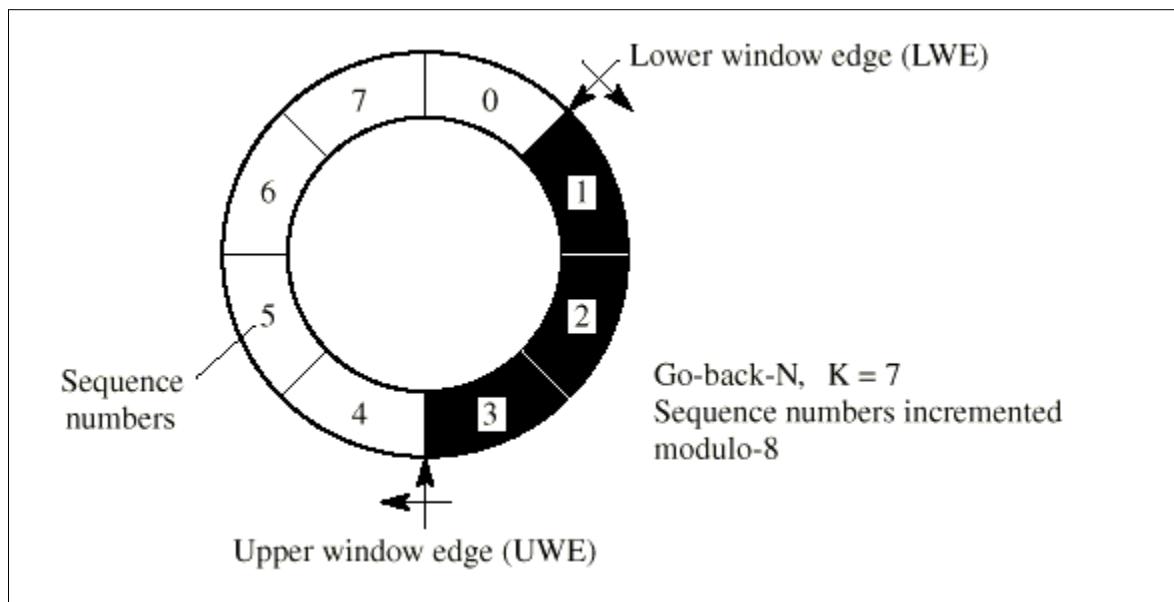
Aangekomen in S verwerpt deze ze en stuurt een NAK(3) waarbij hij meldt dat hij een I(3) frame verwacht :  $V(R) = 3$ .

Aangezien de NAK(nr) =  $V(S)$  (= het volgende te zenden I-frame), veronderstelt P dat dit een ACK was van de 3 wachtende frames en schuift zijn venster verder.

Figuur 180 Maximum te verzenden frames in Go-back-N

Als er nu enkel 3 nummers (=K, bv. 0,1,2) zouden gebruikt zijn kan P niet uitmaken of NAK(0) (i.p.v. NAK(3)) handelt over het reeds verzonden I(0) frame, of over het volgende te zenden I-frame.  $\rightarrow$  het venster = modulo -1 :  $K = M-1$ .

Praktisch worden een **aantal bits** voorzien in de header van elk frame voor de frameteller. Bij HDLC zijn dit er **3** ( $\Rightarrow K=7$ ,  $M=8$ ) of **7** ( $\Rightarrow K=127$ ,  $M=128$ ).



Figuur 181 Windowing techniek voor een venstergrootte  $K = 7$ , modulo  $M=8$ .

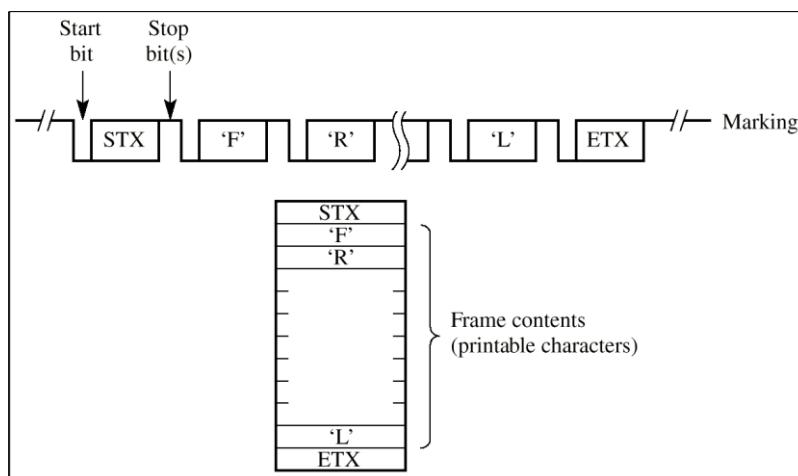
### 2.1.4 TRANSPARANTIE

De datalink moet een onderscheid kunnen maken tussen protocolinformatie en data. D.w.z. dat **alle mogelijke data** moet kunnen overgezonden worden, ook al heeft deze de vorm van bvb. controletekens zoals STX, ETX, ACK, ...

Er moet hier een duidelijk verschil gemaakt worden tussen **bit**-georiënteerde en **byte**-georiënteerde protocollen.

#### BYTE-GEORIËNTEERDE PROTOCOLLEN → CHARACTERSTUFFING.

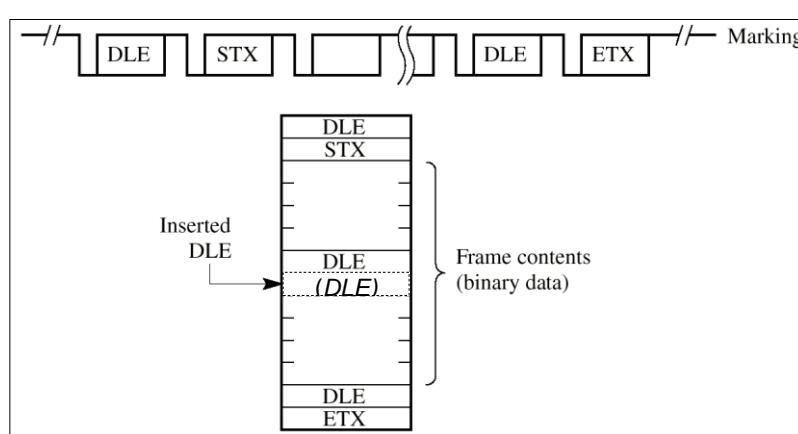
Er worden hierbij een (groot) aantal besturingstekens gebruikt die moeten kunnen onderscheiden worden van de data. We onderscheiden asynchrone- (alle bytes **met** start/stopbit ↗ vb. RS232) en synchrone transmissie **zonder** start/stop-bits.



- **Asynchroon :**

Als deze data bv. **ASCII** karakters zijn is er geen echt probleem : de karakters worden tussen STX en ETX geplaatst, en deze geven resp. het begin en het einde van de datablok aan. STX en ETX kunnen niet midden in het frame voorkomen.

Gaat het echter over **willekeurige** bytes (bv. afkomstig van metingen) Dan kan de *STX* of *ETX* combinatie tot de data behoren. Daarvoor voegt de zender voor de START -STX en STOP-ETX de code **DLE** toe : **Data Link Escape ↗ character- of bytestuffing**.



De start wordt dus uniek = DLE-STX , stop = DLE ETX.

**Figuur 182 Transparantie voor een asynchroon byte-protocol d.m.v. bytestuffing.**

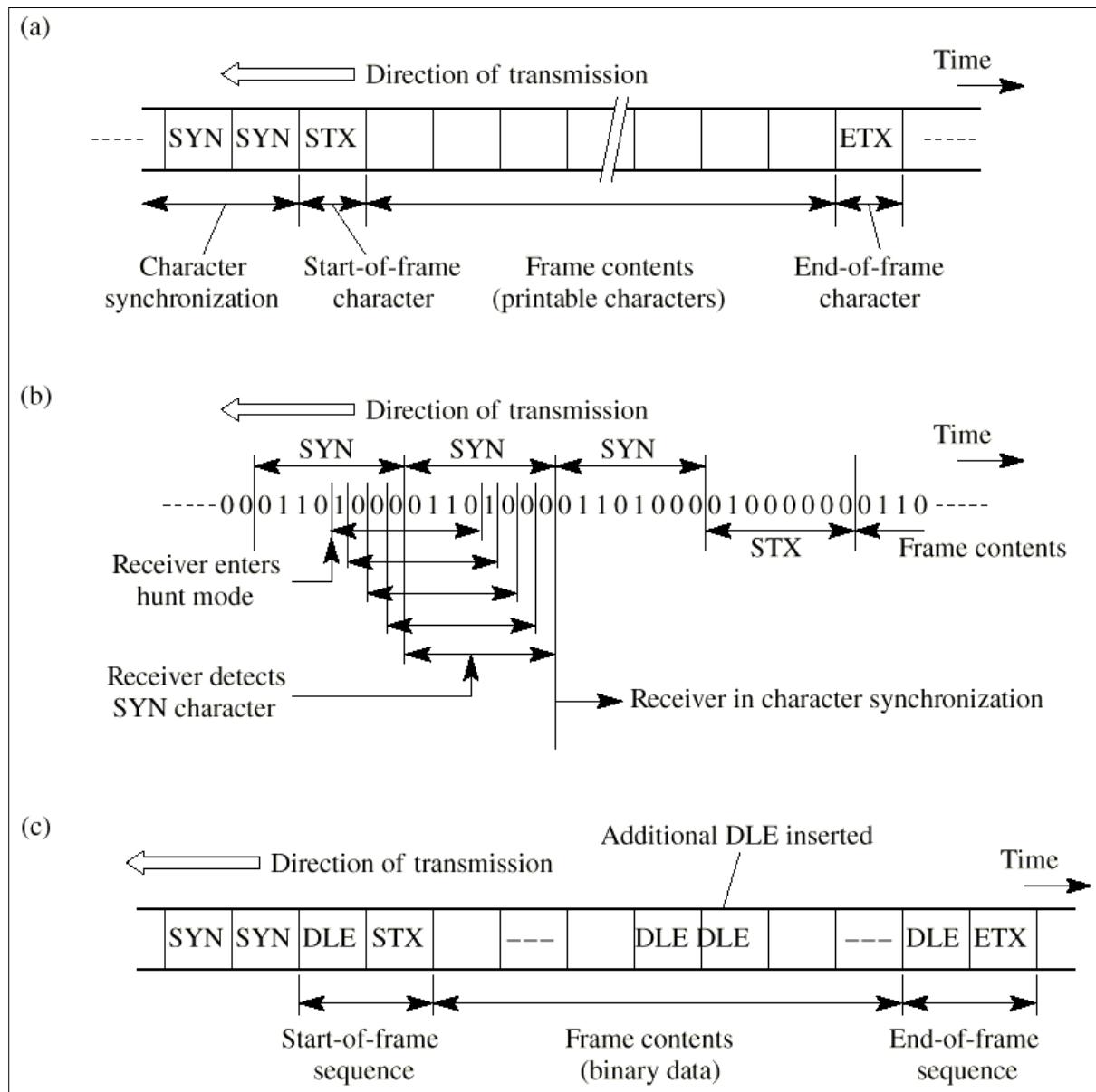
Komt nu in de data óók een *DLE* voor, dan zal ook dit (en enkel dit) worden 'gestuffed' door DLE. Het enkelvoudig strippen van DLE in de ontvanger levert de oorspronkelijke data op.

- **Synchroon :**

Voor karakter georiënteerde **synchrone** transmissie wordt de frame-start **niet** gegeven door een startbit na een rusttoestand ('marking', die bovendien elk karakter terugkomt), maar door een **SYNC** combinatie die de bitsynchronisatie op gang brengt.

Na de eerste SYNC-bits gaat de ontvanger in 'hunt-mode' om een SYNC karakter te onderscheiden, en komt na een tijdje in karakter-synchronisatie. Daarna verwacht hij, al of niet 'gestuffed' de start of frame, resp. DLE-STX of STX. Het frame wordt afgesloten door resp. DLE-ETX of ETX.

In het geval van characterstuffing moet hij uiteraard alle dubbele DLE's herleiden tot 1 DLE.



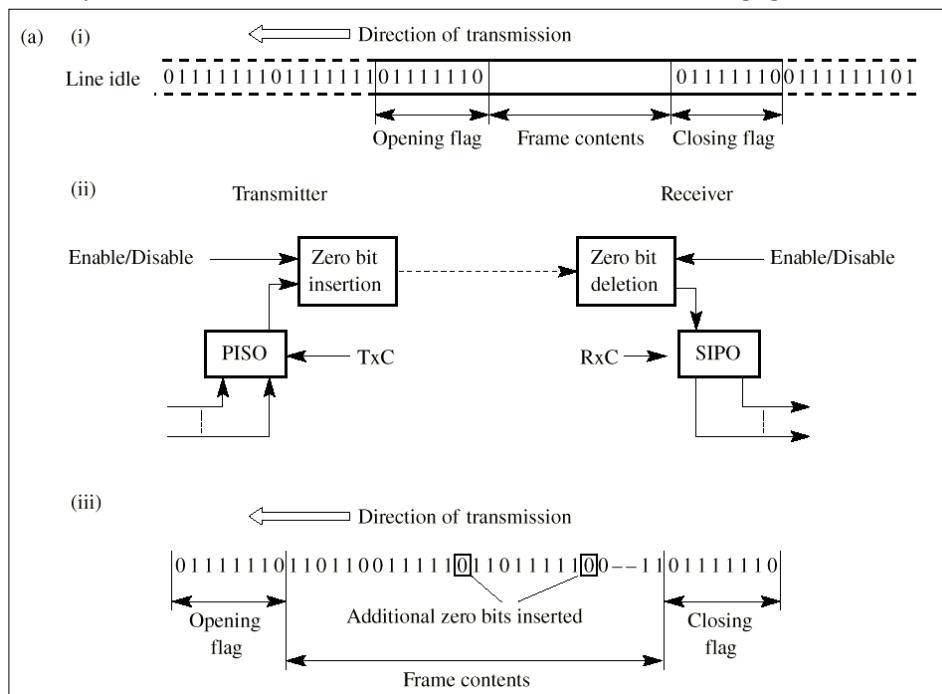
Figuur 183 Transparantie voor een synchrone byte-protocol d.m.v. bytestuffing.

## BIT-GEORIËNTEERDE PROTOCOLLEN.

Karakter georiënteerde protocollen zijn weinig efficiënt voor het transporteren van binaire data vandaar dat tegenwoordig veel **meer** bit-georiënteerde protocollen worden gebruikt. Bit-georiënteerde protocollen zijn per definitie **synchroon**. We onderscheiden 3 verschillende situaties :

### EEN P2P LINK → BITSTUFFING.

In ‘point to point’ connecties zal de zender in de ‘idle’ toestand het patroon 01111111 **aanhouden** om de ontvanger toe te laten zijn bit-synchronisatie te behouden. (voor NRZ codering dient de 0 voor **clocksynchronisatie**). Start en einde van een frame wordt aangegeven door een uniek 01111110 patroon,



**flag** genoemd. Bij ontvangst van de ‘opening flag’ worden de binnengekomen bits onderzocht tot de ‘closing flag’ wordt gedetecteerd, waarbij het frame is afgelopen.

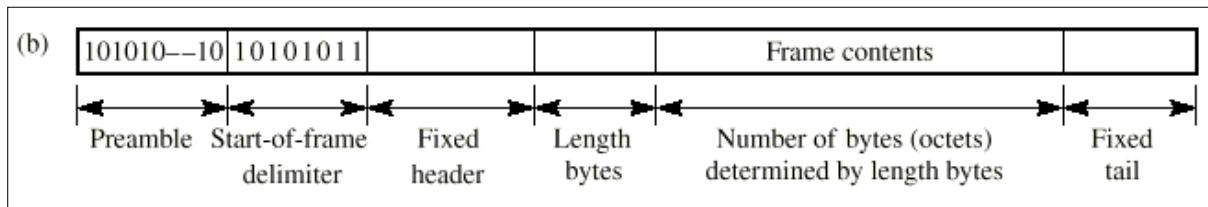
**Figuur 184 Transparantie voor een bit-georiënteerd point to point protocol d.m.v. bitstuffing**

Om **datatransparantie** te verzekeren mag het (idle- en) flag-patroon niet voorkomen in de ‘frame contents’. Hiervoor wordt ‘bit-stuffing’ toegepast : telkens een sequentie van 5 opeenvolgende 1<sup>L</sup>-bits wordt gedetecteerd in het **data-gedeelte** wordt er een 0 tussen gevoegd. Omgekeerd zal de ontvanger na elke 5 opeenvolgende 1’s de 0<sup>L</sup> die daarop volgt strip. Volgt er een 1<sup>L</sup>, dan moet dit een ‘closing flag’ zijn. Op deze manier zal nooit het 01111110 patroon optreden in het frame.

Normaal zal op het einde van een frame een **CRC** opgestuurd worden. Ook dit wordt onderworpen aan de **bitstuffing**.

In de Figuur 184 zien we de PISO (Parallel In Serial Out) met daarna de zero bit insertion die enkel ge’enabled’ wordt gedurende de frame **contents**, niet tijdens idle of open/close flag.

### EEN (ETHERNET-) LAN VERBINDING → BITS (BYTES) TELLEN.



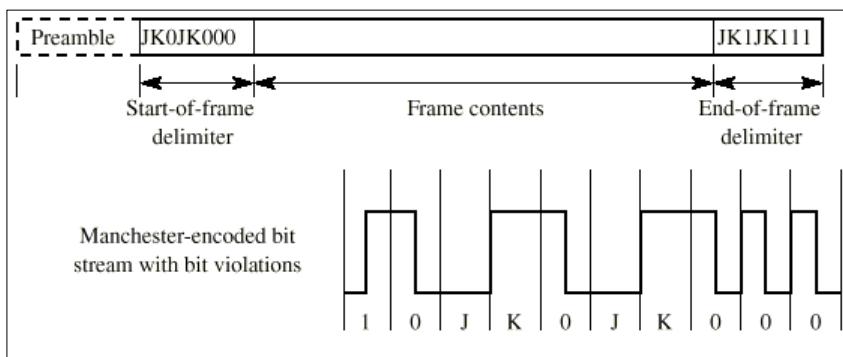
Figuur 185 Transparantie voor een bit-georiënteerd LAN protocol d.m.v. bits tellen.

In LAN's gebaseerd op een 'broadcast' medium dat **gedeeld wordt** door alle stations kunnen de zenders niet continu een 'idle' karakter sturen. Er worden frames verzonden die er uitzien als Figuur 185. (Bvb. toegepast in CSMA/CD (*Carrier Sense Multiple Acces / Collision Detect*) netwerken zoals ethernet ...)

In rust is er geen signaal op de bus, en een station controleert eerst of er niemand aan het zenden is (*Carrier Sense*). Is dat het geval stuurt hij eerst een **preamble**, een opeenvolging van '10' paren om de ontvanger(s) **bit-synchronisatie** toe te laten. Daarna volgt een SFD, *Start of Frame Delimiter*, (..10101011). Vervolgens komt een **header** met vaste lengte, afgesproken in het protocol, die o.a. de zend- en ontvangst-**adressen** bevat. Hierop volgen dan weer 2 bytes die de **lengte** van de 'frame contents' aangeven in aantal bytes, waarop de echte frame inhoud volgt. Het komt er hier nu op aan van **bytes te tellen** om transparantie te verzekeren.

### EEN (TOKEN RING-) LAN VERBINDING → BIT-ENCODING VIOLATIONS.

Start en einde van een frame worden aangegeven door **niet-standaard bitcoderingen**, bit-encoding violations genoemd. Bvb. voor manchester code, zoals gebruikt bij token ring, moet er normaal een



overgang zijn in het midden van de bitperiode. Een J-violation blijft op hetzelfde niveau als de vorige bit, een K-violation gaat naar het tegengestelde niveau, beiden voor de volledige bitperiode.

Figuur 186 Transparantie voor een bit-georiënteerd LAN protocol d.m.v. bit-encoding violations

De start van het frame, SD (*Start Delimiter*), wordt aangegeven door de JK0JK000 code, het einde, ED, door JK1JK111. Hierdoor wordt transparantie verzekerd.

## 2.2 HET HDLC PROTOCOL

**Geschiedenis :** IBM had voor zijn SNA een datalink-protocol ontwikkeld : SDLC (= Synchronous Data Link Control). Het is door ANSI omgezet in een **Amerikaanse** standaard → ADCCP (=Advanced Data Communication Control Procedure) en door ISO tot een **internationale** standaard gemaakt → **HDLC**.

HDLC staat voor High-level Data Link Control en is dé **standaard** voor datalinks of diende als grote voorbeeld. Het wordt zowel gebruikt bij **asynchrone**- als **synchronne** transmissie, met **byte**- zowel als **bitprotocollen** op **point to point** (P2P)link's, **radio** kanalen zoals satellieten of **fysische** en logische linken door een '**switched**' netwerk.

M.a.w. kent u HDLC, dan (her)kent u alle andere datalink protocollen :

- LLC : Logical Link Control in LAN's
- LAPD : Link Access Procedure D channel voor ISDN
- LAPB : Link Access Procedure Balanced → X25
- LAPM : Link Access Procedure Modem
- PPP op internet

De **service** aangeboden door HDLC aan de hogere laag kan zowel '**best-try**' voor **Connection-Less**-verbindingen (**unACK - CL**), als '**reliable**' voor **Connection-Oriented** verbindingen (**ACK-CO**).

'**Best-try**' (**CL**), ook wel **unacknowledged service** genoemd, betekent dat, ongeacht de aanwezigheid van foutcontrolebits (CRC's), foutieve frames gewoon worden **verworpen** door de datalink. Retransmissie is dan een taak van een hoger protocol (bv. layer 4, transportlaag bij LAN's, TCP/IP, UDP/IP, ...). Dit wordt toegepast op netwerken met een **lage BER**, m.a.w. weinig kans op retransmissie, zoals LAN's, ISDN, ... . Zo kent men dus ook 'windowing' technieken op layer 4!!!

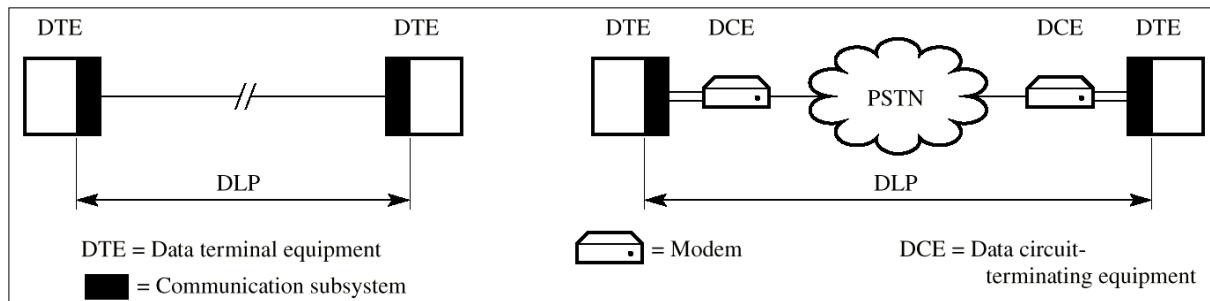
### KARAKTERISTIEKE EIGENSCHAPPEN VAN HDLC

- **Foutdetectie** : op het einde van het frame wordt een **CRC-16** doorgestuurd, **CRC32** voor LAN's
- **Error control** : **Go-back-N** continuous RQ
- **Flow control** : **window**-mechanisme K=7 (M=8 → 3bits) of K=127 (M=128 → 7bits)
- **piggy backing** : bevestigingen van ontvangen I-frames van A→B **kan** door ACK frames van B→A gebeuren, maar efficiënter kan dit eveneens **in I-frames** van B→A.

## 2.2.1 HDLC TOEPASSINGSOMGEVINGEN.

HDLC wordt toegepast in verschillende omgevingen : soms is het Data Link Protocol (**DLP**) gedefinieerd tussen **2** communicerende **DTE's** (hier bv. computers), soms op een 'local link' tss bv. een computer en een **netwerk**. Het protocol heeft dan alleen een **lokale** betekenis.

### POINT TO POINT



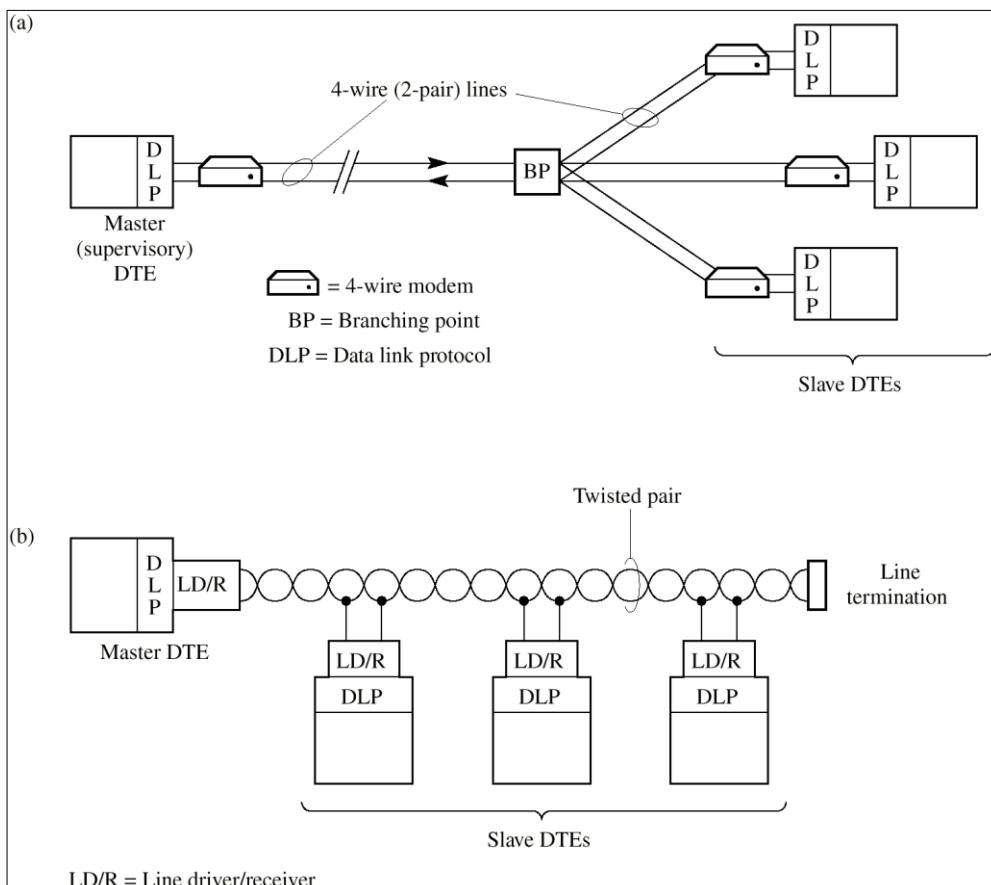
Figuur 187 Data link protocol point to point

Een P2P verbinding kan een directe fysische verbinding zijn (TP, coax, fiber), een geschakeld circuit door een analoge telefooncentrale, maar ook een circuit door een private netwerk-multiplexer of zelfs een radio verbinding met satellieten. De datalink onderhoudt een '**end to end**' verbinding die zelfs soms rechtstreeks de applicatie bedient, zonder tussenliggende lagen. Daarvoor wordt normaal een 'Connection Oriented' (CO) dus 'reliable' service gebruikt.

Voor lage bitsnelheden, bv. modems, wordt ook wel eens een **karakter georiënteerd** idle RQ (stop and wait) protocol gebruikt → bv. **kermit** of **X-modem**. Voor hoge snelheden en lange afstandsverbindingen wordt **HDLC** gebruikt : bit georiënteerd, go back N continuous RQ, windowing.

## MULTIPOINT/MULTIDROP

Fysische lay-out van multipoint/multidrop.



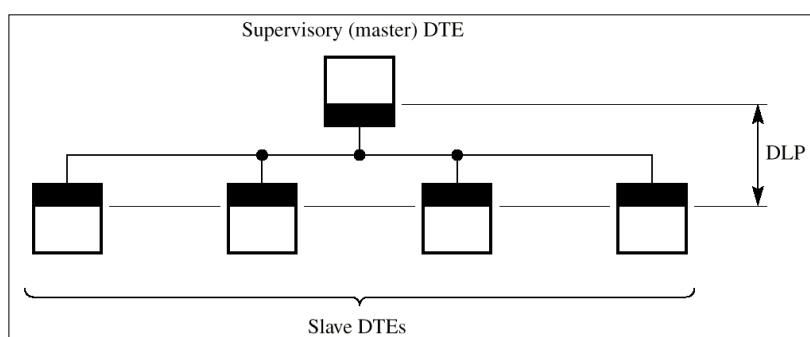
Er is in beide topologien een 'master' en verschillende 'slaves'.

Elke communicatie loopt tussen de **master**, die de verbinding ook controleert, en een geselecteerde (ge'polled'e) slave.

Een reeds besproken voor-beeld : RS485.

Figuur 188 Fysische lay-out van a) multipoint en b) multidrop netwerken.

Logisch zijn beide netwerken voor te stellen als :

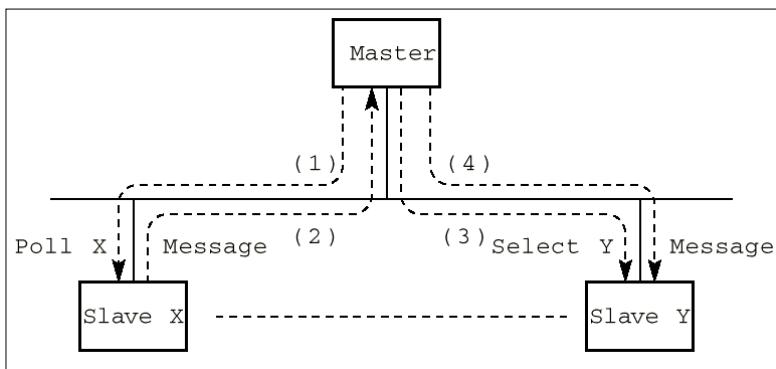


Ook hier wordt een CO (Connection Oriented) DLP gebruikt : vroeger een karakter georiënteerd idle RQ protocol BSC (Bisync or Binary Synchronous Control), recenter HDLC, bit georiënteerd in NRM (Normal Response Mode, zie later).

Figuur 189 Logische lay-out van multipoint / multidrop netwerken

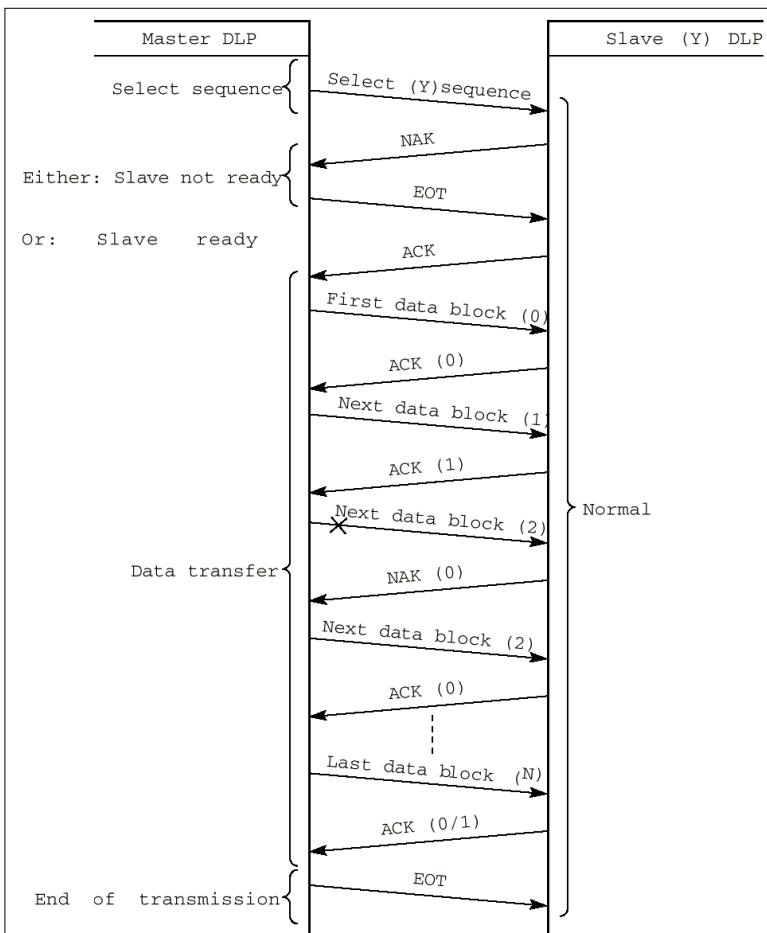
Een bijkomend probleem is dat de DTE's gemeenschappelijk gebruik maken van hetzelfde medium.

Daarom voorziet de datalink een supplementaire deellaag, de **medium-access** laag, zie ook pt. 2.4 op p.181.



Figuur 190 'Poll-Select' werking bij multipoint/multidrop verbindingen.

### SIDE TRACK : POLLING TECHNIEKEN, BSC OF BISYNC- PROTOCOL VAN IBM



Het polling mechanism wordt gebruikt bij **byte georiënteerde** protocollen in master-slave configuraties (zie ook RS485, 4 wire) om communicatie op gang te brengen. De master vraagt ('poll') aan de slave of deze klaar is om data te ontvangen door ENQ en deze antwoord met ACK (of NAK).

In het geval van ACK wordt er een datablok tussen STX en ETX doorgestuurd, en na datum de verbinding verbroken door EOT.

Er is geen venster ⊗ 'idleRQ', m.a.w. de ACK nr. kan volstaan met 0 en 1.

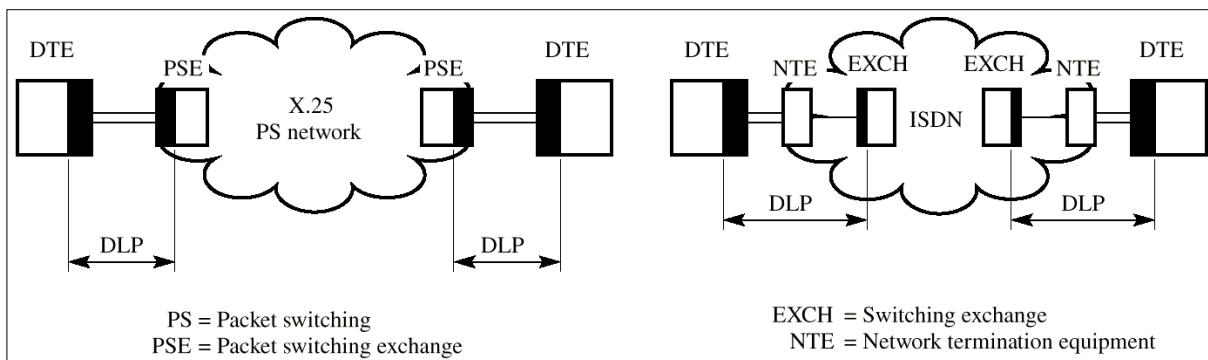
Een station dat niet gepolst wordt mag uiteraard niet reageren.

Figuur 191 Een BSC – select sequentie.

Bij netwerken, die geen onderscheid maken tussen masters en slaves en met **bit**-protocollen werken, worden speciale 'medium access' technieken gebruikt zoals CSMA CD of Token passing, zie verder.

### WAN

Er wordt onderscheid gemaakt tussen X25 (oud) en ISDN netwerken.

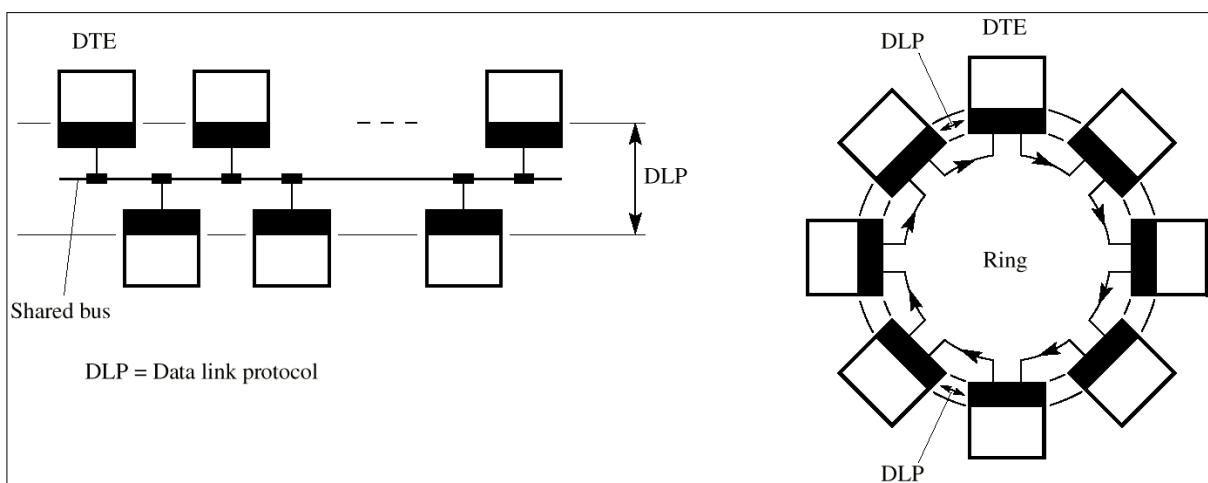


Figuur 192 Data link protocol bij WAN's

Bij X25 heeft het DLP enkel plaatselijke betekenis tussen de DTE en PSE. Het is gebaseerd op HDLC en wordt **LAPB** (Link Access Procedure Balanced) genoemd.

Voor ISDN wordt er eerst een verbinding opgezet ('call setup procedure' via kanaal D) doorheen het netwerk met als gevolg een P2P verbinding met de EXCH, ook wel virtueel circuit (VC) genoemd, voor de datatransfer via de B-kanalen. Het protocol kan zowel CO, reliable, als CL, best-try, zijn en is een variant op HDLC : **LAPD**, link access procedure D channel.

### LAN



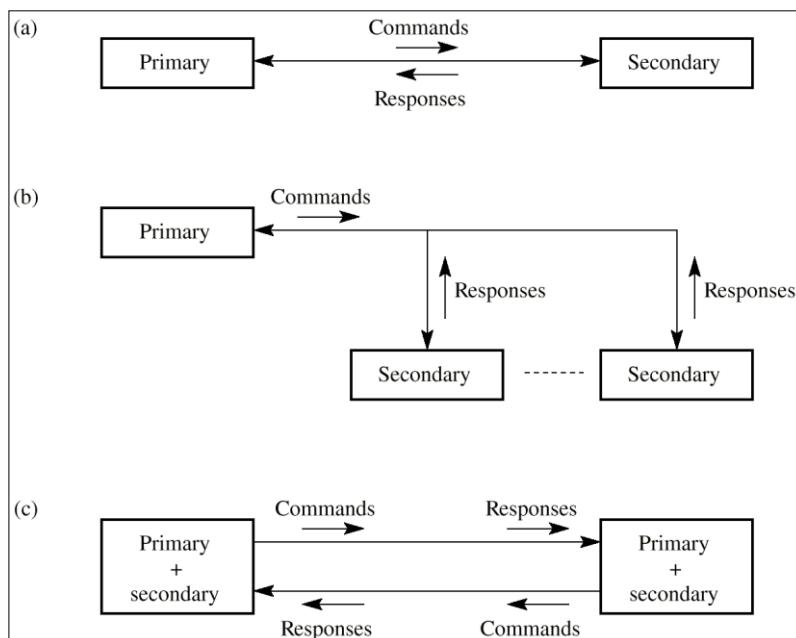
Figuur 193 Data link protocol bij LAN's

Lan's gebruiken meestal relatief korte en **foutarme** verbindingen met hoge 'bit rates' (10 ... 100 ... 1000 Mbps). Omwille van de **zeer lage foutkans** werken deze netwerken normaal **CL**, best try, op de datalink. Alle error- en flow-controle functies worden overgelaten aan een **hoger** protocol, bv. layer 4 of TCP, UDP. Het link-protocol bij LAN's wordt **LLC** genoemd : **Logical Link Control**.

Ook hier is er het probleem van de 'multiple access' op het medium. De oplossingen hiervoor worden besproken in pt. 2.4, op p. 181.

## 2.2.2 HDLC STRUCTUUR.

### NETWERK-CONFIGURATIES EN WERKINGSMODI



De frames die verzonden worden van Primary → Secondary worden 'commando's' genoemd, van S → P : 'responses'.

De eerste 2 configuraties a) en b) hebbens slechts 1 Primair station (master) wat neer komt op een 'unbalanced' configuratie.

Figuur c) heeft stations die zowel primair als secundair kunnen zijn, 'combined' stations, en deze configuratie wordt 'balanced' genoemd.

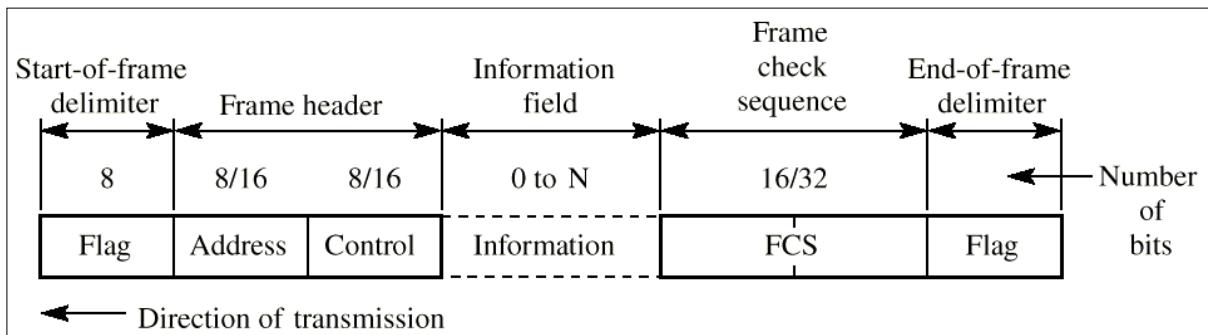
Figuur 194 HDLC – netwerkconfiguraties

HDLC kent 3 werkingsmodi :

1. **NRM** : Normal Response Mode. Dit wordt gebruikt in een 'unbalanced' configuratie a) en b). Hierbij mogen 'slaves' (=secondary, S) alleen zenden op uitdrukkelijk verzoek ('poll') van de master (=primary, P).
2. **ARM** : Asynchronous Response Mode, ook gebruikt in een 'unbalanced' configuratie. Nu mogen 'slaves' (=secondary, S) wel zenden zonder toelating ('poll') van de master (P). (*Dit wordt normaal toegepast bij P2P duplex verbindingen (a) : slechts 1 slave die zendt op een andere lijn dan P. Zeldzaam → meer NRM of ABM*).
3. **ABM** : Asynchronous Balanced Mode. Uiteraard enkel voor 'balanced' configuraties c). (*P2P verbindingen waar beide toestellen zowel P als S zijn en een gelijke status hebben*).

## FRAME FORMAAT

Er zijn 2 vormen : standaard (veelvouden van **8** bits) of extended (~ **16** bits)



Figuur 195 HDLC frame formaat.

(Zie ook Figuur 184 op p. 152)

- **Start- en sluitvlag** (=01111110) : hiertussen zit het frame vervat, transparantie door **bitstuffing**.
- **Adres** (8/16bit) : hangt af van de werkingsmodus. In NRM op multidrop krijgt elke **S** een uniek adres, wat in dit veld wordt ingevuld als het frame van **P** → **S** gaat (**commando's**). Bij **responses** (**S** → **P**) staat hier **terug** het adres van **S** (slave!, secondary!). Zie ook opm<sup>1, 2, 3</sup> en <sup>4</sup>.
- **Controleveld** (8/16bit) : De eerste commando's tussen **P** en **S** overleggen over de lengte van dit controleveld. Lay-out → zie frame types.
- **Informatieveld** : bevat de eigenlijke **data** bij een **I-frame** of **niks** bij **S-** (en **U-**)frames, zie verder. De lengte van dit veld is niet vastgelegd in de HDLC-norm, dit moeten de deelnemers zelf vastleggen.
- **FCS**, Frame Check Sequence: Dit is een CRC-16 van ISO en ITU-T. In de berekeningen worden niet de start- en einde- vlaggen opgenomen, en ook niet de ge'stuffed'e 0-bits, maar FCS wordt zo nodig zelf wél gestuffed.

<sup>1</sup> Naast **unieke** adressen worden ook **groepadressen** toegekend aan meer dan 1 **S** of zelfs **broadcastadressen** aan alle **S**. Voor **broadcast** zijn alle adresbits=1.

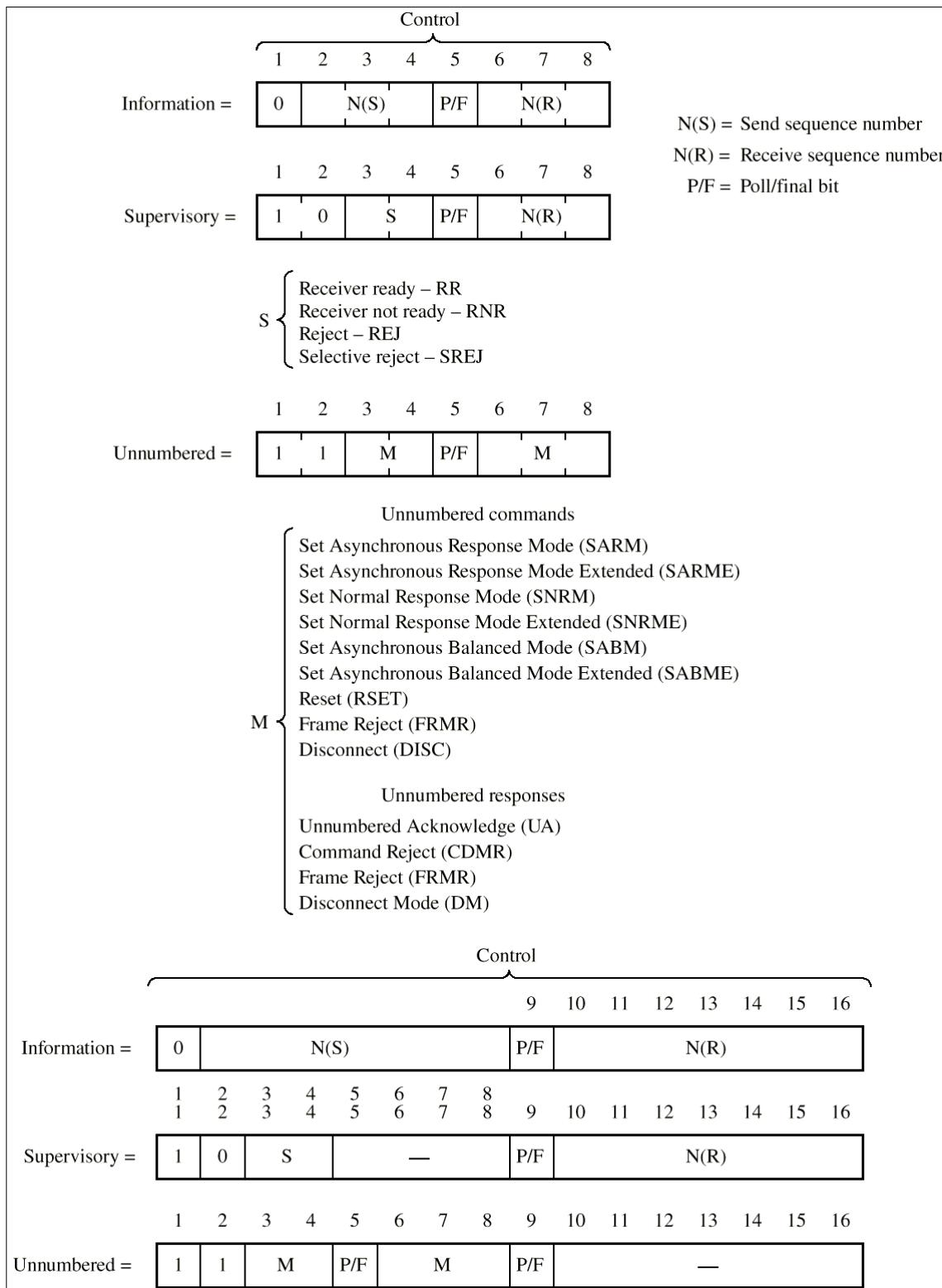
<sup>2</sup> Bij grote aantallen **S** kan het adresveld worden uitgebreid voorbij de 8bits. De LSBit van elk 8-bit adresveld geeft aan of er nog een adresbyte volgt (→ LSB=0), of dat dit de laatste (of enige) adresbyte is (→ LSB=1).

<sup>3</sup> Voor ABM komt de adrestoekenning op hetzelfde neer, alleen kan 1 station tegelijk **P én S** zijn. M.a.w. een commando van **1** → **2** draagt adres 2, de respons hierop (van **2** → **1**) draagt óók adres **2**.

<sup>4</sup> In vele gevallen zal zowel bestemmings- als zender-adres worden opgegeven, bvb. ethernet.

### FRAME TYPES :

In het **controleveld** wordt onderscheid gemaakt tussen **3** soorten frames : I, S en U.



Figuur 196 HDLC frame types in standaard en extended vorm.

- I-frame :** (bit1 = 0). Deze transporteren de eigenlijke **data**, zie ook Figuur 195. Ze kunnen bovendien ook ‘piggy backed’ een **ACK**-controle dragen voor frames in de andere richting via N(R).

- **S-frame** : (bit1-2 = 1-0). Ze worden gebruikt voor **error-** en **flow-control**, en bevatten dus ontvangers N(R).
- **U-frame** : (bit1-2 = 1-1). Gebruikt voor ‘link control’ : opzetten, afbreken en beheren, bv. SNRM: P meldt aan S dat hij wil communiceren in NRM mode. U-frames bevatten **nooit** ACK-informatie, dus geen frame-nummers → ‘unnumbered’.

Merk op dat in HDLC de **LSBit’s eerst** worden opgestuurd, m.a.w. dat de ontvanger na 1, max. 2 bits weet welk type frame binnenkomt.

- **I-frame** : N(S), de zendteller geeft het frame nummer aan van het te verzenden frame.  
N(R), de ontvangsteller, geeft ‘piggy backed’ een ACK voor de ontvangen frames.  
N(R) geeft het frame aan dat **verwacht** wordt in de ontvanger (m.a.w. het **volgende**).  
N(S) en N(R) zijn in een standaard controleveld 3 bits groot → modulo M=8, max. venster K=7. In een extended vorm 7 bits groot → modulo M=128, venster K=127.
- **S-frame** : de 2 S-bits, functiebits, maken een onderscheid tussen de 4 S-frames : RR, RNR, REJ en SREJ.  
Opm<sup>1</sup>  
Aangezien dat S-frames een N(R) teller dragen dienen ze als ‘gewone’ **bevestiging** (ACK) van ontvangen frames (m.a.w. **niet** piggy backed). Het is dus een verkeerde veronderstelling dat Supervisory frames commando’s zouden zijn : het zijn **responses**!
- **U-frame** : de 5 M-bits, 2+3 functiebits, maken een onderscheid tussen (meer dan 16) U-frames waarvan de belangrijkste zijn vermeld in Figuur 196.  
Hierin zitten zowel **commando’s** als **responses**,  
Deze frames bevatten geen N(S) of N(R) nummer → unnumbered.
- **P/F vlag** : de poll/final vlag heeft **2** mogelijke functies afh.v.h.frametype (commando of respons)  
(C) → **poll**-functie : bv. een master vraagt (*poll*) een slave om te antwoorden → p/f=1,  
(Als P dit niet vraagt → p/f=0.).  
de slave zal dit moeten beantwoorden (→ response R) met p/f=1 → **final**.  
(R) → **final**/functie : de slave antwoord op een poll → p/f=1.

---

<sup>1</sup> RR en RNR geven aan of S (receiver) data **kan** (of niet→RNR) ontvangen, of het is een **ACK** van een I-frame.

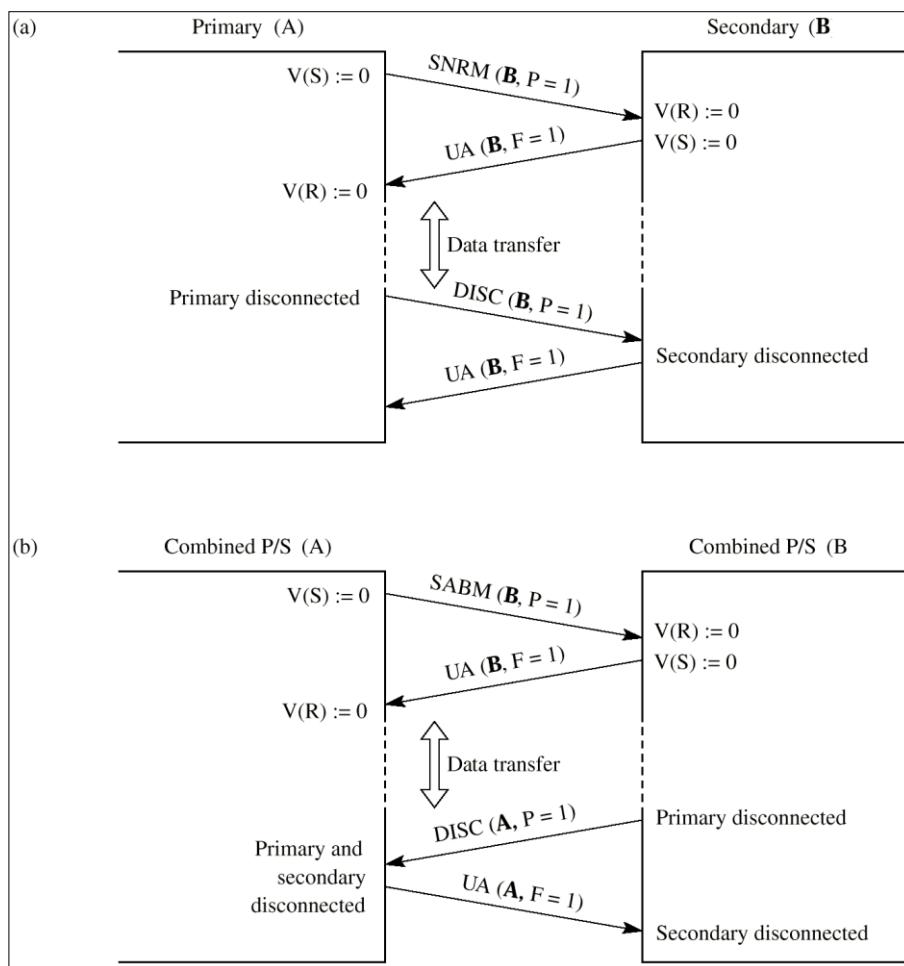
REJ en SREJ worden enkel gebruikt in ABM en geven aan dat een I-frame ‘out of sequence’ ( $N(S) \neq V(R)$ ) is ontvangen, REJ voor go-back-N, SREJ voor sel.repeat.

## PROTOCOLWERKING VOOR EEN CO-ACK SERVICE.

In eerste instantie start de link in **standaardmode** met  $N(S) = N(R) = 3$  bits, een venster van 7 frames wat voldoende is voor de meeste toepassingen. Voor zeer lange (bv. satelliet-) verbindingen of zeer hoge datarates is dit venster te klein en wordt er overlegd om de **extended** vorm (7 bits →  $K=127$ ) te gebruiken, zie de unnumbered frames SNRME, SARME, SABME.

### LINK SETUP.

Het opzetten van een link gebeurt door uitwisseling van een aantal unnumbered frames.



Figuur 197 Link management in a) NRM, b) ABM.

Tegelijk worden de I-frame-tellers  $V(S)$  en  $V(R)$  in P en S geïnitialiseerd, en start de dataflow (zie verder). Nadat alle informatie is verzonden stuurt de master P, een DISC (disconnect) frame (met adres B, p=1) waarop S antwoordt met UA (met adres B, f=1).

b) Voor ABM mode is de setup praktisch hetzelfde via SABM. Data kan vloeien in beide richtingen, onafhankelijk van elkaar (Asynchrone **Balanced** Mode). Nu kan station B echter zelf de verbinding afbreken, aangezien het óók een master is. Let op het adres (A) dat nu meegegeven wordt in Command én Response.

Weigert de ontvanger een link-setup in gelijk welke mode dat antwoordt hij met DM : Disconnected Mode (op de vraag SNRM of SABM).

Tegelijk worden de I-frame-tellers  $V(S)$ ,  $V(R)$  in P én in S geïnitialiseerd.

a) In een multidrop verbinding zal P (master) eerst een SNRM (Set Normal Response Mode) frame naar S (slave) sturen met adres van de slave B én poll vlag p/f=1, m.a.w. vragend om antwoord.

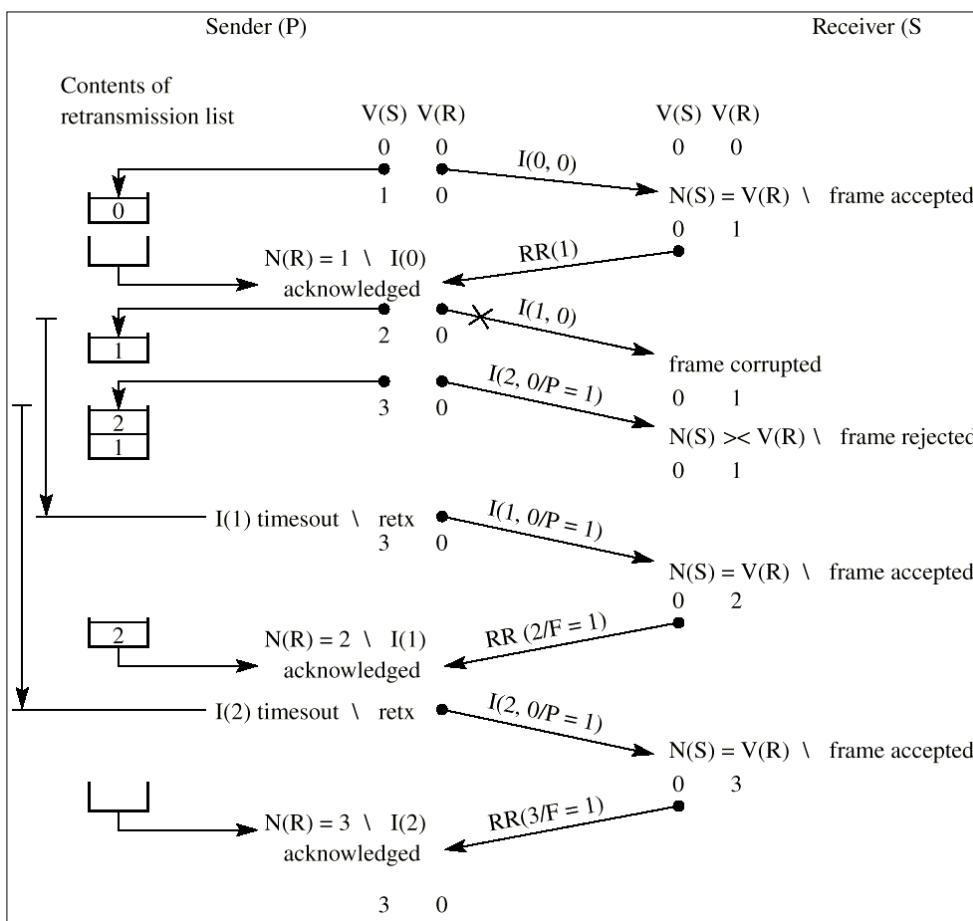
Als de slave kan ontvangen zent hij een UA (Unnumbered Acknowledge) terug, met **zijn eigen** adres B (er is maar 1 master in NRM) en p/f=Final = 1.

## DATA TRANSPORT IN NRM MODE.

We bespreken enkel de 2 meest voorkomende procedures : NRM en ABM, zie Figuur 194 .

Alle datatransport wordt gecontroleerd door P. Wil de P info van S dan stuurt hij een **Poll** : UP (Unnumbered Poll) met  $p=1$ . Heeft S niks dan zendt hij **RR** (receiver ready) terug met  $f=1$ . Is er wel data, dan stuurt hij de I-frames met in het laatste  $f=1$ .

Herinner u : HDLC = **Go-back-N, windowing!**



In Figuur 198 beschouwen we voor de eenvoud enkel verkeer in 1 richting waarbij elk I-frame een ACK via (=RR) verwacht.

Beide kanten hebben een zend- **V(S)** en ontvangersteller **V(R)**.

**Zender P:**  $V(S)$  duidt het volgende te zenden I-frame aan, en wordt meegegeven in dit frame als  $N(S)$ .

Figuur 198 Dataverkeer op HDLC in NRM mode, Go-back-N, implicit ACK

**Ontvanger S :**  $V(R)$  geeft het te verwachten I-frame en wordt vergeleken met het ingesloten  $N(S)$ . Is  $V(R) = N(S)$  dan is het frame '**in sequence**' (go-back-N!), wordt het geaccepteerd,  $V(R) := V(R)+1$ , en de nieuwe  $V(R)$  wordt teruggezonden als  $N(R)$  in een **RR** of 'piggy backed' I-frame als **ACK**.

De andere kant zal  $N(R)$  gebruiken om de I-frames [tot  $N(R)-1$ ] uit zijn retransmissielist te schrappen.

**S :** Een I-frame  $I(1,0/ p=0)$  heeft een foute CRC en wordt verworpen. Dit is implicit ACK ↗ alleen ACK's ↗  $RR(1)$  als ACK van  $I(0,0)$  was reeds verzonden, m.a.w. de ontvanger reageert niet.

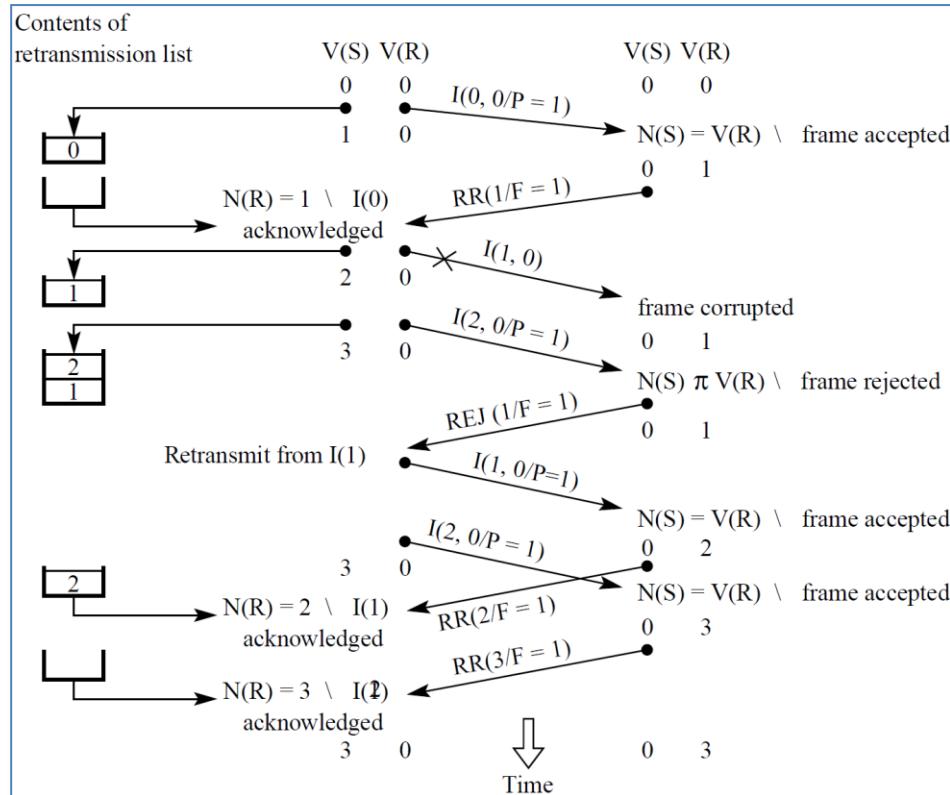
**S :** Een I-frame  $I(2,0/ p=1)$  waarvan  $N(S) \neq V(R)$  wordt verworpen, hij verwacht  $I(1,0)$ . In dit I-frame is  $p=1$  gezet door P : er wordt een reactie gevraagd van S, maar dit is implicit RQ ↗ S doet niks.

**P** : I(1) wordt ge'timeout'.  $\rightarrow$  herzonden (met  $p=1$ )  $\rightarrow$  N(S)=V(R)  $\rightarrow$  OK  $\rightarrow$  S antwoordt met RR(2/f=1) als antwoord op de poll  $\rightarrow$  I(1) ACK, dus uit retx. lijst.

**S** : I(2,0/ p=1) was verworpen wegens N(S)  $\neq$  V(R)  $\rightarrow$  geen ACK van I(2,..)  $\rightarrow$

**P** : I(2) wordt ge'timeout'.  $\rightarrow$  herzonden  $\rightarrow$  N(S)=V(R)!  $\rightarrow$  OK  $\rightarrow$  S antwoordt met RR(3/f=1)  $\rightarrow$  I(2) ACK, dus uit retx. lijst.

In Figuur 199 worden foutieve frames aangegeven door S door REJ  $\rightarrow$  explicit retransmissie.



**Explicit** kan een REJ frame teruggezonden worden op  $N(S) \neq V(R)$  wat de retransmissie sneller maakt zowel voor I(1,0) als I(2,0).

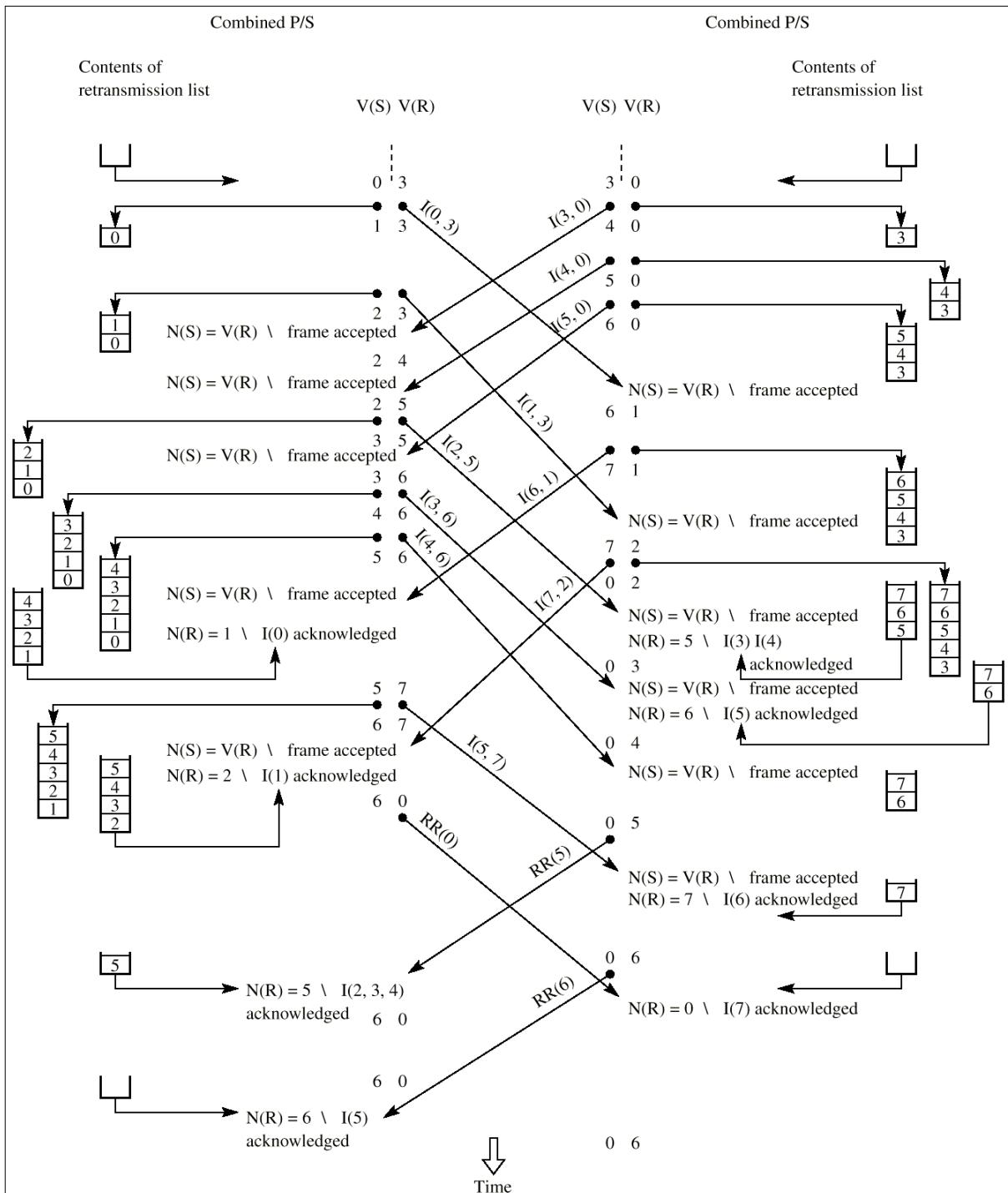
Dit is ook Go-back-N, dus I(2,0) wordt verworpen.

Figuur 199 Dataverkeer op HDLC in NRM mode, Go-back-N, explicit ACK

(Moest selective repeat gebruikt worden, dan heeft S een buffer nodig en kan hij dus meer frames opslaan, dan zou I(2,0/p=1) worden aangenomen en een SREJ(1/f=1) worden teruggestuurd.)

## DATA TRANSPORT IN ABM MODE.

Bij gebalanceerde stations is **duplex** verkeer mogelijk en kunnen ACK's in terugkerende I-frames worden ge'piggy backed'. We veronderstellen foutvrije communicatie.



**Figuur 200** Dataverkeer op HDLC in ABM mode, foutloos, met 'piggy backing'

→ Van elk binnenkomend frame wordt **N(S)** gecontroleerd met **V(R)**.

$N(S) = V(R)$   $\oplus$  I-frame aangenomen,  $V(R) := V(R) + 1$

$N(S) \neq V(R)$  ? verworpen, REJ- of SREJ-frames worden teruggestuurd.

→ **N(R)**, de teller die ‘piggy back’ in een I-frame van de andere richting wordt meegestuurd, wordt gebruikt om uitstaande frames ( $A \rightarrow B$ ) in de retx lijst te ‘acknowledgen’. Zijn er geen I-frames ( $A \leftarrow B$ ) meer te zenden, dan worden RR-frames opgestuurd met N(R).

De **flow-control** werkt met een **3-bits** venster van **7** mogelijke uitstaande I-frames. S en U-frames tellen hierin niet mee, en kunnen dus altijd worden gestuurd, ook al is het max. venster bereikt waardoor I-frames in deze richting stoppen. (→ ACK voor andere richting kan niet meer ‘piggy back’ → RR frames).

Max venster →  $UWE = LWE + K$ , of  $V(S) = \text{laatste } N(R) + K$

## 2.3 PRAKTISCHE DATALINK-PROTOCOLLEN.

We zien hier enkel de voor ons belangrijkste realisaties op basis van HDLC.

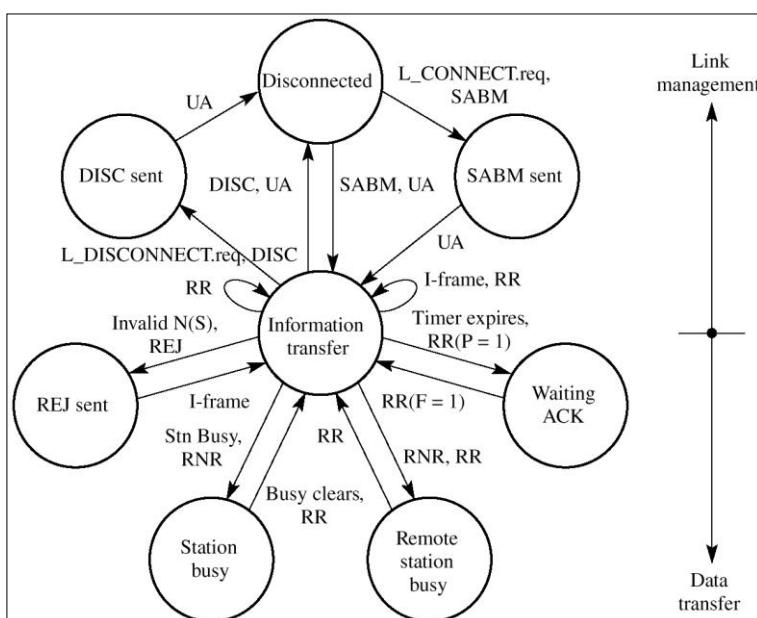
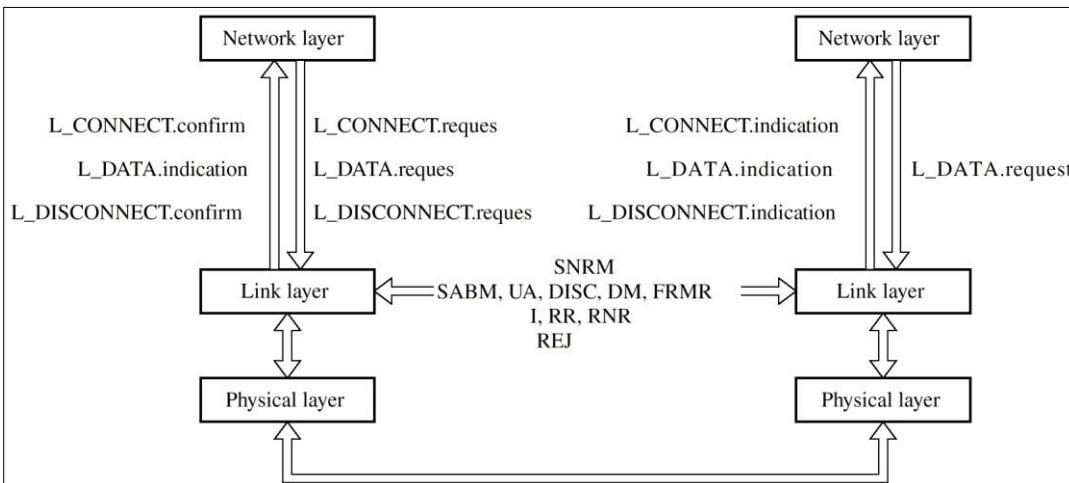
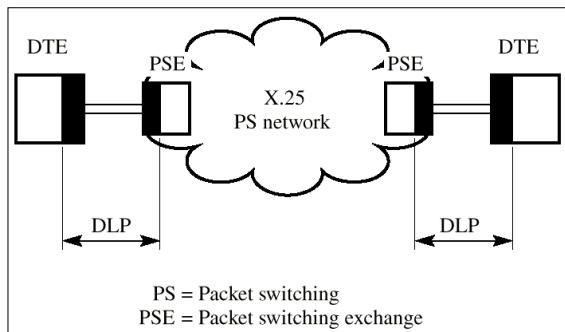
### 2.3.1 LAPB.

Link Access Procedure version B is een ‘subset’ van HDLC dat gebruikt wordt voor X25, ‘packet switched’

netwerken waarvan we de verdere eigenschappen bespreken in de cursus CO-PDN.

Het zijn P2P duplex linken tussen DTE en PSE, beiden **gecombineerde stations** → werken in ABM.

De service primitieven naar de netwerklaag zijn hieronder beschreven. Tezamen met de gebruikte commando's en reponses op de datalink.



**REJ**-frames worden gebruikt voor **error**-controle, **RNR** voor **flow**-control.

Beide stations kunnen de link opzetten. In normale SABM mode wordt er slechts 1 byte gebruikt als controleveld, m.a.w. N(S) en N(R) zijn 3 bit, een venster van 7 frames.

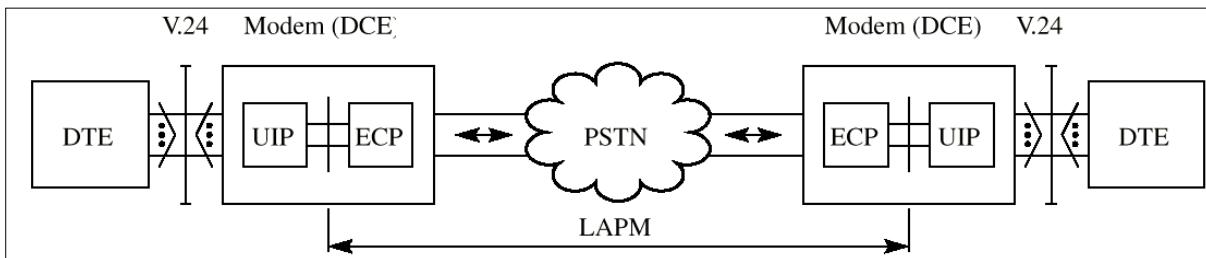
Voor lange links wordt ook wel SABME gebruikt → 7 bit tellers, max 127 uitstaande frames.

Figuur 201 LAPB werking : link, service primitieven en state diagram.

### 2.3.2 LAPM.

Gebruikt bij modem-verbindingen, bv. V32. Deze modems ontvangen data **asynchroon** (start-stop) via RS232 (=V24), maar verzenden ze **bit-georiënteerd synchroon**.

Figuur 202 Een modemverbinding.

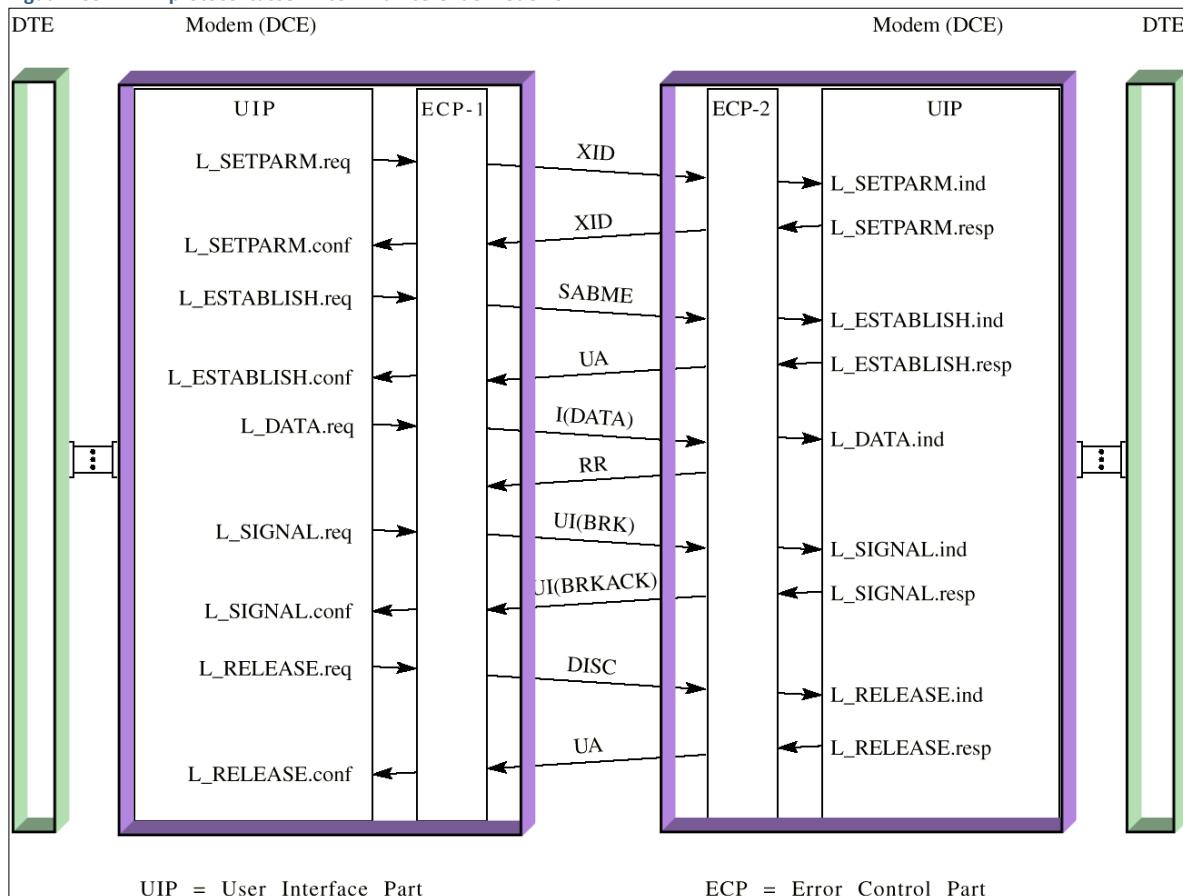


Elke modem bestaat uit 2 delen :

- **UIP** : User Interface Part, communiceert asynchroon via RS232
- **ECP** : Error Correction Part, communiceert over de telefoonlijn **bit-synchroon** LAPM.

De **UIP** communiceert met **ECP** via een set van 'service primitives' die op de link worden vertaald in HDLC frames.

Figuur 203 LAPM protocol tussen 2 communicerende modems.



Voor een link is opgezet moet er o.a. overlegd worden over (anders ‘default’ waarden) :

- Lengte van het I-frame
- Time-out tellers voor de ACKnowledges
- Max. aantal retransmissies van een I-frame
- Venstergrootte

Dit gebeurt door het primitief **L\_SETPARM.request** met de gewenste parameters. Er wordt door de 2 ECP’s overlegd via 2 U-frames : **XID**, exchange identification.

Zijn de parameters afgesproken, dan kan een link worden opgezet via **L\_ESTABLISH.request** → ECP-1 zet **SABM(E)** op link. → Ontvangende ECP-2 geeft aan UIP **L\_ESTABLISH.indication** → **L\_ESTABLISH.resp** → ECP-2 zet **UA** op de link ECP-1 vertaalt dit in ‘**confirm**’ en de link staat.

UIP-1 verzamelt vervolgens de bytes die hij ontvangt via RS232, en geeft de volledige blok door aan zijn ECP-1 door **L\_DATA.request**. De ECP-1 verpakt dit in een **I-frame** en transporteert het. → ECP-2 (die ACKnowledged door RR) → UIP-2 → DTE-2 computer.

Ontvangt bv. UIP-1 een X-off conditie van **DTE-1** (of DTR is inactief), dan stopt hij met data door te sturen op de RS232 lijn naar de DTE-1, en geeft een **L\_SIGNAL.request** aan ECP-1. Deze verstuur hierdoor een **UI, Unnumbered Information frame (BRK)** naar ECP-2 → **L\_SIGNAL.indication** UIP-2 → ... **L\_SIGNAL.response** naar ECP-2 → **UI = 'BRKACK'** naar ECP-1.

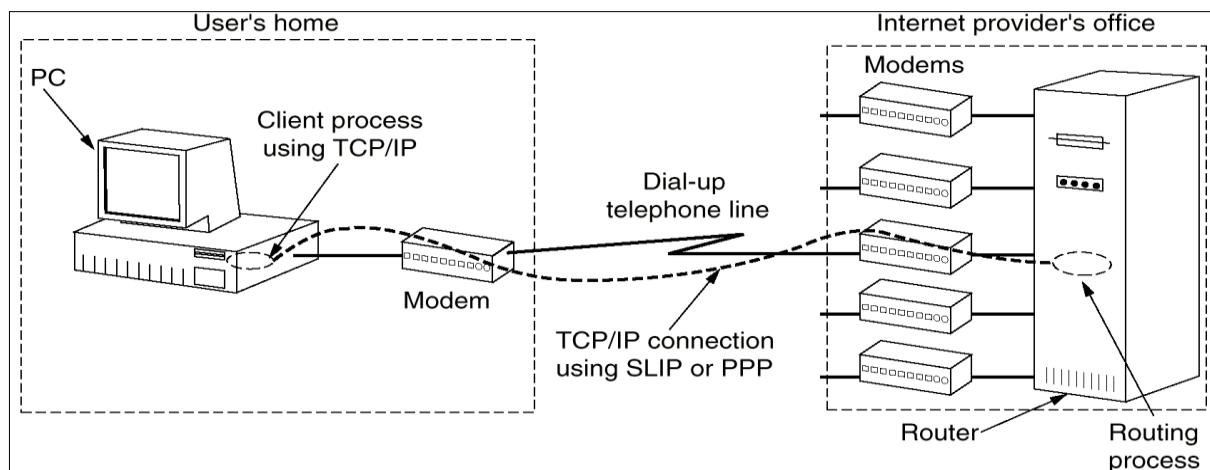
De link wordt stilgelegd door de **release** primitive en **DISC – UA** op LAPM.

Een volgend voorbeeld van HDLC is **LAPD**, het datalink protocol op kanaal D van een ISDN interface. Omdat hiervoor de praktische organisatie van ISDN interfaces moet gekend zijn verschuiven we deze bespreking naar dat hoofdstuk.

We geven wel nog 2 datalink protocols uit de internet-wereld, SLIP en PPP die mogelijk pas hun volledige inkleuring krijgen als de TCP/IP suite is besproken.

### 2.3.3 SLIP, SERIAL LINE INTERFACE PROTOCOL

Het internet bestaat uit verschillende machines, hosts en routers, verbonden met een communicatiestructuur. Binnenin één gebouw worden vaak LAN's gebruikt voor de verbinding, maar het grootste deel van de WAN bestaat uit gehuurde P2P lijnen. Ook de gebruiker die met zijn PC inbelt bij een ISP (Internet Service Provider) gebruikt een P2P verbinding.



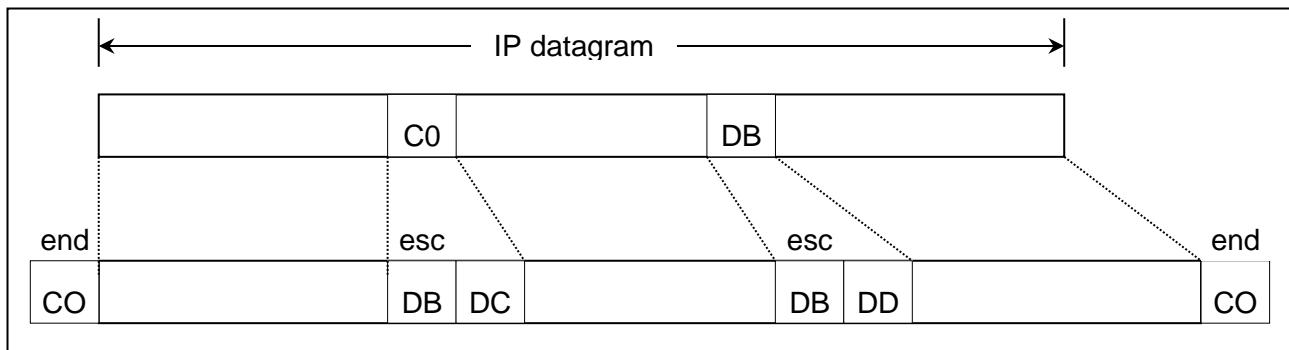
Figuur 204 Een P2P internet verbinding

De 2 meest gebruikte DLP's zijn SLIP, *Serial Line Internet Protocol* (verouderd), en PPP *Point to Point Protocol*.

**SLIP** situeert zich op de **datalink-laag** en is dus een alternatief voor ethernet vanuit het standpunt van IP. Het verpakt (**enkel!**) IP datagrammen om over RS232 verbindingen te sturen. Het is ontworpen in 1984 door SUN en wordt besproken in RFC 1055.

Afspraken :

- Het IP datagram wordt beëindigd met het 'END' karakter 0xC0. Meestal start men het datagram er ook mee om evt. reeds ontvangen bytes af te sluiten, dit foutief datagram zal dan worden verworpen.
- Transparantie wordt bekomen door databytes met waarde **0xC0** te vervangen door **0xDB:0xDC**. 0xDB noemt men het **SLIP ESC** karakter wat verschilt van het ASCII ESC karakter : 0x1B. Databytes = **0xDB** worden vervangen door **0xDB:0xDD**.



Figuur 205 Slip verpakking van een IP datagram

Nadelen aan SLIP :

- Er is geen protocol/type veld (zie verder bij PPP of ethernet) → de seriële lijn kan dus **enkel** gebruikt worden voor SLIP, geen ander protocol.
- Er is **geen checksum** (≈CRC bij ethernet), hogere lagen, bv. TCP, moeten de errordetectie alleen opknappen, en dat met toch wel redelijk hoge BER's op bv. modemlijnen.
- SLIP werkt alleen met IP
- Het is geen goedgekeurde standaard
- Er is geen MTU (Maximum Transfert Unit) gedefinieerd, SLIP ontvangt IP pakketten tot 1006 byte.

Ondanks deze nadelen wordt het nog, zij het sporadisch, gebruikt.

CSLIP, Compressed SLIP, is een recentere versie van SLIP, die de 40 byte TCP-IP header reduceert tot 3 à 5 byte en toch nog 16 TCP connecties ondersteunt. Deze optimalisaties worden in RFC 1144 beschreven.

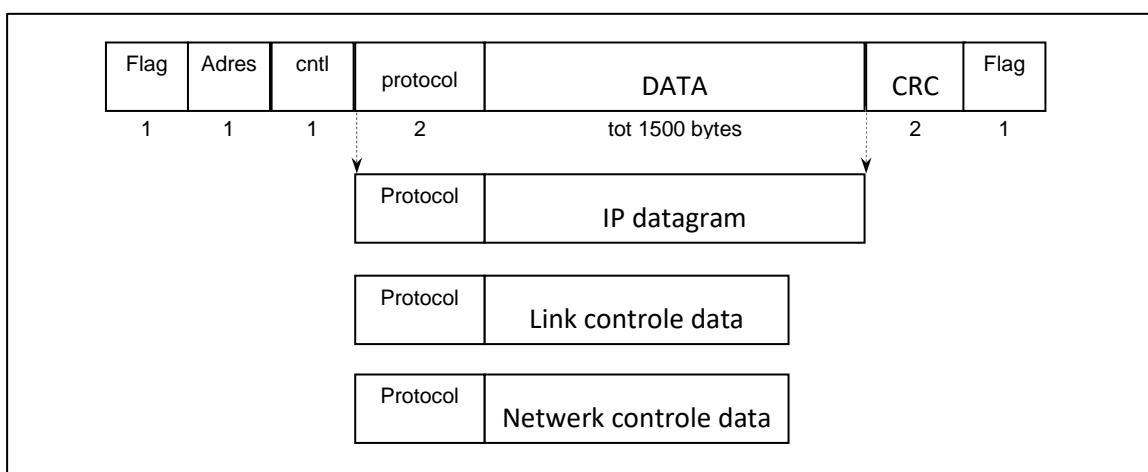
### 2.3.4 PPP, POINT TO POINT PROTOCOL

PPP is een moderne, verbeterde versie van SLIP die de onvolkomenheden van SLIP wegwerkt (RFC 1171, 1172, 1144, 1661 tot 1663).

Het bestaat uit 3 componenten :

1. IP datagrammen verpakken op een seriële link : meestal byte-georiënteerd asynchroon maar kan ook bit-georiënteerd synchroon.
2. Een **LCP**, Link Control Protocol dat datalink-connecties opzet, configureert en test.
3. Een familie van Netwerk Control Protocols (**NCP**) voor verschillende netwerklagen : IP, OSI, DECnet, appletalk.

Het frame-formaat van PPP is gebaseerd op de ISO HDLC standaard :



Figuur 206 Het PPP frame formaat.

Flag, adres en cntl zijn vaste waarden, resp.

- Flag = 7E (=0111 1110), de standaard HDLC vlag.

- adres = FF : het is een P2P link → geen adres nodig →  $\forall 1^L$  → altijd luisteren.
- Cntl = 03 : Het is een unnumbered frame : HDLC UI unnumbered information.

Het 16 bit **protocol** veld laat i.t.t. SLIP wel **verschillende netwerkprotocols** toe en is te vergelijken met het 'type' veld van ethernet (alleen niet dezelfde waarden : IP → 0x 0021 !!). Zie ook p. 201, Figuur 245.

Het geeft aan welk soort pakket in de 'payload' staat. (Vb. IPX : 0x 002B, OSI : 0x 0023, IPX control prot : 0x 802B)

De CRC is een 16 bit CRC, **softwarematig** berekend (ISO 3309).

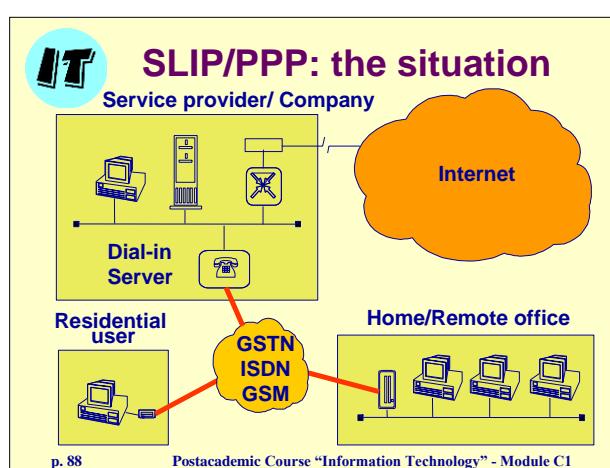
Transparantie van de start en endvlag 0x7E wordt op verschillende manieren opgelost :

- Synchrone link → bit-stuffing zie HDLC.
- Asynchrone link → byte stuffing met esc 0x7D + inversie van **bit5** van het karakter.

Vb.

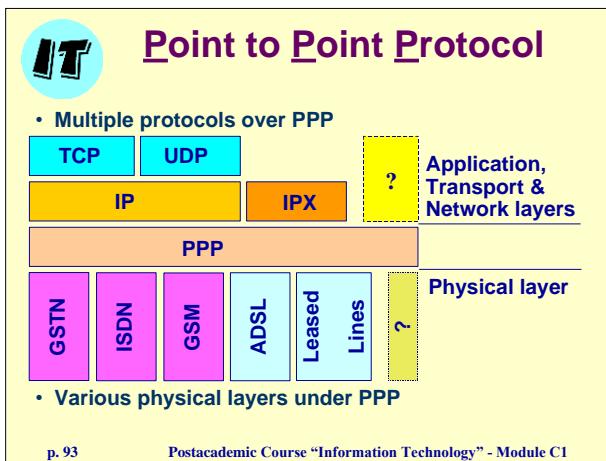
1. de byte 0x7E wordt verstuurd als 0x7D – 0x5E. (bit 5 → 0<sup>L</sup>)
2. de byte 0x7D wordt verstuurd als 0x7D – 0x5D.
3. elke byte onder 0x20 (→  $\forall$  ASCII controle karakters) worden ge'ESC'aped : vb. 0x01 wordt verstuurd als 0x7D – 0x21. (bit 5 wordt nu **aangezet**. Sommige modems interpreteren deze ASCII controle karakters, daarom worden ze vermeden in het datagedeelte.)

Er kan genegotieerd worden via LCP over het weglaten van het constante adres en controleveld én over het comprimeren van de IP en TCP header, 'Van Jacobson' compressie.



PPP is een **datalink** protocol dat niet alleen gebruikt wordt om in te bellen bij een ISP, Internet Service Provider, via het telefonienetwerk (PSTN=GSTN), maar dat ook netwerken, i.e. routers, verbindt d.m.v. een seriële lijn.

Figuur 207 Gebruik van PPP en SLIP.



PPP zorgt ervoor dat verschillende netwerklagen, IP, IPX, ... met verschillende fysieke lagen, GSTN, ISDN, ... kunnen worden verbonden.

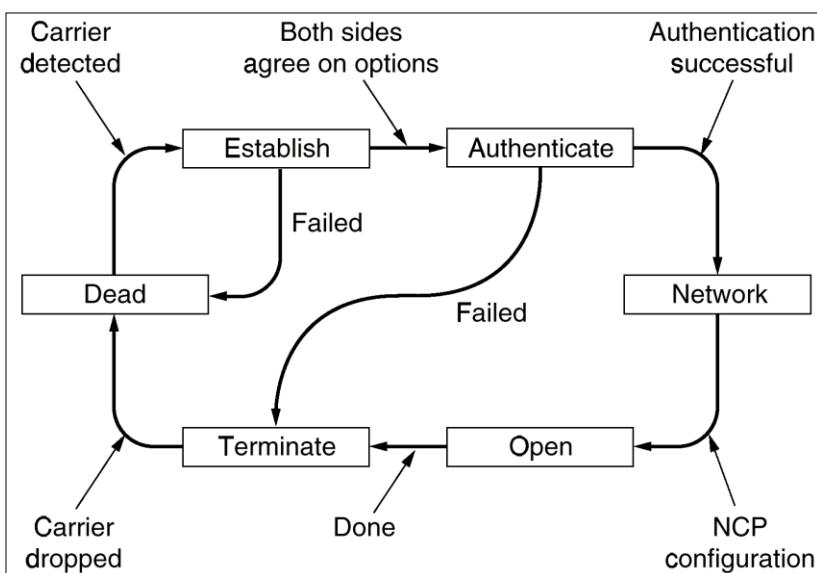
PPP is dus een verpakkingsmechanisme voor frames, komende van verschillende protocollen, te versturen via modems, bit-seriële HDLC lijnen, SDH en andere fysieke lagen.

Figuur 208 Situering van PPP in de TCP/IP suite

Hiervoor worden 2 **control protocols** voorzien die we kunnen situeren in een fase diagram :

1. **LCP**, Link Control Protocol dat **datalink**-connecties opzet, configureert en test.
2. **NCP** Netwerk Control Protocol dat voor elke netwerklaag (IP, IPX, OSI, DECnet, appletalk) de configuratie onderhandelt.

### PPP FASE DIAGRAM.



Dead : hier begint (en eindigt) elke link. Er is geen draaggolf, geen verbinding in de fysieke laag.

Komt er wel een fysieke verbinding, dan komt PPP in de 'establish' fase.

Establish : LCP gaat onderhandelen over opties door uitwisseling van 'configure-packets'. (alleen LCP pakket-ten worden geaccepteerd).

Figuur 209 Het PPP fase diagram.

Authenticate : hier moeten beide

partijen elkaar identiteit controleren. In geval van 'failure' wordt de link afgebroken.

Network : Wanneer deze toestand wordt bereikt wordt het passende NCP protocol aangeroepen om de specifieke netwerklaag te configureren. Elke netwerklaag moet dus apart geconfigureerd worden door de geschikte NCP. (bvb. ATCP (RFC 1378), IPCP (RFC1332) en IPXCP (RFC 1552)).

Als de configuratie slaagt wordt de toestand 'open' bereikt en kan het datatransport plaatsvinden.

Terminate : PPP kan elk ogenblik de lijn afsluiten door uitwisseling van LCP terminate pakketten.

### LCP PAKKETTEN :

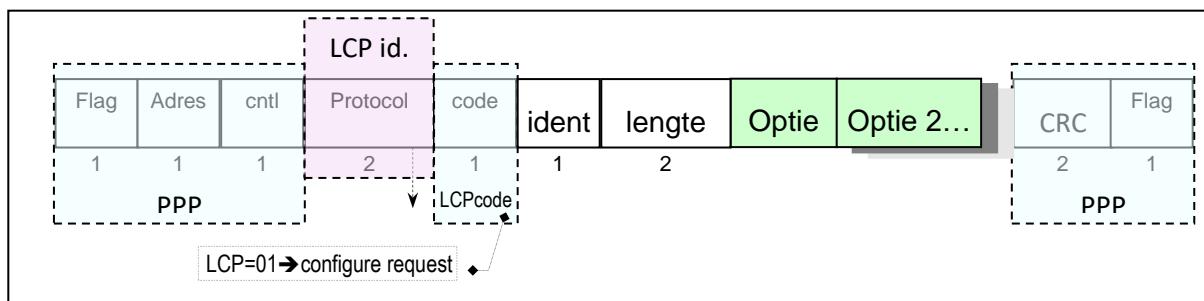
In RFC 1661 worden 11 types LCP pakketten gedefinieerd, allen krijgen een unieke 'code'

Tabel 4 : LCP codes voor PPP.

Code	Name	Direction	Description
1	Configure-request	I → R	List of proposed options and values
	Configure-ack	I ← R	All options are accepted
2	Configure-nak	I ← R	Some options are not accepted
	Configure-reject	I ← R	Some options are not negotiable
3	Terminate-request	I → R	Request to shut the line down
	Terminate-ack	I ← R	OK, line shut down
4	Code-reject	I ← R	Unknown request received
	Protocol-reject	I ← R	Unknown protocol requested
5	Echo-request	I → R	Please send this frame back
	Echo-reply	I ← R	Here is the frame back
6			
7	Discard-request	I → R	Just discard this frame (for testing)

- Met de 4 'configure'- typen kan de Initiator (I) waarden voor opties voorstellen, en kan de Reagerende partij (R) deze accepteren of afwijzen en een ander voorstel doen.
- De 'terminate' codes worden gebruikt om de link te beëindigen.
- Type 7 en 8 melden van R → I dat iets niet begrepen wordt. De 'echo'-typen dienen om de kwaliteit van de lijn te testen, en 'discard request' voor te debuggen.

Bvb. een configuration request



Figuur 210 Een PPP – LCP pakket voor configue request.

Het **ident**. veld moet bij elk pakket van I → R gewijzigd worden en wordt door R gekopieerd in het ident veld van het resp. 'reply' pakket. (bvb. code 2 voor configue ack → ∀ opties OK).

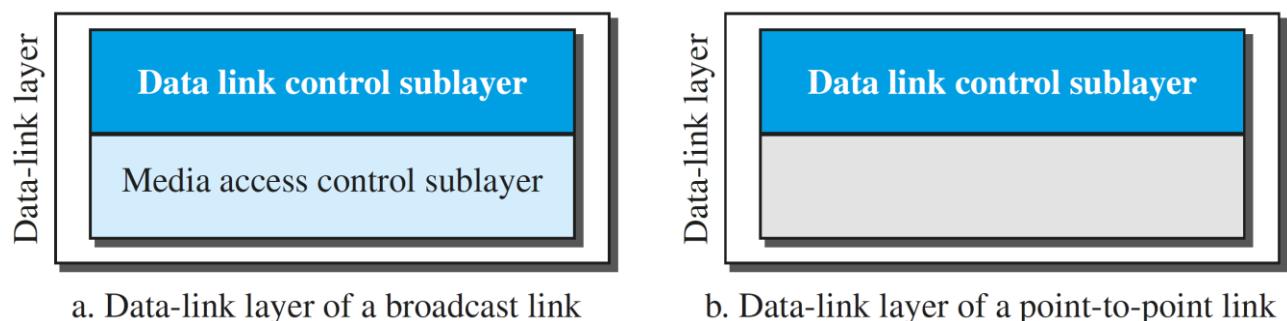
De opties (RFC11) zijn bvb. : 1=max. receive unit, 2=async.contr.char.map, 3=authentication prot., ...

### 2.3.5 DE DATALINK LAAG IN LAN'S → LLC

#### SITUERING LLC.

LAN's, zowel bus als ring, gebruiken LLC (Logical Link Control) als DLP. Beide types netwerken gebruiken een '**shared medium**' om frames over te zenden. I.t.t. multipoint-/drop netwerken is er geen master zodat een **verdelingsalgoritme** ervoor moet zorgen dat al de aangesloten stations, zowel werkstations als servers, aan hun trekken komen → **Medium Access Control (MAC)**.

Daarom is, in het geval van shared medium LAN's, de **datalink laag** opgesplitst in **2 deellagen** :



Figuur 211 Opsplitsing datalink laag in 2 sublagen voor LAN's

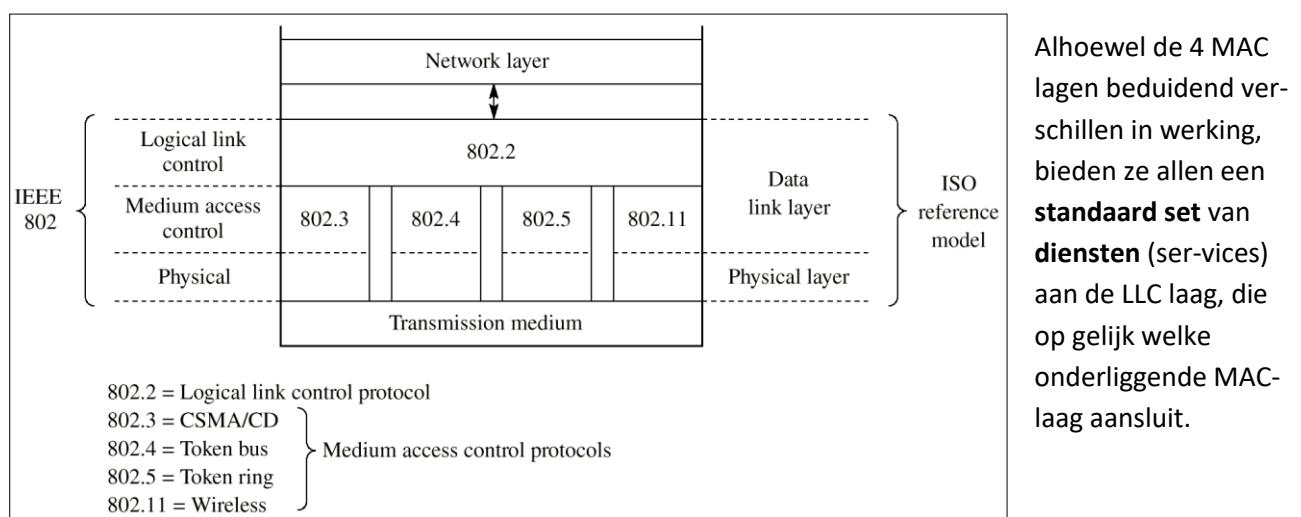
Met resp.

1. **LLC** : Logical link control, met functies gelijkaardig aan HDLC. bvb. IEEE802.2.
2. **MAC** : Medium Access Laag, die de toegang tot het medium regelt. bvb. IEEE802.3

Door zijn complexiteit wordt de MAC-laag besproken in een aparte paragraaf : pt. 2.4 op p. 181.

De fysische en datalink-standaarden zijn vastgelegd door IEEE in de 802 documenten (en overgenomen door ISO met toevoeging van een 8 : IEEE802.3 → ISO 8802.3 ...).

Bovenaan IEEE netwerken bevindt zich steeds de 802.2 LLC laag. Daaronder, afh. v. de fysieke bekabeling, de verschillende MAC lagen.



Figuur 212 Datalink structuur bij LAN's

Wat opvalt is dat de **fysische** laag (L1) bepalend is voor het type MAC-laag (L2!) dat gebruikt kan worden. (Dit is in tegenspraak tot wat het OSI model nastreeft : uitwisselbaarheid per laag en een sterke scheiding van de verantwoordelijkheden).

Deze **fysische** laag zelf kan dan weer, zoals u weet, ook verschillende vormen aannemen, bv. voor ethernet, IEEE 802.3: 10Base2 ... 10GBaseT ...

802.3 MAC layer CSMA/CD (ethernet)						
10Base2	10Base5	10BaseT	10BaseF	100BaseT4	100BaseTX	100BaseFX
Dunne coax	Dikke coax	Twisted pair	Fiber	4 TP paren	2 TP paren	fiber
185m max	500m max	100m		100m	100m	

Normaal wordt de **MAC**- en de **fysische** laag uitgevoerd in **firmware** in speciaal ontworpen IC's.

## LLC, LOGICAL LINK CONTROL – IEEE 802.2

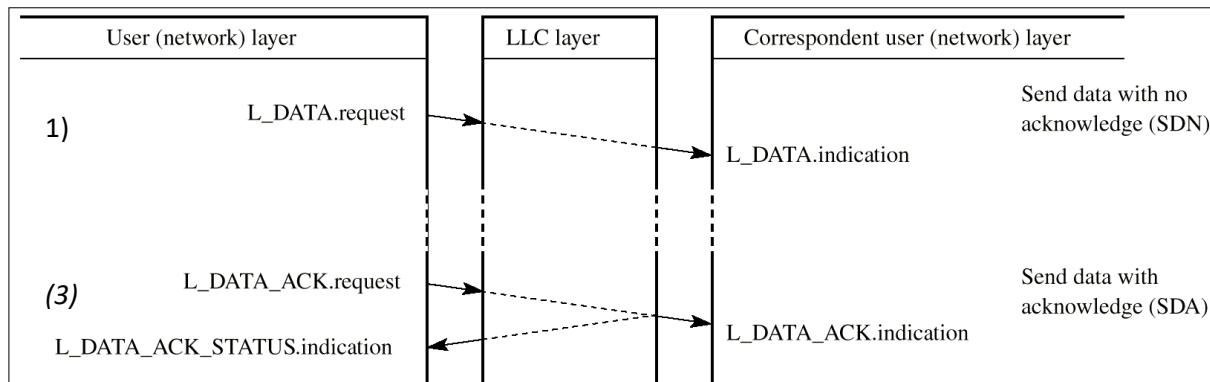
We kunnen stellen dat de LLC's, van de 2 communicerende DTE's, 'peer to peer' werken.

In principe zijn er 2 types services :

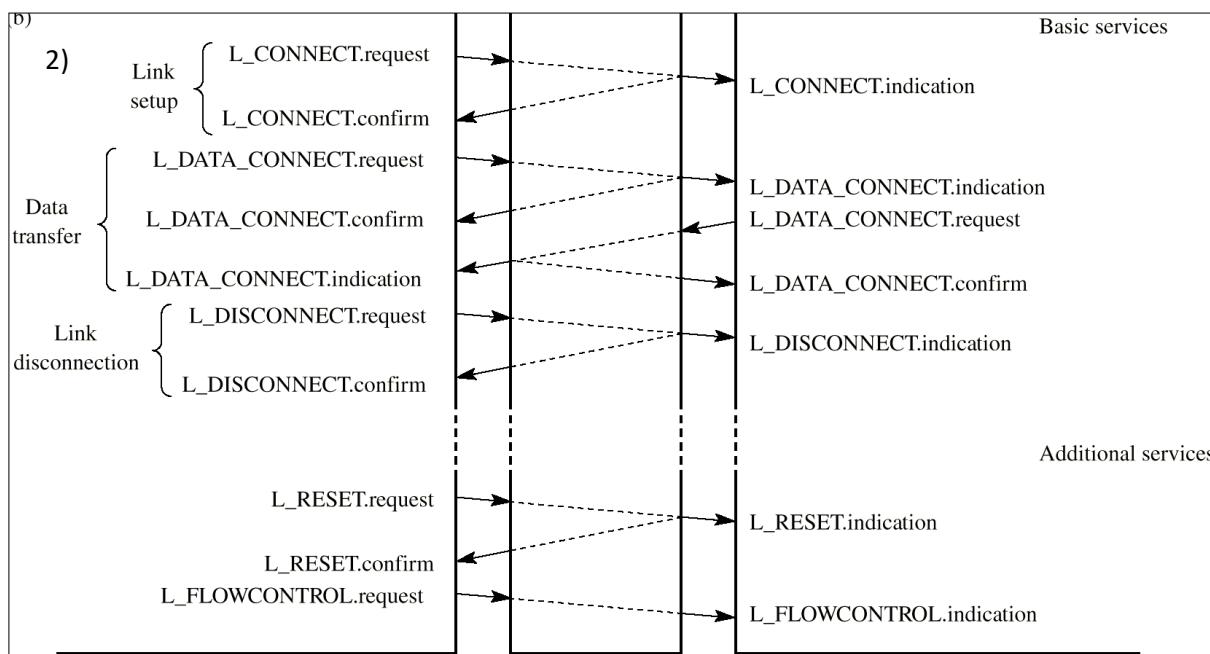
1. **CL-unack**, Connection-Less, **unacknowledged** service
2. **CO-ack**, Connection-Oriented, **acknowledged** service.

(3. In sommige automatiseringsmiddens is de **reactietijd** op een event van belang (... → CL) én moet er **betrouwbare** communicatie gebeuren → **acknowledged connectionless service**.)

1. **CL** : er is een minimale protocol overhead bij links met een lage BER → error/flow control en sequencing wordt uitgevoerd door een hoger protocol (typ. laag 4). Deze service wordt veruit het meeste gebruikt.



2. **CO** : anderzijds impliceert het opzetten en afbreken van de link + error/flow control en sequencing.



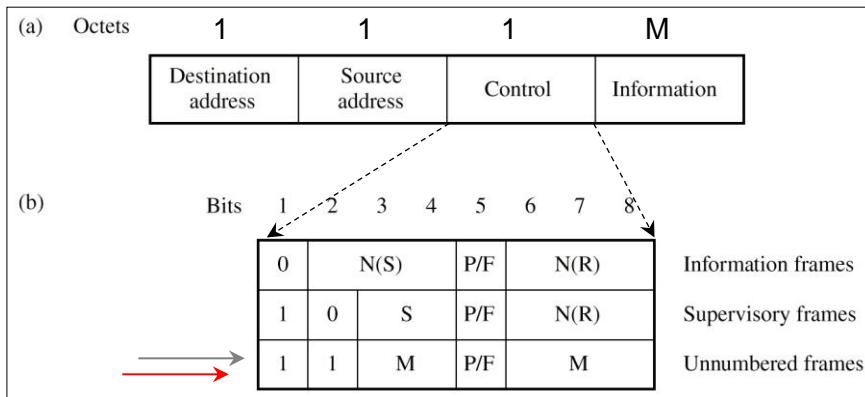
Figuur 213 LLC werking : 1) CL, unack en 3) ack, 2) CO

1. CL – ‘unack’ zal het LLC protocol ‘zo goed mogelijk’ de data versturen over de MAC laag zonder te weten of dit correct is aangekomen. (➔ hogere lagen)
2. CO – ‘ACK’ : hier moet de LLC eerst een logische connectie leggen (en later afbreken), en bovendien moeten alle (correcte) dataframes worden bevestigd.

**Reset** wordt gebruikt als de ontvanger het noorden kwijt is (bv. zijn vensterteller) en **flowcontrol** duidt de venster grootte aan, m.a.w. hoeveel bytes men wil reserveren als buffer voor deze connectie.

### PROTOCOL WERKING.

Aangezien de datalink laag nu gesplitst is in 2 deellagen zijn ook de verantwoordelijkheden gesplitst.  
Bijgevolg heeft elk LLC frame de volgende (voor een deel bekende ➔ HDLC) vorm :



Dest. en Source adressen zijn enkel de LLC **service access points**, het zijn niet de befaamde fysische- of MAC adressen die onder de verantwoordelijkheid van de MAC laag vallen.

Er is ook geen CRC : error controle ∈ MAC laag.

Figuur 214 LLC frame format (a) en controle veld (b).

Net zoals bij HDLC is het controle veld 1 byte lang. Het bepaalt het type frame, en indien nodig, zend en ontvangstellers voor error/flow controle. Meestal worden bij ethernet ‘unnumbered frames’ in een type 1 service gebruikt ➔ UI = ‘Unnumbered Information’..

Voor **type 2** services (CO-ack) is de werking praktisch identiek aan **HDLC**, behalve dat framing- en error-detectie overgelaten wordt aan de MAC laag. ➔ frames : I, RR, RNR, REJ, SABME, DISC,...

Veel meer gebruikt is de **type 1** service die gebruik maakt van UI-, XID- en TEST-frames. UI-frames worden gebruikt voor datatransport naar de bestemming(en ➔ multi- of broadcast!). Aangezien er geen ack’s noch ‘sequence control’ is bij type 1 bevat het UI-frame geen tellers ➔ control = 0x 03.

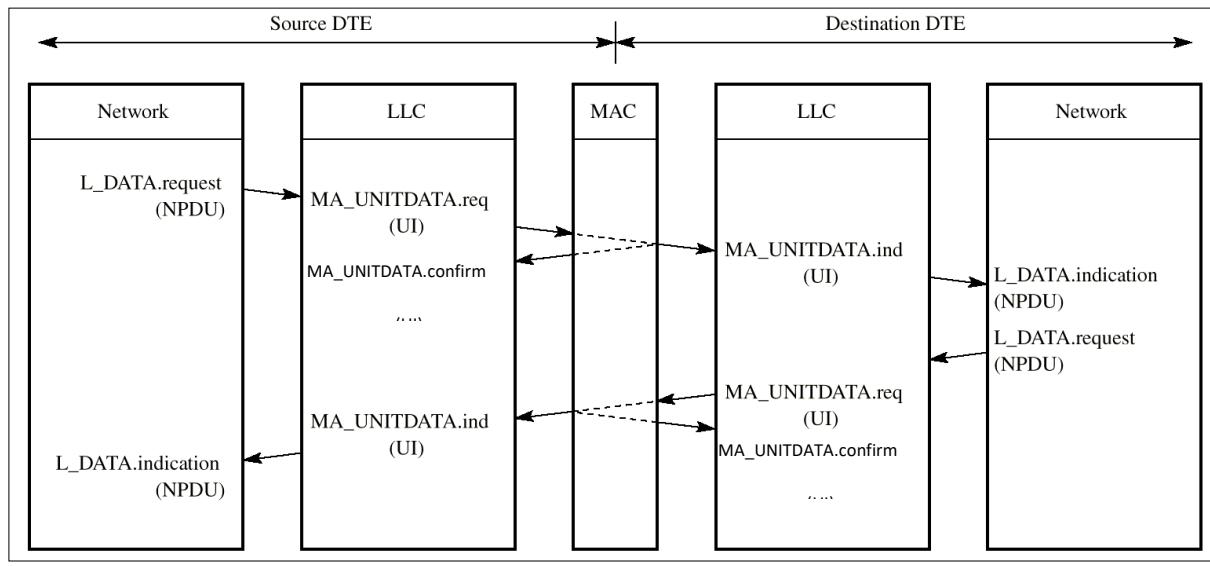
XID en TEST frames zijn optioneel, maar LLC moet wel hierop antwoorden. XID met een groep adres bepaalt de deelname aan deze groep : elk lid antwoordt door een XID response frame. Een ander gebruik van XID door een (nieuwe) LLC is om zijn aanwezigheid op het medium te broadcasten.

Het TEST commando voorziet een ‘loopback’ functie tussen elke LLC verbinding.

Onafhankelijk van de verschillende onderliggende MAC lagen (zie Figuur 212) wordt er een **standaard set** van **MAC-services** gedefinieerd om gebruikt te worden door de LLC :

- MA\_UNITDATA.request
  - MA\_UNITDATA.indication
  - MA\_UNITDATA.confirmation

De totale sequentie voor type 1 wordt dan :

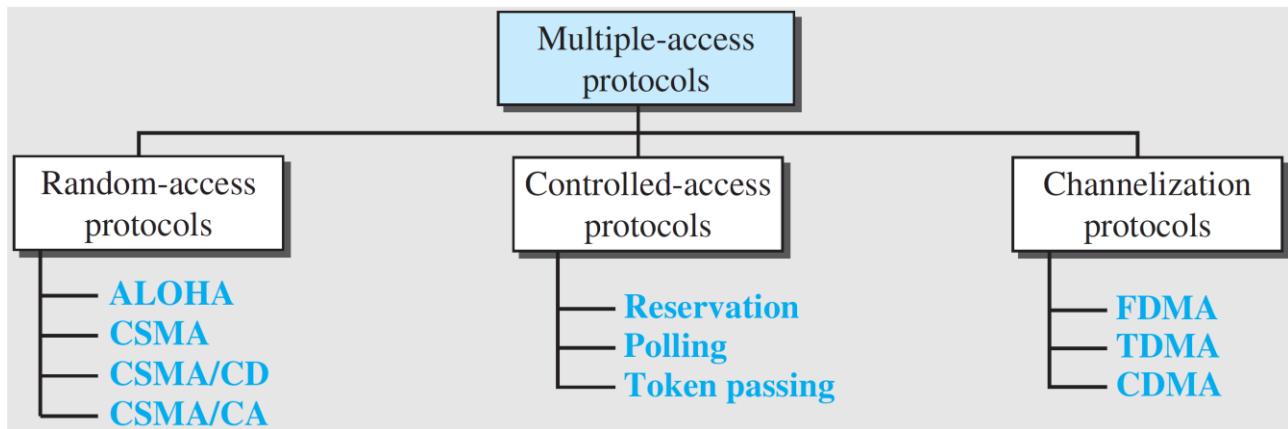


### **Figuur 215 LLC werking : overzicht.**

## 2.4 DE MEDIUM ACCESS CONTROLE DEELLAAG.

Op LAN's die door meerdere deelnemers tegelijk worden gebruikt (→ 'shared media' LAN, ook wel broadcast netwerken genoemd) moeten afspraken gemaakt worden om het wedijveren, = contentie, op de kabel te voorkomen. Er mag niet door elkaar ge 'praat' worden, men mag het medium niet monopoliseren, mekaar onderbreken...

Al deze problemen worden opgelost door een **deellaag** van de datalink-laag (L2): de **MAC-laag**. We delen ze in in 3 groepen met een verschillende benadering:



Figuur 216. Indeling van de MAC principes.

Er zijn 3 categorien :

- **Random-access.** Veel gebruikt in LAN en WAN **omgevingen**.
 

De meeste er van bespreken we hieronder. Kort Aloha en CSMA,

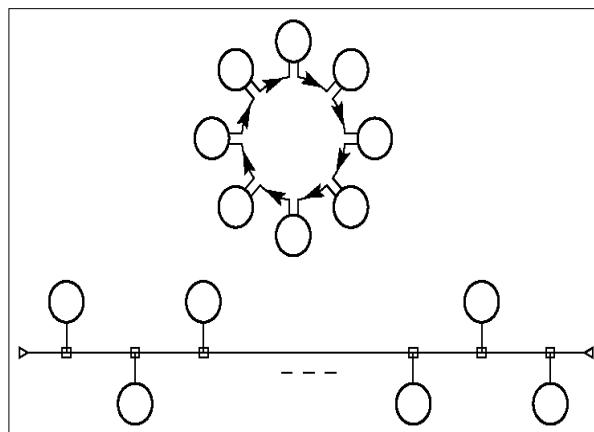
  - **CSMA/CD** voor **bus-topologien** (IEEE 802.3) en
  - **(CSMA/CA)** bij wireless netwerken IEEE 802.11 .)
- **Controlled-access.** Deze protocollen zijn complexer en worden soms ook gebruikt in LAN omgevingen. We bespreken hier:
  - **token passing** voor **ring-** of **bus-topologien**. (IEEE802.5 resp 802.4)
- **Channelised access.** Deze protocollen komen vooral in **wireless** technologie voor en vallen daarom buiten het bestek van deze cursus.

We zien dat fysieke- en datalink-laag **niet volledig los** van elkaar kunnen worden ontworpen. Elke specifieke fysieke laag heeft zijn eigen noden en ontwerp-oplossingen... , i.t.t. het OSI voornemen!!!.

### 2.4.1 FYSIEKE TOPOLOGIEËN.

LAN's hebben/hadden bij voorkeur 2 mogelijke topologien : **bus** en **ring**. ➔ Later vooral **STER-topologie**.

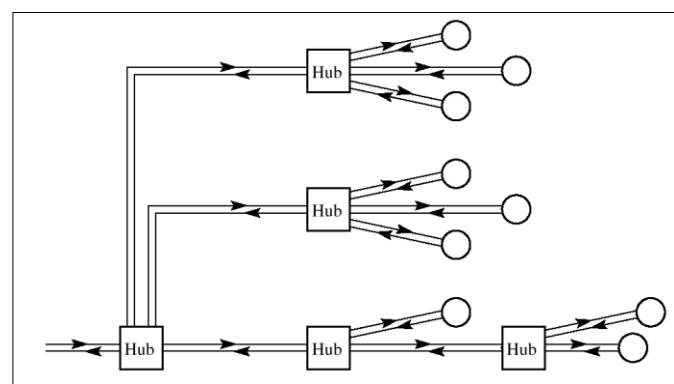
Bij een **ring topologie** zal er een netwerkabel van 1 DTE naar een andere vertrekken totdat de ringvorm is bereikt. Elke DTE is direct verbonden met zijn buur als een **P2P** verbinding die **unidirectioneel** is. De MAC algoritmes moeten de ring delen tussen alle gebruikers.



Een **bus-topologie** kan aanzien worden als **1 kabel** waarvan elke DTE 'aftakt'. De MAC-circuit's en -algoritmes moeten bijgevolg de bandbreedte van het medium delen. Er staat op elk moment slechts 1 frame van 1 zender op de bus. Wie die zender is bepaalt het CSMA/CD mechanisme.

Figuur 217 Bus en Ring topologie van LAN's.

Een variatie op bus en ring is de **hub/tree topologie** (... zie CAT5 bekabeling) die vrij snel volgde op de **bus**~ in bv. Ethernet. ... (ook in token ring). Hubs kunnen hiërarchisch worden geschakeld om een tree te vormen ➔ **'hub/tree'**. Dit is fysiek een **ster-topologie**.



Maar ... ook al is de **fysische** verbinding een **ster-topologie**, de **logische** verbinding is ofwel **bus** of **ring**. De hub is hier enkel de bus of ring, **gesublimeerd** in 1 toestel.

In een bus topologie is de hub gewoon een **repeater** die de signalen bit per bit vermenigvuldigt en verder stuurt... wat dus neerkomt op een bus: op elk moment is er slechts 1 frame van 1 zender op de tree.

Figuur 218 Fysische hub/tree topologie van LAN's.

(Voor een ring moet de hub weten wat de logische layout van de ring is, m.a.w. welke buren elke DTE heeft. De frames worden dan **alleen** doorgeschakeld naar de 'volgende' buur.)

#### STER-topologie.

Vervang in bovenstaande figuur de hubs (=L1) door **switches** (L2) en je hebt een **modern geswitched ster netwerk**. Daarbij is elke verbinding een segment op zichzelf. Er kunnen dus tegelijk veel meer frames op de ster staan aangezien switches een **geheugen** hebben waarin ze de frames kunnen bewaren. Hubs hebben dit niet!.

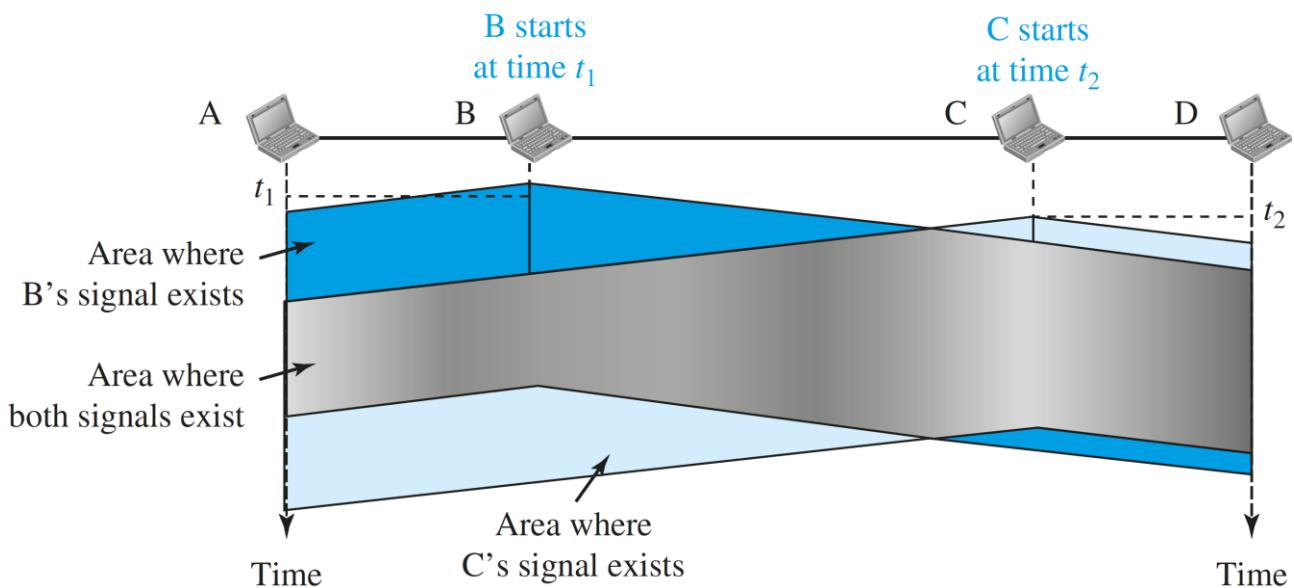
## 2.4.2 CSMA/CD MEDIUM ACCESS CONTROL VOOR EEN BUS

### HISTORIEK

- **Aloha** : geen MAC, is er iets te zenden, zend het dan. Geen overleg. Weinig efficient.
- **Slotted Aloha** : alleen toegelaten te zenden op vaste tijdstippen → TDMA.
- **Collision avoidance** : via een reserveringskanaal wordt een tijdsinterval ‘besproken’. → satellietcommunicatie en mobiele telefonie.
- **Collision detection (en carrier sensing → CSMA/CD)** : elk station dat wenst te zenden zal eerst luisteren op het kanaal en enkel zenden als er geen verkeer is. (carrier sense) Als hij begint te zenden zal hij ook tegelijk controleren of de eigen boodschap niet wordt verstord door een zender die op hetzelfde moment is begonnen zenden. (collision detect).

### WERKING CSMA/CD.

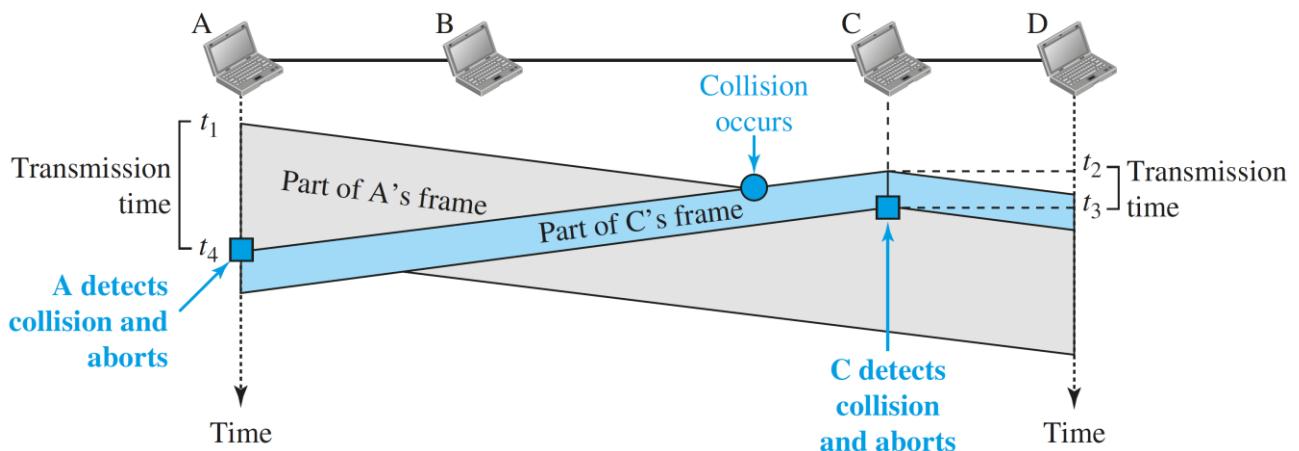
Wanneer 2 DTE's ± tegelijk (op  $t_1 - t_2$ ) beslissen om een frame te versturen (aangezien ze geen verkeer op de bus merken), zal het signaal worden verstord en treedt een **collision** op:



Figuur 219 Tijdsverloop (theoretisch) van een collision.

Beide zenders, B en C, zien op het moment dat ze hun frame starten geen verkeer op het shared medium. Hun frame verplaatst zich op het medium in beide richtingen (en wordt geabsorbeerd op de uiteinden door afsuitweerstanden). Onderweg vindt de collision plaats. DAT moet wel opgemerkt worden door beiden!

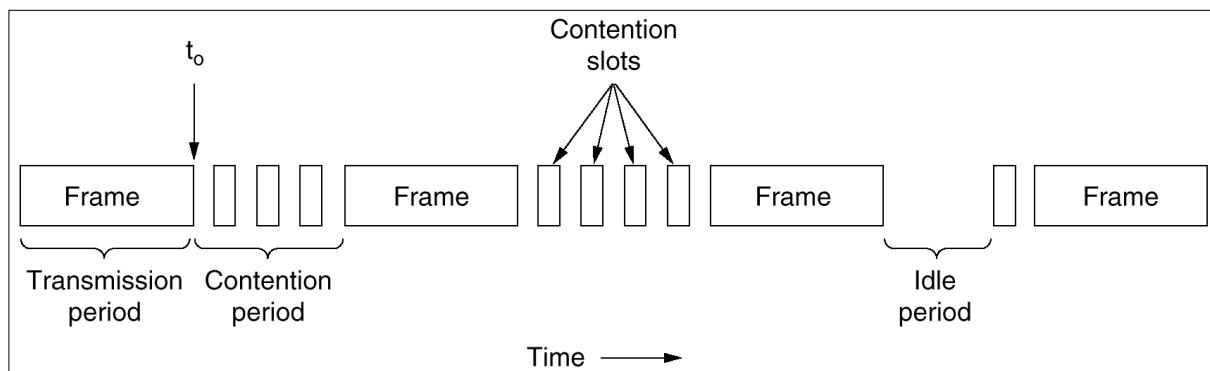
Daarom controleren de DTE's hun eigen boodschap en zullen ze na een tijdje de collision detecteren (→CD). Een frame dat onderweg een collision heeft ervaren is niet meer leesbaar door de stations op het medium, en, i.t.t. voorgaande Figuur 219, zal een DTE zijn transmissie **stopzetten** zodra hij een collision opmerkt, Figuur 220 Praktisch verloop van een collision..



Figuur 220 Praktisch verloop van een botsing.

Om ervoor te zorgen dat de andere DTE's, en zeker diegene die mee de botsing veroorzaakt, dit merken **stopt** de zender ogenblikkelijk met het zenden van zijn frame en plaatst hij een '**jam sequence**' op de kabel.

De 2 DTE's die in de botsing waren betrokken wachten vervolgens een 'random' tijd alvorens terug te beginnen zenden. (Tenzij natuurlijk een van de andere stations ondertussen reeds aan het zenden is.)



Figuur 221 CSMA/CD kent 3 toestanden : transmissie, contentie en rust.

Het transmissiekanaal kan zich dus in 3 toestanden bevinden :

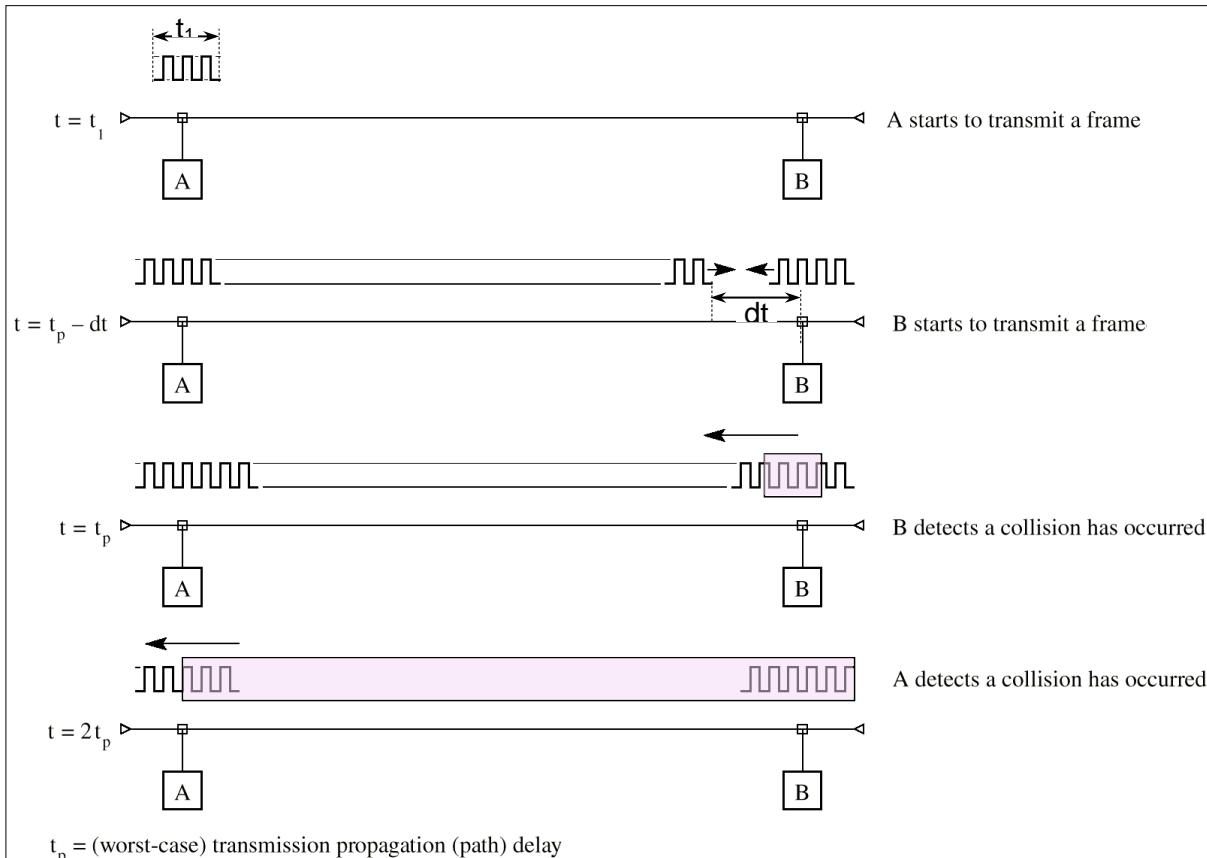
- transmissie,
- contentie en
- rust (als er niets te zenden valt).

Collisions moeten ten allen tijde worden opgemerkt (door alle stations). Daarom heeft een (CSMA/CD-) frame een minimum lengte: het frame mag niet 'voorbij' zijn voor de zender vóór dat alle mogelijke botsingen zijn opgemerkt.

De minimale tijd die nodig is om een botsing te ontdekken is de tijd die het signaal nodig heeft om zich van het ene naar het andere station voort te planten en terug. We bekijken dit nauwkeuriger.

## MINIMUM FRAME LENGTE VOOR EEN CSMA/CD FRAME

Stel A ziet geen carrier en begint te zenden op tijdstip 0. Dit signaal heeft een tijd  $t_p$  nodig om in station B te geraken, het station het verst verwijderd, dus de (worst case) propagatietijd  $t_p$ .



Figuur 222 CSMA/CD collisions.

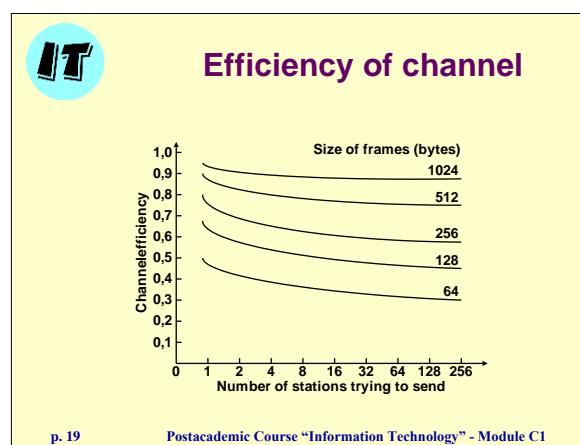
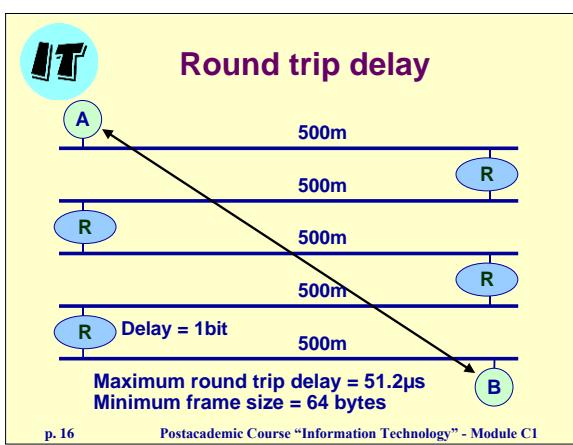
Op moment  $t_p - dt$  ziet B nog geen carrier en begint ook te zenden. Beide golven botsen en de collision wordt in B opgemerkt op tijdstip  $t_p$ . B stopt het zenden en eindigt met een korte JAM sequentie. (= 32 bits, om de botsing duidelijk te maken voor iedereen).

Aangezien A op de hoogte moet zijn dat er een collision is opgetreden moet hij zeker blijven zenden tot het vermeende signaal van B hem zou bereikt hebben → de collision moet ook A bereiken, en doet dit, worst case, op tijdstip  $2 \times t_p$ . (-dt maar  $dt \rightarrow 0$ ).

Botsingen moeten **ALTIJD** kunnen worden gedetecteerd. Daarom is er een **minimum lengte** voorzien voor een frame. Zoniet zou bv. A, die het laatst de botsing opmerkt, reeds kunnen **gestopt** zijn met zenden tegen de tijd dat het frame (= de collision) van B bij hem aankomt. B zal wel ‘jammen’ maar dat kan even goed over een andere collision gaan → A denkt **verkeerd** dat zijn boodschap is verstuurd!

⇒ **min. framelengte** is afh. v.  $t_p$ , m.a.w. van de **max. lengte van het medium** ! .

Nu is de max. fysieke afstand van een ethernet-segment te vinden in de eerst gedefinieerde standaard : **10base5**, of dikke coax met een lengte van **500m**. Bovendien kunnen er zo **5** segmenten na elkaar geplaatst worden, gescheiden door een repeater (L1 → bit per bit) .



Figuur 223 a) Round trip delay op ethernet b) channel efficiency op ethernet.

Totale afstand =  $5 \times 500m = 2500m$

Er zit ook een kleine vertraging op de repeaters ( $\approx 1\dots$ bit) → men rekent met **3000m**.

Voor de propagatie van een EM golf op koper rekent men met een snelheid =  $2/3 \times C$ ,  $= 2/3 \times 3 \cdot 10^8$ .

Dit betekent dat voor 1 trip  $t_p = \frac{3000}{\frac{2}{3} \times 3 \cdot 10^8} = 15 \mu s$  !

Een **roundtrip**  $2t_p$  is dan = **30 μs**.

Men telt hierbij een veiligheidstolerantie en neemt als 'slottijd' : = **51,2 μs**.

- Een **geldig frame** moet minstens **langer** zijn dan de **slottijd**.
- Een **collision** moet alle stations bereikt hebben in een tijd **kleiner** dan de slottijd!.

Voor 10base5, 10Mbps, een slottijd van **51,2 μs** = **512** bits of **64 bytes**. Dit leidt dan tot:

→ De min. frame size bij ethernet = **64 bytes**, = 512 bits tegen 10Mbps → **51,2 μs**.

Een voorwaarde die tot op heden nog altijd (minimaal) van kracht is!

De slottijd is tegelijk de **minimum tijd** dat een zendende DTE moet wachten om er zeker van te zijn dat geen collision is opgetreden. Na de slottijd kan er dus geen collision meer plaatsvinden omdat dan het medium ondubbelzinnig in gebruik is door 1 DTE!

Aangezien herhaalde collisions op een druk medium wijzen zal retransmissie steeds langer worden uitgesteld. Dit proces heet '**truncated binary exponential backoff**'.

Het aantal slottijden dat gewacht wordt voor de ' $N$ 'e retransmissie is een 'random' geheel getal **R**, gelegen tussen  $0 < R < 2^K$  met  $K = \text{MIN}(N, \text{backoff limit})$ . ( max =  $2^{10}$  ). Slaagt een station er niet in om na 16 pogingen (16 = attempt limit) met een random wachtijd van max 1024 slottijden een frame te verzenden dan zal de MAC functie een melding geven : "*excessive collision error*" en er mee ophouden.

Op te merken valt nog dat collision's frequenter worden naarmate het **verkeer** op de bus toeneemt, alsook als de **lengte** van de bus toeneemt door de **propagatietijd** over een lange lijn (B heeft meer kans om 'op

*hetzelfde moment*' als A te beginnen zenden). M.a.w. het **rendement zakt** naarmate het **gebruik** (drukte én aantal station's, dus lengte) **toeneemt** : de werkelijke capaciteit ligt tussen 40 en 80% van het maximum.

Het aantal collisions kan worden gereduceerd door de frame-lengte te vergroten. Het maximum wordt hier bereikt bij +/-1024 bytes, zie vorige Figuur 223 a) Round trip delay op ethernet b) channel efficiency op ethernet.

→ **max.frame size** wordt vastgelegd op **1518 bytes**,

Dit heeft ook belang bij de keuze: **ethernet vs IEEE802.3** , zie ook verder op **p.201**.

Resume CSMA/CD bij standaard ethernet:

Bit rate	10 Mbps (Manchester encoded)
Slot time	512 bit times
Interframe gap	9.6 us
Attempt limit	16
Backoff limit	10
Jam size	32 bits
Maximum frame size	1518 octets
Minimum frame size	512 bits (= 64 byte)

## 2.5 ETHERNET - IEEE 802.3

IEEE heeft als grote verdienste dit CSMA/CD-principe op de MAC-layer te standaardiseren. Zo zijn ethernetverbindingen gestandaardiseerd als **IEEE 802.3**.

Deze **MAC** laag is aansluitbaar op verschillende **fysische** layers, ook genormeerd door IEEE 10(0)(0)BaseXX:

802.3 MAC layer CSMA/CD						
10Base2	10Base5	10BaseT	10BaseF	100BaseT4	100BaseTX	100BaseFX
Dunne coax	Dikke coax	Twisted pair	MMFiber	4 TP paren	2 TP paren	fiber
Manchester	Manchester	Manchester	Manchester/ on-off	8B6T	4B/5B MLT3	4B/5B NRZI
185m max	500m max	100m	500m	100m	100m	100m

10Base5 staat voor 10Mbps, basisband transmissie, 500 m lengte kabel. (10base2 → +/- 200m)

### 2.5.1 STANDAARD ETHERNET 10BASEXX

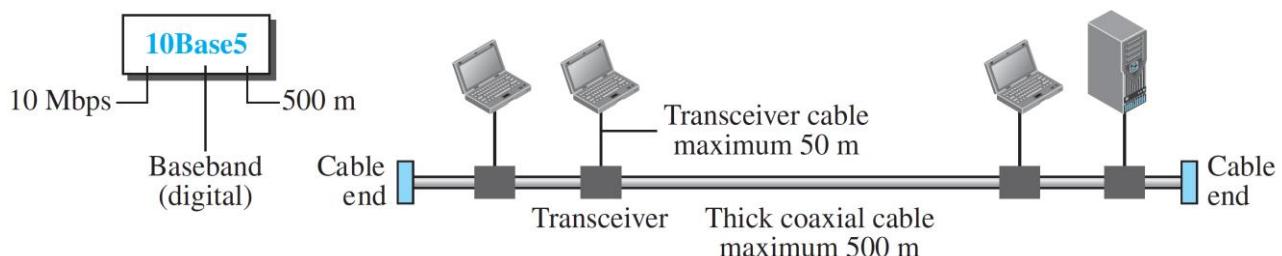
Slechts 4 uitvoeringen van Ethernet werden algemeen erkend:

Implementation	Medium	Medium Length	Encoding
10Base5	Thick coax	500 m	Manchester
10Base2	Thin coax	185 m	Manchester
10Base-T	2 UTP	100 m	Manchester
10Base-F	2 Fiber	2000 m	Manchester

10Base: = 10Mbps, allen basisband encoding: manchester code. De bits worden niet gemoduleerd, maar rechtstreeks op de kabel/fiber geplaatst.

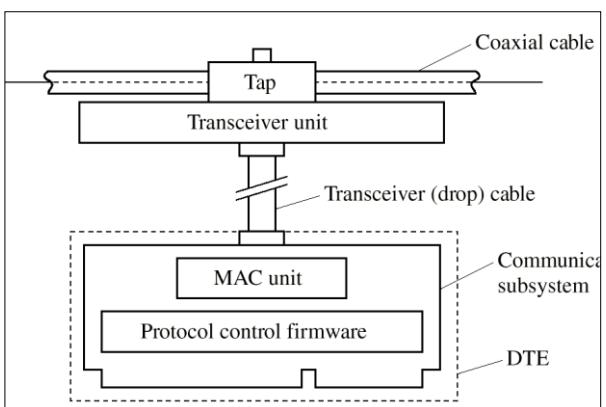
#### 10BASE 5

De eerste uitvoering is op dikke coax (duimdikte, niet buigbaar) waarbij de DTE's via een transceiver werden aangesloten. De 'BUS' of coax is maximaal 500m lang. (met repeaters, 4x max, uitbreidbaar).



Figuur 224. 10BASE5 dikke coax

Om reflecties op de uiteinden te vermijden moet de kabel afgesloten worden op zijn karakteristieke impedantie, **50Ω**.

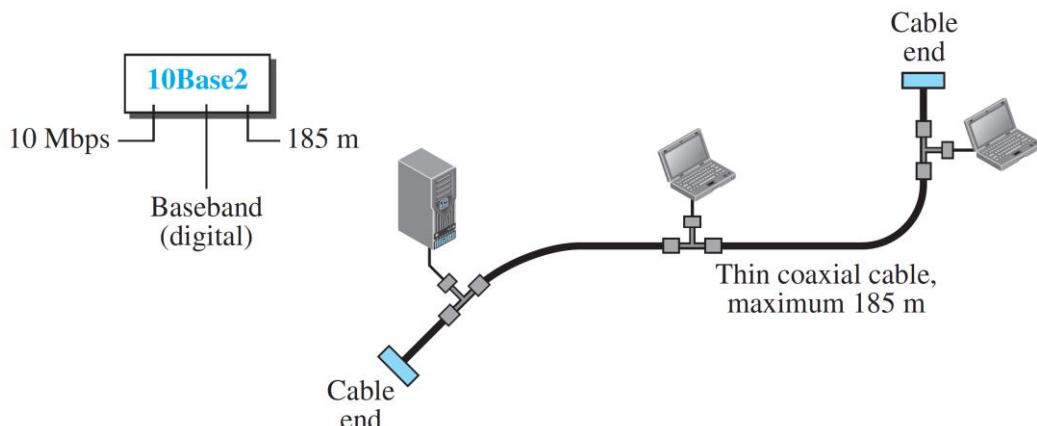


Het verschil tussen dikke en dunne coax is de **plaats** van de **transceiver**. Bij 10base5 is deze op de kabel geprikt waarbij met een drop cable de DTE wordt bereikt.

Voor 10base2 is transceiver en drop cable geïntegreerd op de **interfacekaart** in de DTE, zie verder.

## 10BASE 2

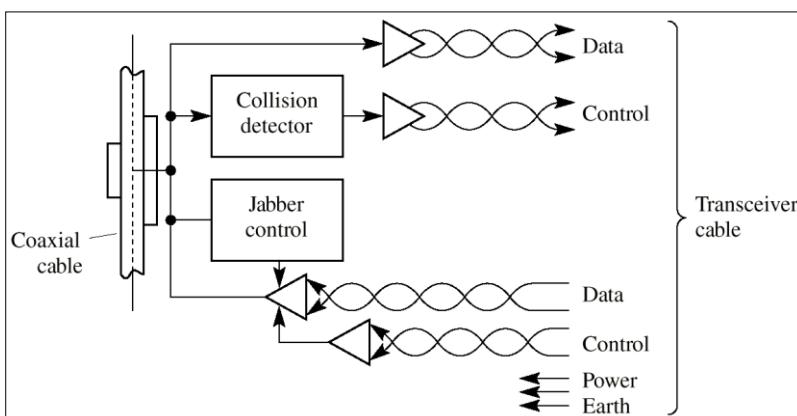
Ook wel ‘thin ethernet’ genoemd, gebruikt soepele coax kabel die veel dunner is, maar daarom ook beperkt is in segment lengte: max 185m (groteverzwakking van het signaal in de minder ‘perfecte’ coax kabel). De kabel kan ‘langs’ de DTE’s gelegd worden waardoor de drop kabel verdwijnt in het T stuk. De receiver zit nu op de NIC (Network Interface Card).



Figuur 225. 10BASE2 dunne coax.

Ook hier wordt de kabel afgesloten met **50Ω**.

## DE FUNCTIES VAN DE TRANSCEIVER:



- Data zenden en ontvangen
- Collisions op de kabel detecteren.
- De kabel beschermen tegen fouten van de transceiver of DTE : **jabber control**.
- elektrische isolatie tussen kabel en DTE.

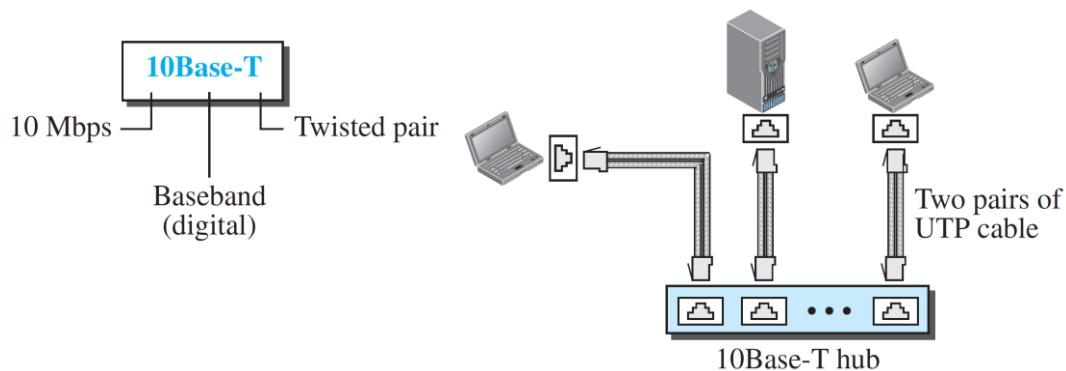
Figuur 226 Het schema van een 10base5 transceiver

(Jabber control isoleert de kabel van de transceiver als deze laatste verstoord is : men kan niet toelaten dat 1 NIC het hele netwerk-segment bezighoudt met wartaal. Als bepaalde tijdsintervallen, zoals max. frame lengtes zijn overschreden zal de jabber-control elektronisch ingrijpen zodat de rest van het netwerk verder kan werken.)

In **10base5** is de transceiver **T** dus verbonden via een STP kabel met **5 paren** : 1 paar voor de voeding van DTE → **T**, 2 voor de data FDuplex, 1 voor collision-detectie **T** → DTE en 1 voor jabber controle.

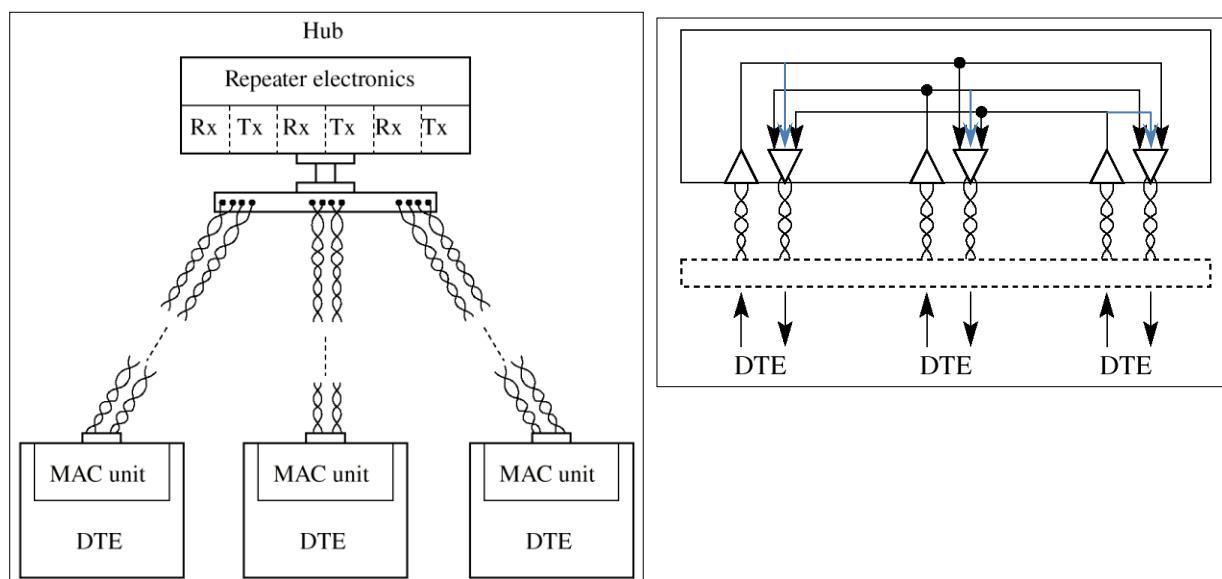
### 10BASE T

Voor TP bekabeling, **10baseT**, heeft de (onvermijdelijke) **hub** alleen de taak de elektrische signalen te **versterken**. Per kanaal zijn er **2 TP** verbonden aan de hub : 1 zend- en 1 ontvangstpaar. De maximale lengte van DTE tot de hub is nu 'slechts' 100m.



Figuur 227. 10BASET Twisted Pair.

De hub zal data, ontvangen op **gelijk welk Rx-paar**, versterken en verzenden op **alle Tx-paren**, wat ertop neerkomt dat alle DTE's in 1 'collision domain' zitten. De hub is het segment, gesublimeerd in 1 toestel.

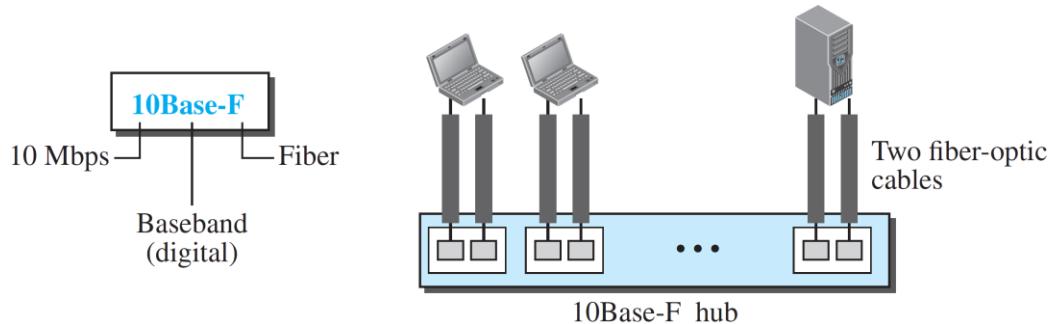


Figuur 228 Hub configuratie topologie en inwendig schema

Het probleem hierbij is de NEXT : alle sterke Tx-signalen beïnvloeden de (ver)zwakt(k)e Rx signalen. Hiervoor worden **adaptieve crosstalk cancellation circuits** gebruikt.

## 10BASE F

De fiber uitvoering van ethernet kan langere afstanden overbruggen 2000m. Het is ook een ster uitvoering naar de fiber hub.



Figuur 229. 10BASEF Fiber

Later is ook fast-ethernet, 100Mbps, 100baseT of F ontwikkeld op basis van CSMA/CD.

Onafhankelijk van de fysische laag bevat een NIC, 'Netwerk Interface Card', altijd :

- Een MAC eenheid, verantwoordelijk voor in- en uitpakken van frames, verzending over het medium, error-detection, en implementatie van het MAC algoritme.
- Een dual-port RAM dat de MAC eenheid toelaat frames te ontvangen en te verzenden tegen de hoge snelheid op het medium, en tegelijk de host-computer toelaat de informatie te lezen en schrijven.

## 2.5.2 IEEE 802.3 FRAME FORMAAT.

L1:

Preamble	Na een <b>preamble</b> van 7 bytes (=10101010) manchester code moeten alle MAC's op het medium in <b>bitsynchronisatie</b> zijn.
SFD	1 octets Daarna komt een SFD, Start of Frame Delimiter, (10101011) die de start van een frame aankondigt → <b>byte-</b> en <b>frame-synchronisatie</b>
Destination address	6 octets
Source address	6 octets
Length indicator	2 octets
Data	≤ 1500
Pad (optional)	
Frame check sequence	4 octets

**Figuur 230 IEEE 802.3 frame formaat.**

L2:

- **Address (S&D)**

Zoals reeds eerder besproken behoorden **adressering** op het gedeeld medium en **error-controle** tot de verantwoordelijkheden van de **MAC deellaag**.

De destination en source adressen zijn **6 bytes** lang en worden **MAC adressen** genoemd. Ze zitten **ingebakken** op de NIC (Netwerk Interface Card) en zijn **uniek**, toegekend door IEEE aan de fabrikanten.

Een MAC-adres heeft de volgende vorm :



- I/G = Individual / Groep adres
- U/L = Universal / Local toegekend adres

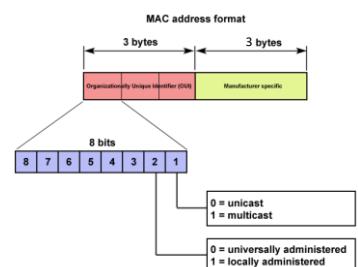
Het eerste bit (I/G) van het (dest.) adres specificeert of het om een **individueel** adres gaat dan wel om een **groepsadres**. Als alle bits van het dest. adres = 1<sup>L</sup>, (= 0x FF:FF:FF:FF:FF:FF), dan is het een **broadcast**-adres.

Het 2<sup>e</sup> bit (U/L) definieert of het adres is toegekend door IEEE of lokaal door software.

Elke netwerkcard bevat een wereldwijd (**Universal**) uniek MAC adres dat 'gebrand' zit in zijn ROM.

IEEE kent fabrikanten een 3 byte code toe : OUI = de eerste 3 byte van het MAC adres).

Hieraan voegen ze zelf een 3byte suffix toe om de kaarten uniek te houden.  
(→  $2^{24} = 16M$  kaarten per fabrikant-code... +/-).



Voor de laatste toekenningen zie [www.IEEE.ORG](http://www.IEEE.ORG) of <https://www.ipchecktool.com/tool/macfinder> → Intel = 0x 12:Ax:xx:, cisco 0x Cx:xx:xx...) waarbij x : 0..F.

(Let op met gecopieerde Virtuele Machine's! kies 'I copied' bij opstart om een nieuw uniek MAC adres te genereren. Anders behoudt de VM het MAC adres van het origineel. Komen beide VM's dan op 1 netwerk → problemen!)

Het 2<sup>e</sup> bit (U/L) = 'Local'. Om systeembeheerders toe te laten hun netwerk te organiseren en zelf groepsindelingen te maken op basis van MAC adressen worden plaatselijke adressen toegestaan. Let op, deze zijn niet meer uniek!!.

- **LENGTH**

Het **lengteveld** dient als transparantie-oplossing om het einde van het frame te herkennen ... (bits-)/ bytes tellen. Het is 16 bit breed en geeft het aantal **bytes** van het **data-veld** dat hierop volgt :

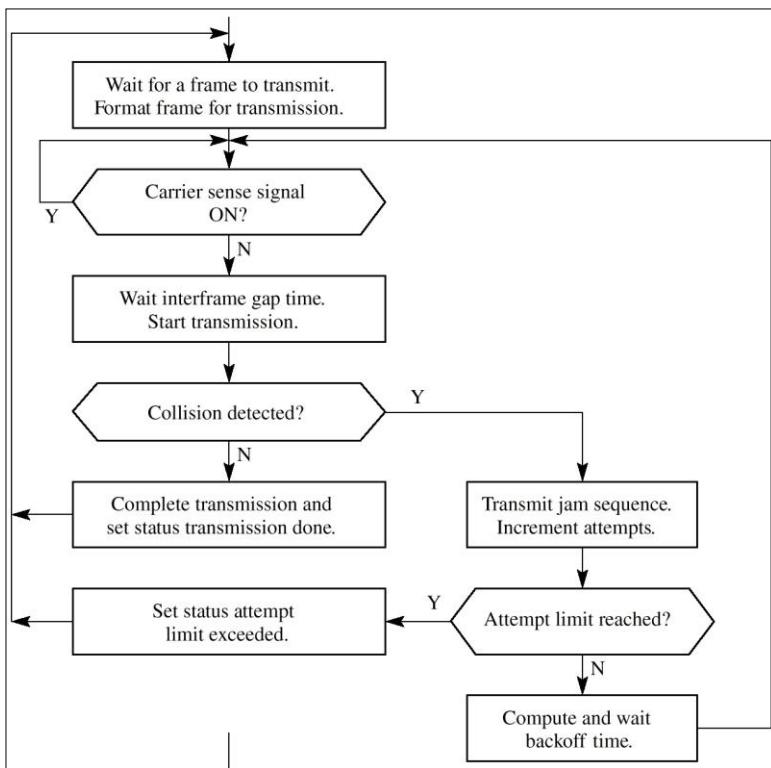
- **minimum = 46.** (een frame is min. 64 byte na SFD →  
length = 64-6-6-2-4= 64-18 = 46) .
- **Maximum** is de length indicator = **1500** (= 1518 –18).

Is er niet voldoende data voor een frame van 64 byte dan wordt het opgevuld met 'padding bytes' om collisions te detecteren. (bv. ARP frame).

- **FCS** : Na de eigenlijke data volgt een CRC32 voor error controle.

Bij een foute controle wordt het frame eenvoudigweg ge'dropped' zonder verdere actie. Eventueel moet dit opgelost worden door de hogere lagen, bv. TCP.

### ZEND- EN ONTVANGSTPROCEDURE.



Om een frame te **verzenden** controleert de MAC-interface het carrier sense signaal, en wacht (evt.) tot een lopend frame is gepasseerd.

Na een kleine wachttijd, de **interframe gap** = 9,6 µs, om de geadresseerde DTE(s) toe te laten het frame te ontvangen en te processen, start hij de transmissie.

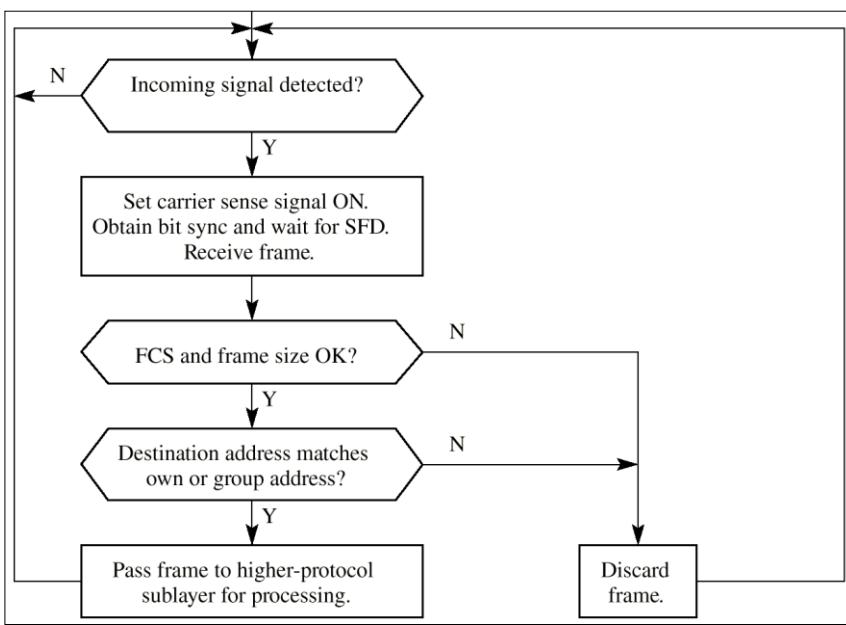
Tegelijk controleert de ontvanger of er geen collision optreedt, en zo ja wordt er ge'jammed'. Het frame wordt dan na een (random) backoff tijd herzonden.

Deze herzend-procedure wordt maximaal 16 keer herhaald, de '**attempt limit**'.

Figuur 231 De zendprocedure voor IEEE802.3

- **Ontvangstprocedure.**

De ontvanger zal naast de **FCS** ook de **lengte** van het frame controleren en het verworpen indien nodig.



We weten dat een collision vrij snel opgemerkt zal worden door een zender (<30µs), waarna de DTE in kwestie de verzending direct **afbreekt** en een jam sequence van **32 bit** (=3.2µs) opstuurt.

Dit stukje frame + JAM moet kleiner zijn dan de **minimum frame size** (=64byte, 51,2us) of de slottijd. Het wordt een 'RUNT' genoemd.

Een RUNT wordt verworpen door alle ontvangende DTE's bij de controle naar de juiste frame size.

Figuur 232 De ontvangstprocedure voor IEEE802.3

Een goed frame wordt alleen doorgegeven als het dest. adres juist is. (Tenzij in 'promiscuous' mode in het geval van bv. een sniffer.)

### 2.5.3 FAST ETHERNET

Fast Ethernet kende 3 uitvoeringen die we ook reeds gezien hebben in de fysieke laag, pt. 1.4.2 op p.95:

Implementation	Medium	Medium Length	Wires	Encoding
100Base-TX	UTP or STP	100 m	2	4B5B + MLT-3
100Base-FX	Fiber	185 m	2	4B5B + NRZ-I
100Base-T4	UTP	100 m	4	Two 8B/6T

#### CSMA/CD ?

Het ontdekken van collisions op een 3000m lang segment (10base5), met een roundtrip van 30μs, verplichte een min. frame van 51,2μs zijnde 512bit = **64 bytes** tegen **10Mbps**.

Wat met fast ethernet... **100Mbps** e.a. → 512 bit betekent dan **5,12μs** ... ???

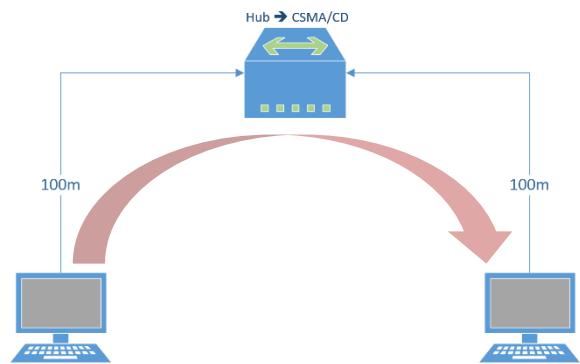
De oplossingen zijn voor de hand liggend : ofwel de **max. lengte** van een segment, en dus  $t_p$ , inkorten, ofwel het **MAC** protocol aanpassen (=min. frame lengte vergroten).

**FAST** ethernet kiest voor het eerste, de lengte : het wordt alleen ondersteund op 'hub/tree'<sup>1</sup> technologie (CAT3 → 100baseT4, CAT5 → 100base TX). In beide gevallen is de maximale afstand tussen DTE en hub **100m**.

De max. afstand tussen **2 DTE's** is bijgevolg 200m en de **roundtrip 400m**, wat neerkomt op :

$$2t_p = \frac{400}{\frac{2}{3} \times 3.10^8} = 2 \mu s$$

... m.a.w. kleiner dan de **5,12μs** van een **64 byte** frame tegen 100Mbps.

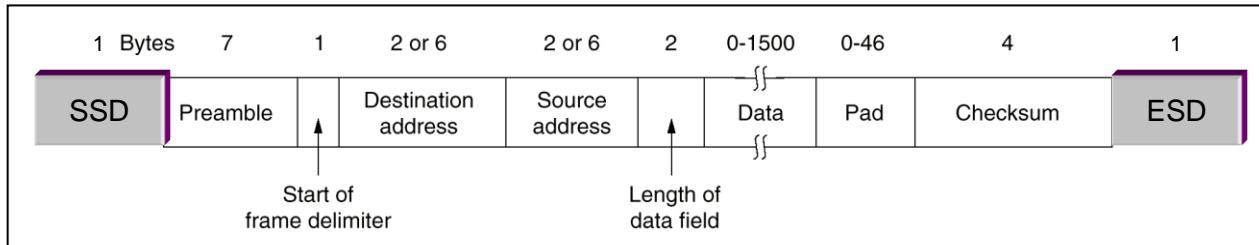


Figuur 233 Hub/tree afstand tussen DTE's.

Gezien de veel kortere round trip time kan men nu zelfs 2 hubs in cascade schakelen waardoor de onderlinge afstand (worst case) op het shared medium 2 x groter wordt (→  $2t_p = 4\mu s$ ).

<sup>1</sup> Merk op dat een hub een 'repeater'- of Laag 1- functie heeft en de fysieke signalen gewoon herhaalt, i.t.t. een bridge of switch op Laag 2 die collisions scheidt.

Er is weliswaar een minieme aanpassing van het IEEE802.3 frame van Figuur 230 (zie p. 192) door de toevoeging van SSD en ESD. Start of Stream Delimiter, SSD, bestaat uit de J en K symbolen van de 4B5B code, besproken op p.97 en p.98 Tabel 2 4B5B code link control, → SSD = 11000 10001. End of Stream Delimiter, ESD, = 01101 00111, zijnde de T en R codes resp.



Figuur 234 Fast ethernet frame.

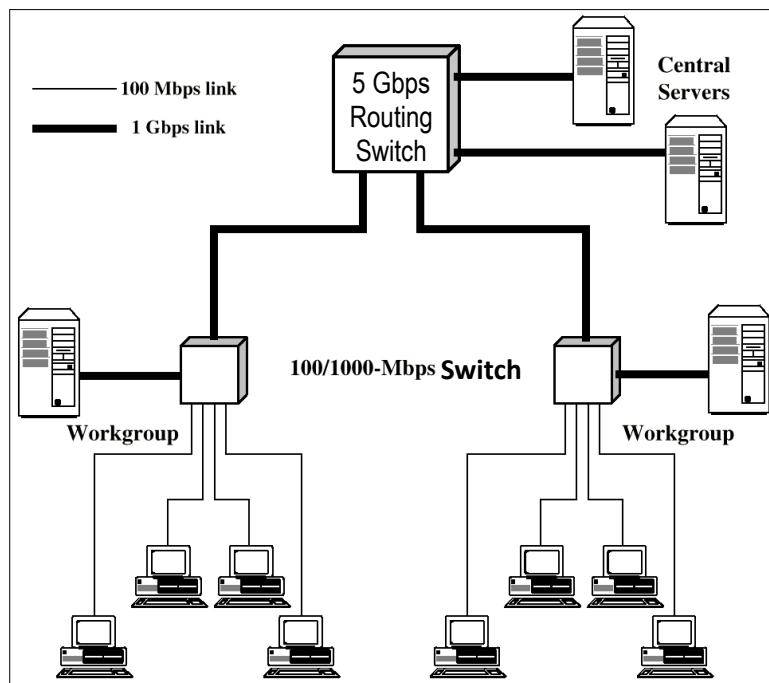
VG ANYLAN, ook gekend als IEEE 802.12, paste daarentegen het MAC-protocol aan, nochtans voorziet het dezelfde services naar LLC. Het ondersteunt grote (cascade-)netwerken zonder bridges, past variabele lengte van het MAC – SDU (Service Data Unit) toe om andere LAN-types te ondersteunen en laat verschillende soorten trafiek toe.

## 2.5.4 GIGABIT ETHERNET

De strategie voor gigabit ethernet is dezelfde als bij fast ethernet : (CAT5).

Gigabit ethernet **behoudt** het **CSMA/CD** protocol én het **ethernet frame-formaat**!

Dit om snelle omschakelingen toe te laten, een '*migration path*'. (praktisch is de '**switch**'-technologie meer van toepassing waardoor er vooral P2P verbindingen worden gebruikt.)



Een typische installatie is te zien hiernaast waar een backbone-routing switch de gigabit uplinks schakelt en snelle toegang voorziet voor de centrale servers.

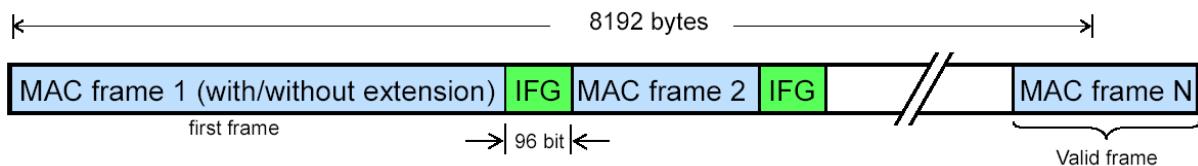
De werkgroep 100/1000 switches connecteren dan bvb. fast ethernet gebruikers, of (niet getekend) evt. kunnen via een nog lagere 10/100 switch (of hub) minder veeleisende gebruikers bediend worden.

Figuur 235 Een voorbeeld van een Gigabit ethernet configuratie.

Deze 100/1000 switches 'zouden' ook hubs kunnen zijn, maar dan is er een probleem : de minimum frame lengte moet aangepast worden van 512 naar **4096** bittijden ( 512 bit duurt nu, 1000bps, slechts **0,512us!**)

Zoniet is de transmissie van kleinere frames korter dan de propagatietijd bij 1Gbps →'carrier extension'. (Praktisch : gigabit-hubs niet te vinden).

In een shared medium omgeving (→CSMA/CD) wordt het dan zeer onefficient om deze zeer kleine frames te verzenden. (Aangevuld met padding/extensions om de minimum lengte te garanderen.) Daarom kent gigabit ethernet een '**burst feature**' waarin het 8192 bytes kan sturen in 1 burst.

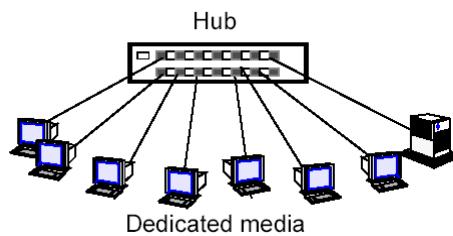
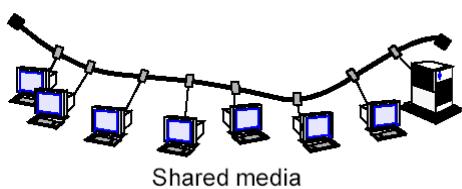


Figuur 236 Een burst frame in een Gigabit ethernet configuratie.

Het eerste frame wordt gewoon verstuurd, maar de carrier wordt nu niet gelost om anderen toe te laten op het medium. Elk frame volgt op het vorige na een kleine IFG, InterFrameGap van 96 bit.

## 2.5.5 EVOLUTIE IN ETHERNET

### VAN SHARED NAAR DEDICATED MEDIA.

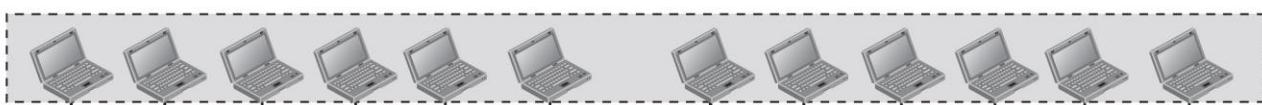


Figuur 237 evolutie van coax naar UTP in ethernet.

De eerste versie van ethernet gebruikte dezelfde kabel om alle hosts te verbinden op het netwerk. Een medium delen betekent niet alleen de bandbreedte delen, maar ook de problemen. In dit geval kon een gebruiker die de **coax** kabel loskoppelde (en dus onderbrak) het gehele netwerk ‘plat’ leggen.

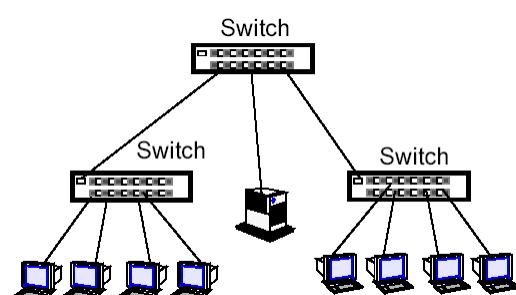
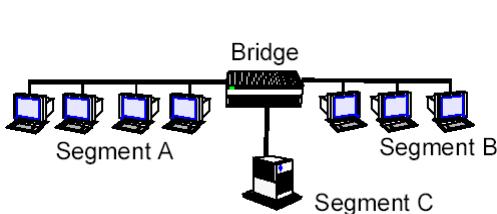
Een eerste verbetering was het gebruik van **hubs** in een sterbekabeling van **UTP**. Als nu een gebruiker het netwerk loskoppelde waren de hubs intelligent genoeg om deze poort gewoon af te sluiten en het netwerk verder te ondersteunen. LET wel OP : **de bandbreedte was nog altijd gedeeld!!** aangezien hubs koppelen op laag1! M.a.w. een ‘**half-duplex** verbinding, met een MAC principe van **CSMA/CD**.

Beide opstellingen vormen 1 ‘**ethernet segment**’ of m.a.w. 1 ‘**collision domain**’.



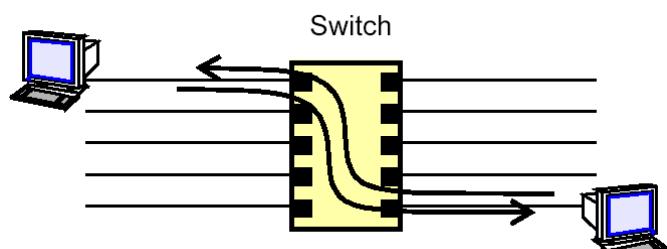
Figuur 238. 1 collision domain.

### VAN SHARED NAAR DEDICATED BANDBREEDTE.

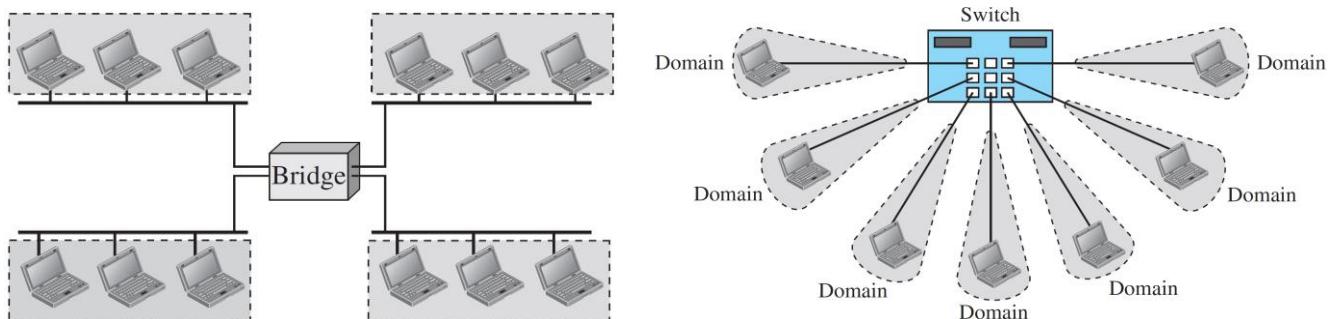


Figuur 239 evolutie naar L2 scheiding in ethernet.

Omdat het rendement op een shared bandwidth ethernet soms te wensen over liet bij hoge belasting heeft men toestellen ontwikkeld die een volledig L2 frame kunnen ontvangen, en die dit, afh.v. het L2 adres, enkel zullen ‘forwarden’ naar het gepaste segment. ➔ bridges voor coax en switches voor UTP.



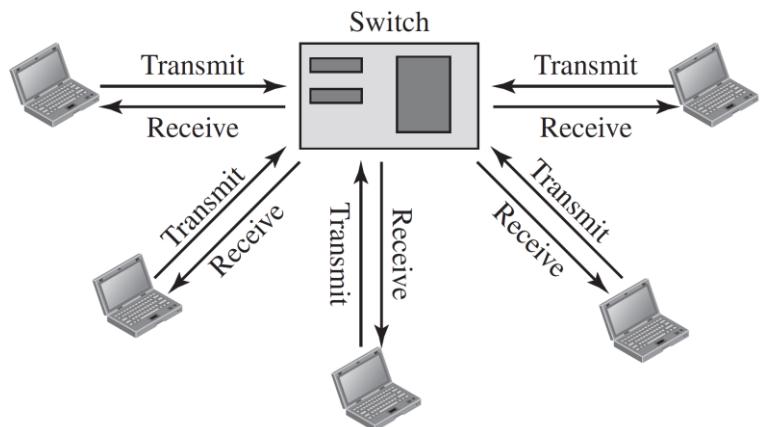
De ‘collision domains’ worden nu opgesplitst : elke uitgang van een bridge of switch vormt 1 ‘collision domain’.



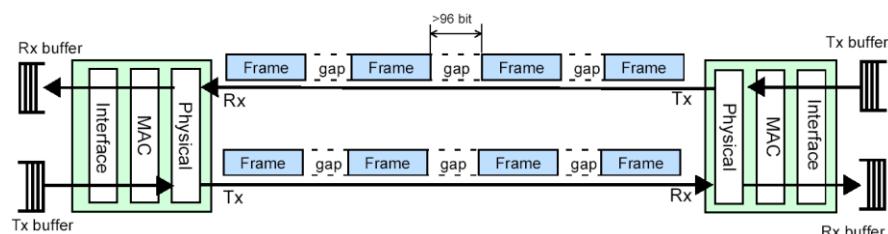
Figuur 240. Opsplitsing van het collision domain

In het geval van een switch heeft elke computer dus een volledig ethernet segment te beschikking, zelfs terzelfdertijd in **2 richtingen → full duplex!**

Er moet dus **geen** rekening worden gehouden met CSMA/CD, er zijn **geen collisions**.



OPM. Alle computers in de figuur zitten nog wel in 1 ('L3') netwerk, of 1 'broadcast domain'.



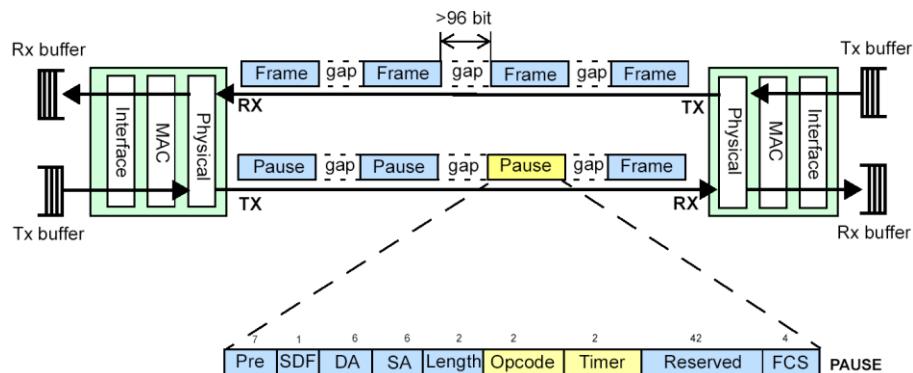
Figuur 242 Full duplex in ethernet.

Voor 10/100Mbps is er in de UTP kabel dan 1 paar voor verkeer van switch → computer, en een ander paar voor de tegengestelde richting. De ‘carrier’ moet niet ge’sensit’ worden omdat het medium niet busy is, en collisions kunnen niet optreden: er is slechts 1 zender per UTP paar.

Om de switches toe te laten het frame verder te zenden naar de juiste bestemming bevat het een MAC address tabel of **‘L2-forwarding-tabel’**. (= routing tabel → L3 routers, wel ± gelijkaardig). Hoe deze tot stand komt verwijzen we naar het lab. (Door het aanleren van de source adressen in de ontvangen frames).

Een probleem dat nu wel kan voorkomen is dat een zeer snelle zender (bv. PC) een ontvanger (bv. switch) overstelt. Daaraan moet er een oplossing geboden worden → **'flow control'**. Een ontvanger kan dus pause- frames

terugzenden om de trafiek te laten afkoelen.



Figuur 243 Flow control in ethernet.

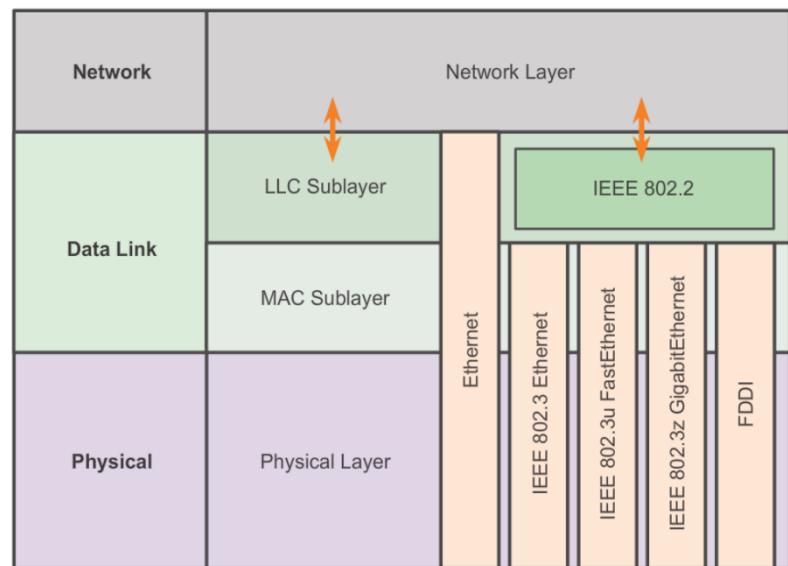
Een laatste opmerking betreft autonegotiation. 2 ethernet verbindingen die op mekaar worden aangesloten moet nu 'overeen komen' tegen welke snelheid ze gaan zenden (bv. 10/100/1000Mbps) én ook of ze FDX (full duplex) of HDX (half-), mét CSMA/CD dus, zullen sturen. Dit gebeurt door korte link pulsen die verschijnen tussen de frames. Voor meer informatie verwijzen we ook hier naar het lab en het internet.

## 2.5.6 VERSCHIL TUSSEN ETHERNET EN IEEE 802.3 FRAMES

Ethernet is een standaard, oorspronkelijk ontworpen door 3 fabrikanten : Digital, Intel en **Xerox** → DIX.

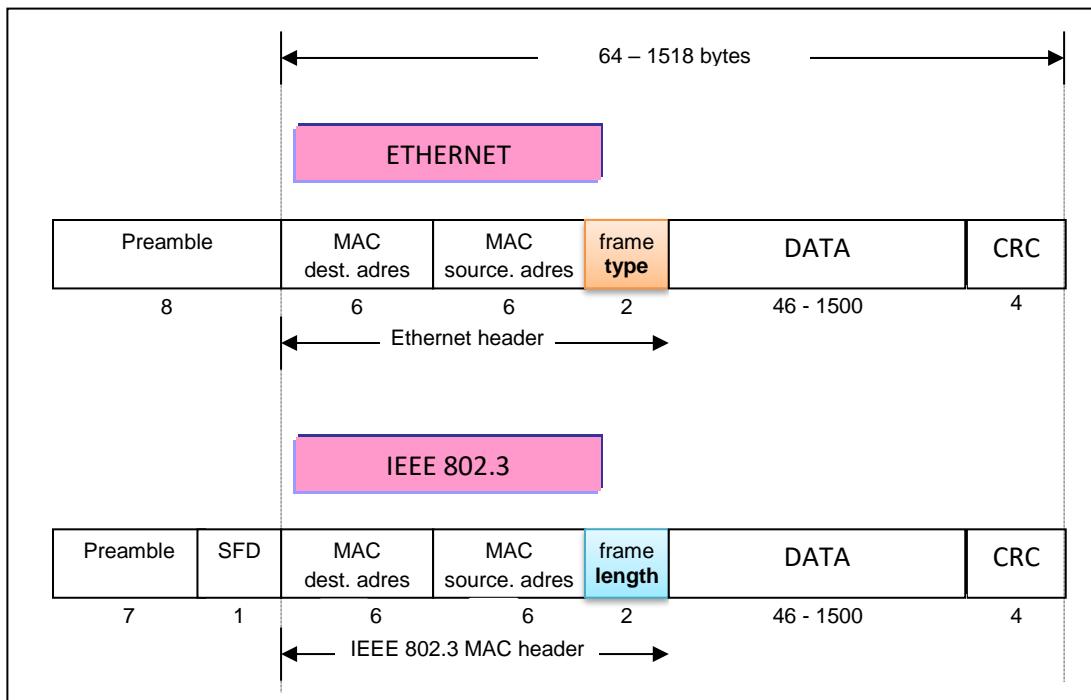
IEEE heeft deze industrie-standaard overgenomen/gestandardiseerd in de **802.3** norm en uitgebreid met 802.4 en 802.5, ... . Overkoepelend voor al deze 802 netwerken is de **802.2 LLC** gedefinieerd. Zie ook Figuur 212. Datalink structuur bij LAN's op p.176. (hiernaast vooral 802.3x).

Maar ethernet op zich is daardoor niet verdwenen, ze bestaan nu **beiden, naast mekaar** op dezelfde L1 **10(0)(0)(G)Base** netwerken die we besproken hebben in hoofdstuk 1.



Figuur 244 Ethernet vs IEEE stack

MAAR ... er is een **verschil** in frame formaat tussen **ethernet** en (het koppel 802.2 -) **802.3** !! In de TCP/IP wereld is de ethernet verpakking gedefinieerd in **RFC 894**, IEEE in **RFC 1042** waarbij alle hosts **beide** standaarden moet ondersteunen, zelfs tegelijkertijd.



Figuur 245 Datalink frame formaat a) ethernet b) IEEE 802.3

'Het' verschil is veld 3, na source en dest. MAC adres : bij ethernet is dit het **TYPE** veld i.t.t. **LENGTH** bij IEEE 802.3.

## ETHERNET : TYPE VELD

Het idee van ethernet was om op 1 fysische bekabeling verschillende netwerkstandaarden te ondersteunen. Het doel van het **type**-veld is om deze standaarden te herkennen en de frames door te geven aan de resp. netwerklaag. Het TYPE veld is dus de **NLPID**, bvb. :

Type-veld (geschiedenis):

- 0x 06 00 : XNS (Xerox, de ontwerper van ethernet kreeg het laagste typenr.)
- 0x 08 00 : **IP** Internet Protocol
- 0x 81 37 : netware IPX
- 0x 80 D5 : IBM SNA
- 0x 80 9B : appletalk
- ... huidige: <http://www.networksorcery.com/enp/protocol/802/ethertypes.htm>

## IEEE : LENGTH VELD

IEEE gebruikt dit veld voor ‘framesynchronisatie’, om het einde van het frame te kunnen berekenen. (‘Bits/bytes tellen ➔ ethernet’ op p. 91. )

Deze verschillen maken beide standaarden **incompatible!**

## ETHERNET ↔ IEEE

Het ‘**toeval**’ wil nu dat het **laagste typeveld** dat is toegekend, **0x 06 00** van XNS, groter is dan de **max. lengte** van een frame : **1500 databytes** zonder CRC. ( $0x 06 00 = 6 * 256 = 1536$ ).

M.a.w. bij ontvangst van een type/length veld bepaalt de inhoud hiervan ( $> \text{of} \leq 1500 = 0x05DC$ ) of het over **etherent** ( $> 1500$ ) dan wel over **IEEE 802.3** ( $\leq 1500$ ) gaat. Netwerkkaarten worden zó gefabriceerd dat ze dit onderscheid kunnen maken.

## ETHERNET: ? FRAMESYNC? LENGTH ?

Alle netwerkprotocollen, IP, IPX, DIX voorzagen zelf lengtevelden in hun headers waardoor ethernet de lengte van het frame kon bepalen. IEEE betrouwde niet op andere (L3) protocollen (= ‘layer violation’) voor de goede werking van zijn datalink laag en heeft daarom het **typeveld** vervangen door de **lengte**.

## IEEE: ?NLPID?

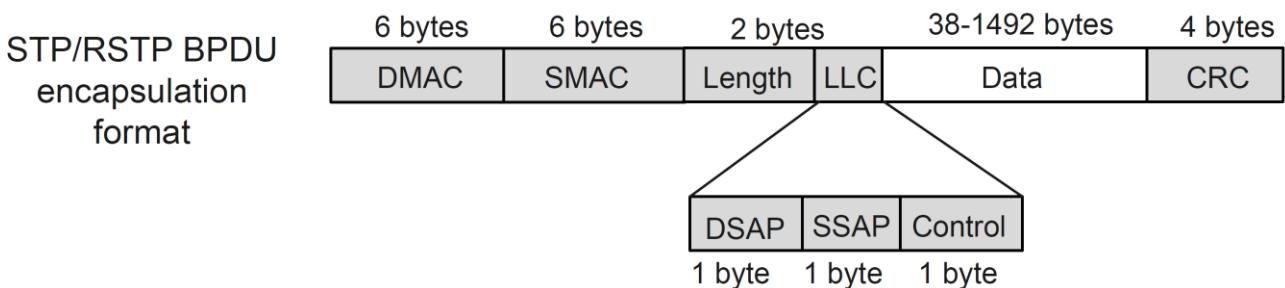
Maar... hoe lost IEEE het probleem van de **NLPID** op?

**IEEE 802.3**, het alternatief voor de ethernet MAC laag, bevat boven zich **STEEDS LLC = 802.2!!**

Deze deellaag zal de NLPID voorzien!.

(**Ethernet** definieert de **volledige L2**, datalink laag, terwijl **IEEE** het splitst in **2 sublayers!**).

Bv. IEEE's eigen spanning tree protocol 802.1d gebruikt 802.3 – 802.2 encapsulatie, waarbij DSAP = SSAP = (0x42) :



Figuur 246 IEEE's STP 802.1d encapsulatie door 802.3 – 802.2

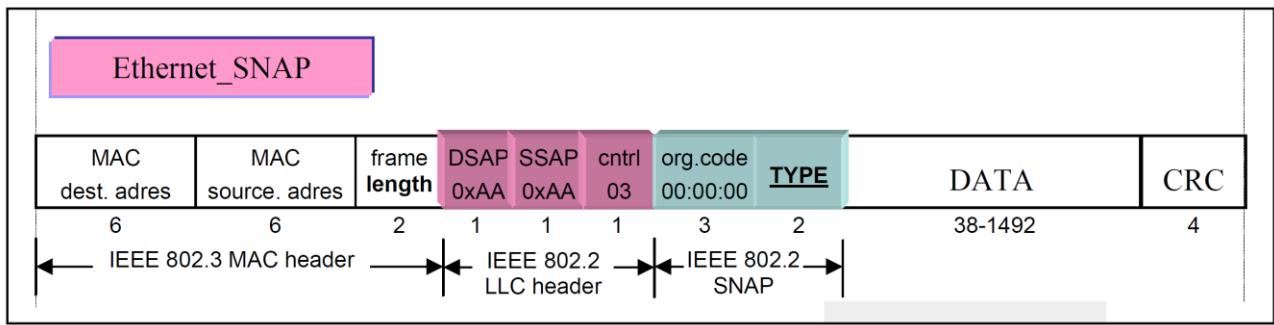
Andere vbn waren : DSAP:SSAP voor SNA = 0x 04:04, = 0x E0:E0 voor NETBUI, ...).

Maar een 1byte SAP als NLPID was onvoldoende om een 2-byte **type** veld van ethernet te vervangen, dus creëerde men een speciale vorm, **SNAP** (= SubNetwork Attachment Point), die eigenlijk het DIX-formaat herschikte →

### ETHERNET SNAP

Om de vele **verschillende** netwerkprotocollen zoals IP, appletalk, netware, ... in het IEEE 802.3 frame te transporteren heeft IEEE de ethernet\_SNAP oplossing gestandardiseerd.

De MAC deellaag IEEE 802.3 is ontworpen om te werken met zijn overkoepelende LLC deellaag IEEE 802.2. Vandaar dat in het dataveld van IEEE802.3 als **eerste 3 bytes** de **LLC** header voorkomen : **DSAP**, **SSAP** en **control**, zie ook Figuur 214 op p. 179.



Figuur 247 Ethernet\_SNAP frame.

Om nu Ethernet\_SNAP te herkennen gebruikt 802.2 in dit geval:

- **LLC** : DSAP:SSAP = 0x AA:AA (vast) en controle-veld = 03 (= unnumbered information).

Dit wijst op een SNAP header die volgt:

- **SNAP** : 3 bytes OUI, Organisation Unique Identifier, = 00:00:00 voor de organisatie die ethernet specificeert.

In dat geval volgt:

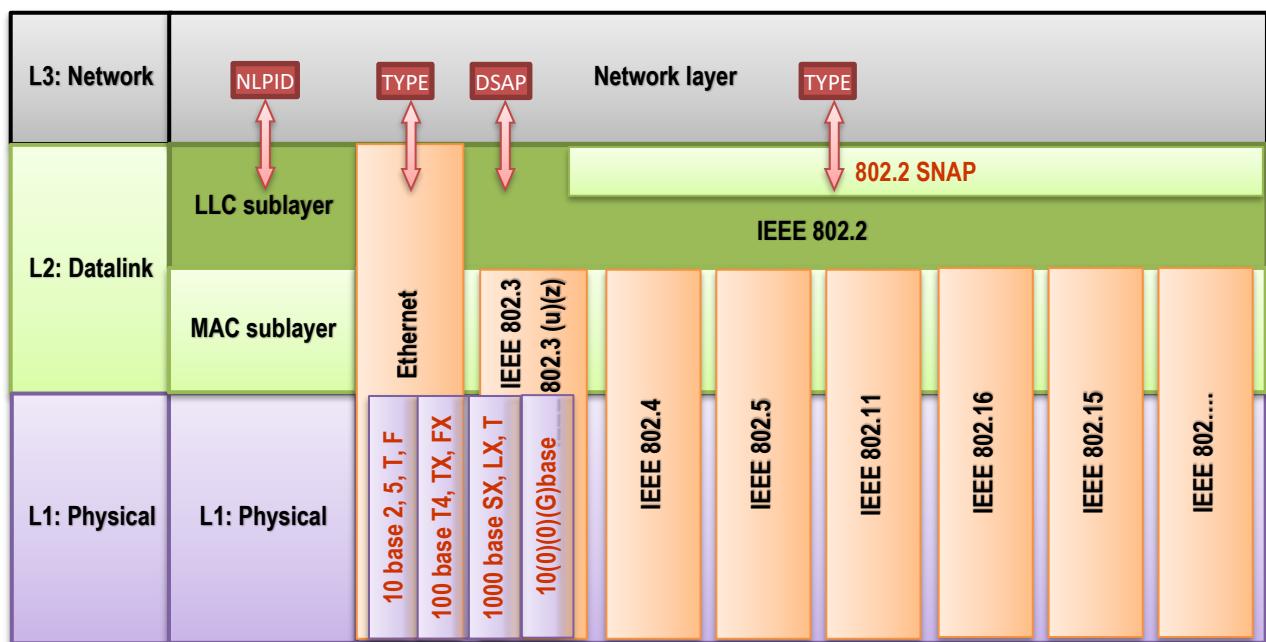
- **typeveld** : 2 bytes, identiek aan het ethernet typeveld, bvb. IP = 0x 08:00

### Besluit

Bij ontvangst van een frame op ethernet/802.3 wordt na de adressen het **type/lengte** veld gecontroleerd.

- Is het  $> 1500 \rightarrow$  dan is het een ethernet frame en is dit veld het **type** veld.
- Is het  $\leq 1500 \rightarrow$  IEEE802.3 frame. De volgende (2) bytes worden onderzocht.
  - ? = AA:AA  $\rightarrow$  **ethernet\_SNAP**, het type veld volgt 8 bytes verder dan bij ethernet.
  - ... (niet AA AA) dan zit hierin de 802.2 NLPID. Zoals bv. bij 802.1d

### OVERZICHT ETHERNET – IEEE 802 PROTOCOLLEN.



Figuur 248 Overzicht ethernet – IEEE 802.3 protocollen

Als we de protocollen vergelijken in het OSI model zien we een aantal merkwaardigheden:

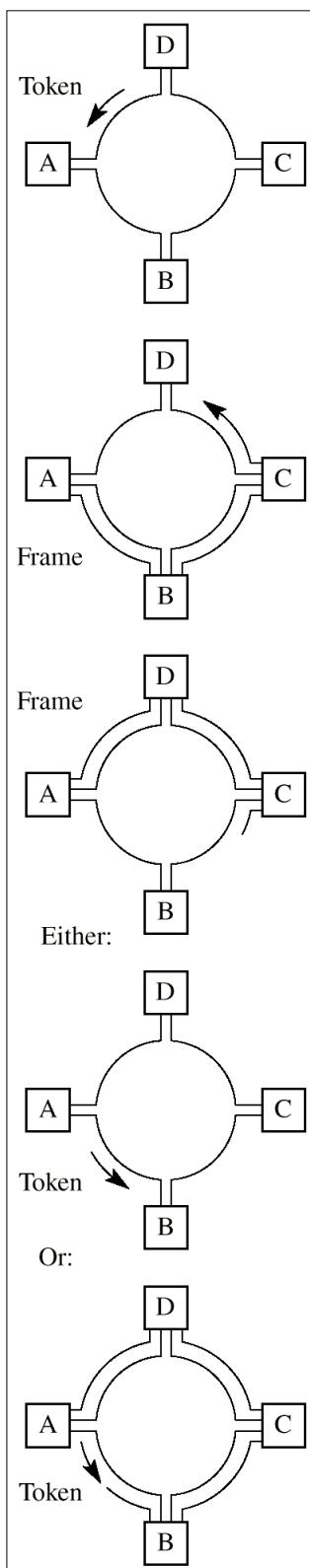
- zowel **ethernet** als **IEEE 802.3** definieren standaarden die de grenzen van het OSI model overschrijden : ze definieren ook hoe L1 eruit ziet, 10/100/1000 base ...
- Ze gebruiken **TEZAMEN** dezelfde L1!
- Bovenop de IEEE standaarden rust steeds de **IEEE 802.2 LLC**, maar **niet over ethernet !**
- Ethernet definieert de volledige laag 2! Geen sublayers.
- 802.2 werkt meestal met de SNAP toevoeging om voldoende netwerklagen te identificeren in zijn NLPID

- Wat we hier missen zijn een aantal nieuwe standaarden die naast IEEE 802.3 en IEEE 802.5, ... netjes passen onder de IEEE 802.2

- [IEEE 802.3 Ethernet](#)
- [IEEE 802.4 Token bus](#)
- [IEEE 802.5](#) Defines the MAC layer for a [Token Ring](#)
- [IEEE 802.11 Wireless LAN](#) & Mesh ([Wi-Fi](#) certification)
- IEEE 802.14 [Cable modems](#)
- [IEEE 802.15 Wireless PAN](#)
- [IEEE 802.15.1](#) ([Bluetooth](#) certification)
- [IEEE 802.15.4](#) ([ZigBee](#) certification)
- [IEEE 802.16 Broadband Wireless Access](#) ([WiMAX](#) certification)
- [IEEE 802.16e](#) (Mobile) Broadband Wireless Access
- [IEEE 802.17](#) Resilient packet ring
- [IEEE 802.20](#) Mobile Broadband Wireless Access
- [IEEE 802.22](#) Wireless Regional Area Network

## 2.6 TOKEN PASSING OP EEN TOKEN RING IEEE 802.5.

### WERKING



Stel DTE A wenst een frame te zenden naar DTE C.

Als een DTE (A) een frame wil zenden moet het wachten op het **token**. Dit komt van zijn 'upstream' buur (DTE D). Krijgt A dit dan start het een frame met het bestemmingsadres erin. Het frame wordt door **elk station** ontvangen en **bit per bit** heropgezonden, tot het terug aankomt in de zender die het **verwijdert** van de ring.

De bestemming (C) zal bovendien een **kopie** van het frame bijhouden en een **respons bit** zetten op het einde van het frame.

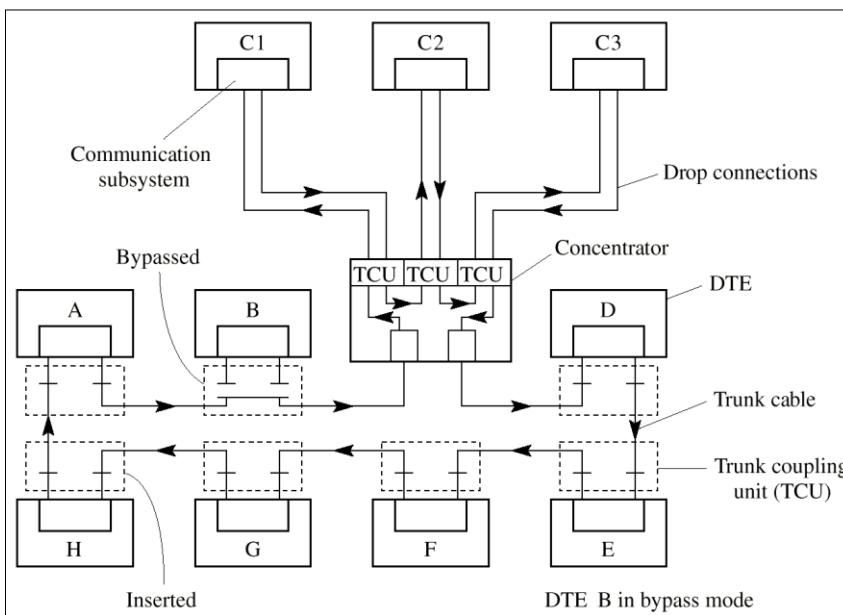
DTE C zal het frame echter doorzenden.

DTE A wacht tot het zijn eigen frame terug ziet verschijnen, maar herhaalt het nu niet, waardoor het frame van de ring wordt gehaald.

Een DTE 'released' het token op 2 mogelijke manieren :

- als hij de laatste bit van zijn eigen frame terug heeft **ontvangen**. (4Mbps)
- als hij het laatste bit van zijn eigen frame heeft **verzonden** → **'early token release'**. (16Mbps)

Figuur 249 Principe van een token ring netwerk.

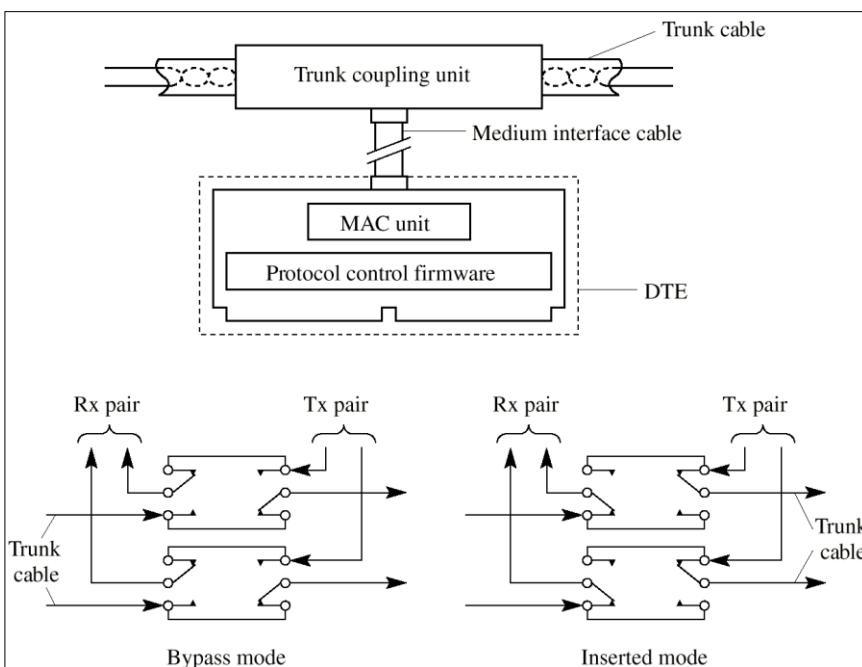


De kabel (het medium) is normaal een (afgeschermd) **twisted pair** die 'differential manchester' gestuurd wordt tegen 4 of 16 Mbps.

Elk segment is een P2P link. De DTE's worden ofwel **direct** verbonden met de ring, ofwel via een 'concentrator' → CAT5 bekabeling via patch panels.

Er worden maximaal 250 DTE's op 1 ring toegelaten.

Figuur 250 Fysische layout van een token ring netwerk.



De TCU, Trunk Coupling Unit, bevat een stel relais (en electronica) die de ring doorschakelen als de DTE afgezet wordt → **bypass state**.

Wordt het toestel aangezet, dan zal de MAC eenheid op de NIC de invoeging in de ring van deze DTE controleren en de relais activeren. Hierdoor moeten alle frames door deze MAC passeren.

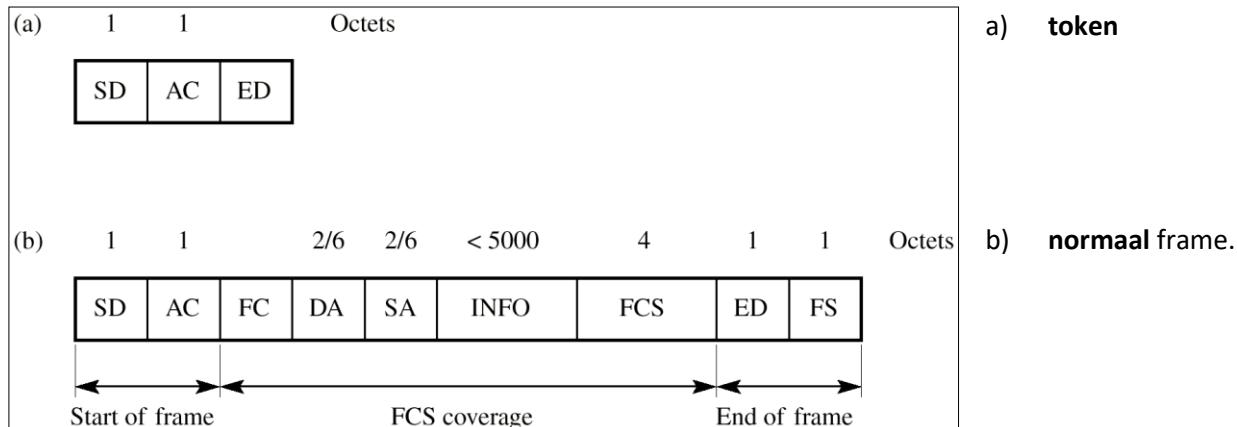
Figuur 251 De werking van een TCU voor token ring.

De bits worden '**differential manchester**' gecodeerd (waarvan de frequentie is opgelegd door de active ring monitor, en waarop de andere DTE's 'locken' met een PLL). Deze active ring monitor moet bovendien zorgen voor de **minimum latency tijd** : d.i. het aantal bittijden dat een signaal nodig heeft om de hele ring door te lopen.

Het probleem stelt zich bij het ronddraaien van het **token** dat alle DTE's ontvangen en direct bit per bit verzenden. In dit geval is het mogelijk dat de eerste bit van dit token (24 bit lang) de laatste zou overschrijven → minimum latency tijd is minstens 24bit. (Het probleem stelt zich niet bij dataframes aangezien de zender het er terug afhaalt). De active ring monitor zal nu een buffer van 24 bits voorzien die effectief deel uitmaakt van de ring, **voor alle frames** dus.

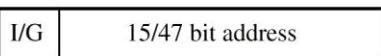
## IEEE 802.5 FRAME FORMAAT

Er zijn 2 formaten te onderscheiden :



Figuur 252 IEEE 802.5 frame formaat : a) token, b) normaal frame.

Voor datatransparantie zijn er de **SD** (Start Delimiter) en **ED** (End Delimiter) velden die opgebouwd zijn uit Manchester-code violations J en K, zie ook transparantie p. 153

	= Start delimiter (SD)
	= End delimiter (ED)
	= Access control (AC)
	= Frame control (FC)
	= Source and destination address (SA/DA)
	= Frame status (FS)

Merk echter op dat **ED** 2 speciale bits op het einde bevat : **I** en **E**.

### I-bit :

- In een **token** : **I=0**
- In een **normaal frame** : **I=0** als het om het **laatste** I-frame gaat in een boodschap,

**I=1** voor alle andere frames.

Figuur 253 IEEE 802.5 veldopbouw.

### E-bit :

- Het E bit is een **error-correction** bit : de zender plaatst E = 0. Als één van de DTE's in de ring een foutdetectie doet (CRC32), dan plaatst die bit E=1. Ook als deze DTE zelf niet de bestemming was.

Een voordeel aan token ring is dat er prioriteiten aan de boodschappen kunnen worden gegeven via het **AC** (Access Control) **veld** (zowel in token als andere frames).

- De **P**-bits duiden de **prioriteit** aan van het token, m.a.w. vanaf welke prioriteit een DTE mag zenden als hij het token ontvangt.
- **T** bit : =0 voor een token frame, =1 voor de andere
- **M** bit : monitor bit, voor de ring-monitor, om te verhinderen dat een frame blijft rondlopen in de ring. De ring monitor zet deze bit, en als het frame nogmaals zou passeren ontvangt hij een

frame met M=1. ➔ d.i. een ‘orphan’, een frame waarvan bvb. de zender is afgezet voor hij het van de ring kon halen.

- **R** bits : de reservatie-bits. DTE’s die **hoge-prioriteitsframes** bevatten kunnen hiermee het volgende token een hogere prioriteit geven (RRR ➔ PPP). Natuurlijk moet deze prioriteit na verloop van tijd terug verlagen.

Het **FC** veld maakt in een normaal frame met de FF-bits onderscheid tussen **Informatie-** en **MAC**-frames (zie verder). Is het een **MAC** frame, dan luisteren **alle** DTE’s naar de controlebits Z, een I-frame wordt alleen geïnterpreteerd door de **destination**-DTE.

**SA** en **DA** zijn net zoals ethernet (ofwel 2 bytes of) 6 bytes lang, het I/G bit duidt op een Individueel adres (I/G=0) of een Groepsadres (I/G=1). Zijn alle bits = 1 voor DA, dan is het een broadcast.

Het **Info**-veld is maximum **5000 bytes!!!** (Problemen voor bridges tussen IEEE802.3 met MTU=1500 bytes en IEEE 802.5). (MTU = Maximum Transfer Unit)

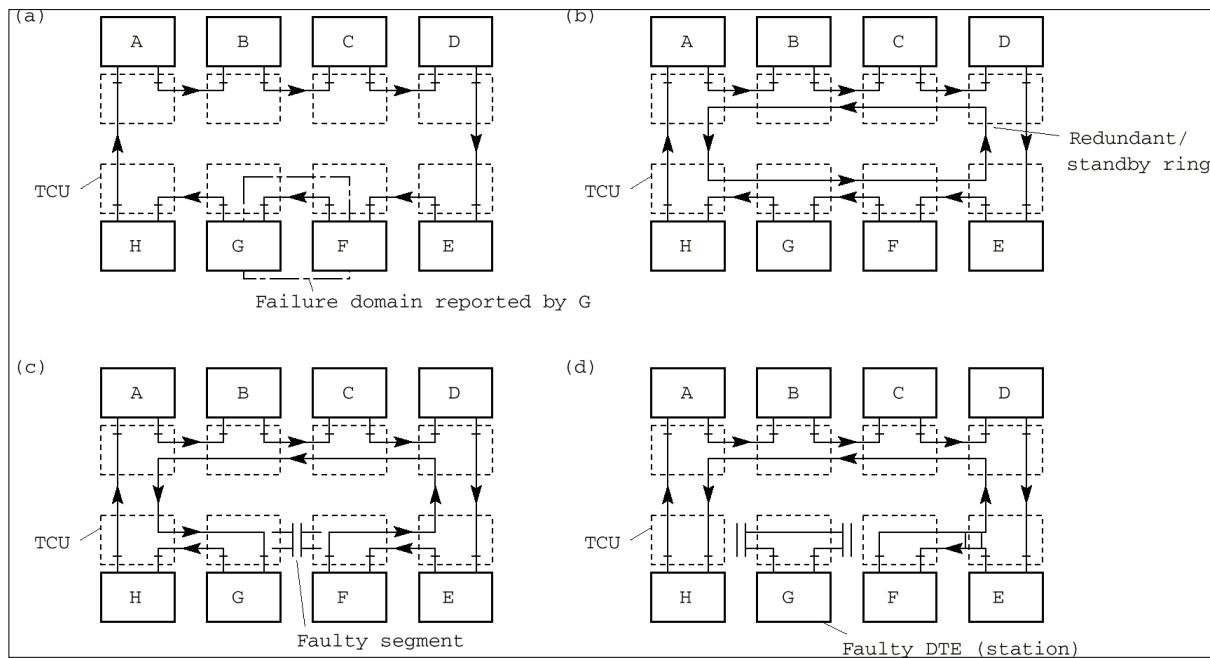
De **FCS** is een CRC32.

Het **FS** veld bevat 2 bits : **A** en **C**, die beiden = **0** worden geplaatst door de zender. Wordt het frame **herkent** door 1 of meer DTE’s op de ring, dan worden de **A**-bits gezet ➔ ‘Address-recognized’. **Kopieert** een DTE het frame, dan worden de **C**-bits gezet. Op die manier kan de zender te weten komen of een DTE al of niet is **ingeschakeld**, en of hij de boodschap heeft **ontvangen**.

**MAC**-frames worden gebruikt voor ‘**ring-management**’ : bv. als een DTE wordt ingeschakeld en actief wil deelnemen in de ring moet hij eerst controleren of er geen andere DTE’s zijn met **hetzelfde adres** (➔DAT frame, Duplicate Address Test). Andere frames geven de aanwezigheid van de actieve monitor weer : **AMP**, (Active Monitor Present), als deze niet meer verschijnen (en ook geen tokens) zal de ‘standby monitor’ een **CT** (Claim Token) frame sturen en zonodig de actieve monitor worden. Ring monitoren worden gebruikt om het verlies van het tokenI te detecteren. Moest dit optreden dan ligt alle communicatie op de ring stil.

## BEACONING.

Als er een ‘kink’ in de kabel komt zal een ‘beaconing’-procedure elke DTE op de hoogte brengen van de breuk en, voor een enkelvoudige ring, stopt de transmissie tot het euvel is hersteld. Daarom wordt echter veel gebruik gemaakt van een **dubbele** (redundante) ring die stuurt in de tegenovergestelde richting van de eerste, zie Figuur 254 b). Is er nu een fout segment tussen F en G, dan wordt de tweede ring ingeschakeld zoals in figuur c). Is dit niet voldoende dan wordt zelfs station G volledig uitgeschakeld, zoals te zien is in figuur d). Merk op dat de DTE-volgorde in de ‘herstelde’ ringen dezelfde blijft.



Figuur 254 Foutdetectie op een ring-netwerk

Op te merken valt dat de MAC procedures op een token ring netwerk een dimensie complexer zijn dan bv. op ethernet.

Token ring's worden toegepast in IBM's token ring en in een vernieuwde versie in FDDI. FDDI is een 100Mbps netwerk dat, gebaseerd op een dubbele glasvezelring, 100km ver betrouwbare communicatie voorziet tussen max. 500 stations. De tweede ring kan gebruikt worden als supplementair transmissiepad, of als backup bij een breuk. De prijs verhindert FDDI, een ANSI standaard, om een uitgebreide standaard te worden.

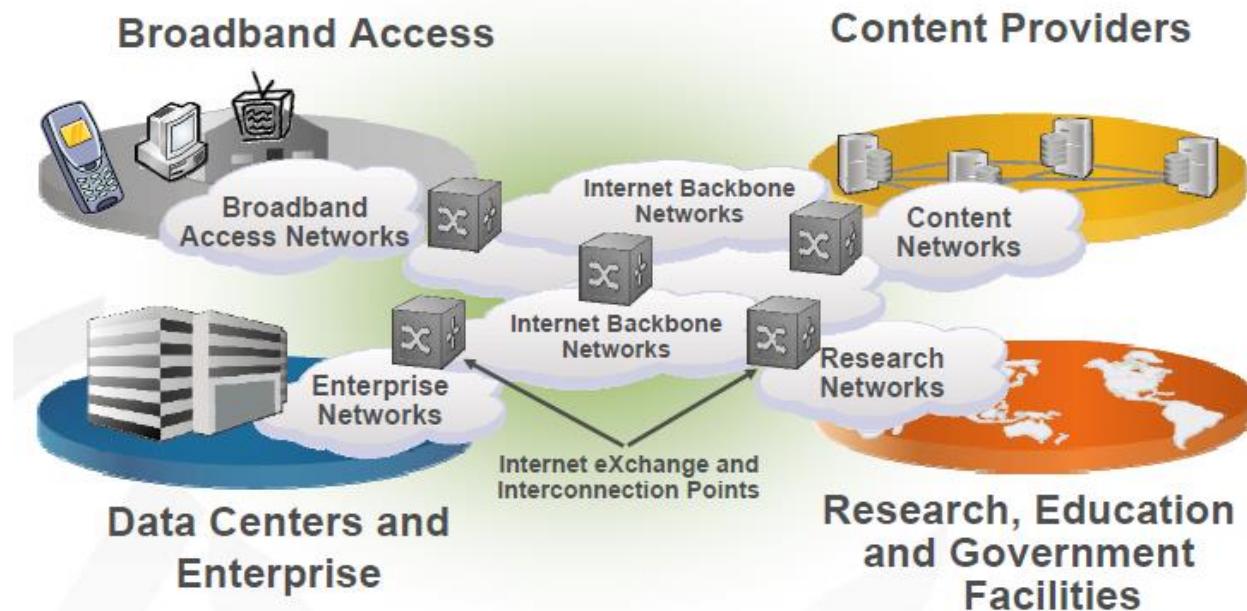
Andere principes van netwerken zijn : **token bus** en **draadloze LAN's**. Token bus maakt gebruik van de token-passing techniek, maar op een bus-medium (meestal coax). Het voordeel om **prioriteiten** aan frames te geven, en zodoende belangrijke (kritieke) informatie eerst door te sturen maakt het uitermate geschikt in de procesindustrie (fabrieksautomatiserings-) middens.

# 3 DE NETWERKLAAG.

## 3.1 INTERNETTEN : ARCHITECTUUR EN ONTWERPASPECTEN.

In voorgaande hoofdstukken hebben we ons bezig gehouden met de fysische laag en datalink-laag. We stappen nu een trede hoger in het OSI-model : de netwerklaag. Om de organisatie hiervan te begrijpen is het handig om een idee te krijgen over de soorten data (trafiek) die er op het netwerk verschijnen, het soort verbindingen, de service die moet geleverd worden, adresseringen, enz.. .

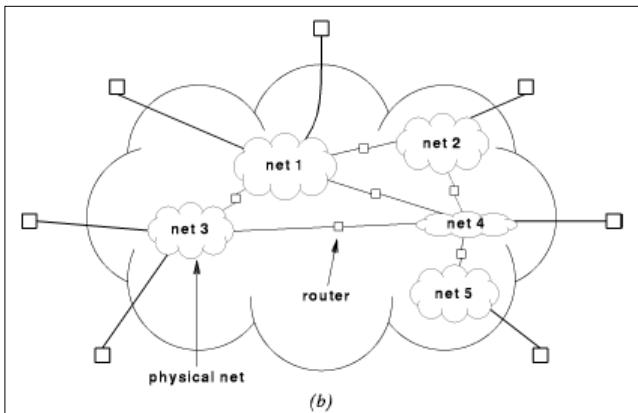
We hernemen hiervoor Figuur 1 Communicatie entiteiten van p. 20 om een totaalbeeld te krijgen van het moderne internet. Bovenaan rechts zien we dat ‘home users’ voor verschillende services, typisch voice video en data, worden aangesloten via access networks die aangeleverd worden door ISP’s (Internet Service Providers). Ook content providers, overheidsinstellingen en de grote bedrijven kunnen rechtstreeks op de nationale backbones worden aangesloten, of voor de kleinere entiteiten gebeurt dit via de ISP’s.



Figuur 1 Communicatie entiteiten

Technisch gezien zijn al deze verbindingen identiek en steunen ze op het IP protocol. Voorlopig is dit nog steeds bij voorkeur IPv4, maar IPv6 is sterk aan het oprukken. Dit hoofdstuk zal dus vooral IPv4 behandelen, voor IPv6 verwijzen we naar de cursus 5 infrastructuur.

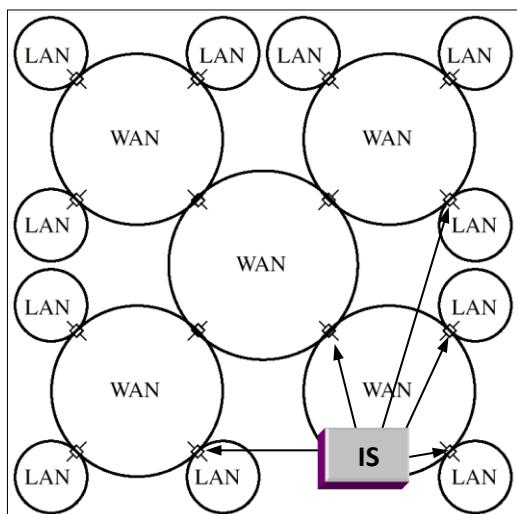
### 3.1.1 ARCHITECTUUR VAN INTERNETTEN



De netwerklaag heeft als doel om **pakketten**, komende van een bron af te leveren aan een bestemming. De aangelegde weg kan **wereldwijd** zijn, over verschillende netwerken, verbonden via een groot aantal routers of 'IS' (Intermediate Systems).

Dit in tegenstelling tot de **datalink-laag** die **frames** moet afleveren, enkel tot het '**einde van de kabel**' !

Figuur 255 Het internetconcept<sup>1</sup>.



De verbindingen tussen geografisch gescheiden netwerken worden WAN links genoemd (→ 'Wide Area Network'), en worden meestal verzorgd door '**ISP's of Internet Service Providers**: telenet, belgacom, BT, ...<sup>2</sup>

Aan de rand van het internet bevinden zich de verschillende (bedrijfs-)LAN's.

Al deze LAN's en WAN's worden op laag 3, IP, op een gelijkaardige manier verbonden via routers (=IS).

Figuur 256 LAN – WAN verbindingen.

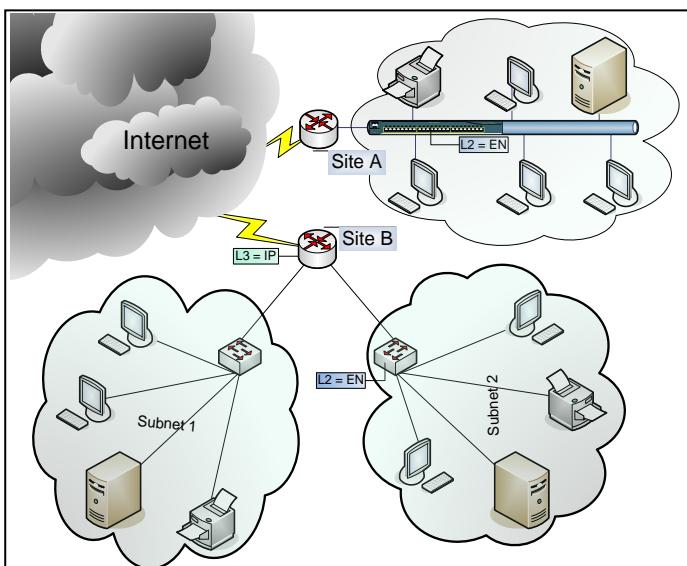
Het enige principiële verschil tussen WAN's en LAN's is de fysieke en datalink laag (L1+L2) die voor WAN's langere afstanden kan afleggen.

<sup>1</sup> **Historiek** : het wereldwijde Internet (met hoofdletter I) is ontstaan omstreeks 1993-'94. Voorlopers waren het (D)ARPA-net ((Defense)Advanced Research Projects Agency), een netwerk van universiteit-computers als project van DoD (Department of Defence) van de USA, 1970.

Halfweg de jaren '80 heeft de NSF, National Science Foundation, een nieuw **NSF**-net gefinancierd dat universiteiten verbond, gebaseerd op de ervaring met ARPA.

<sup>2</sup> Een bedrijf kan weliswaar zijn eigen WAN verbinding verzorgen door 'lijnen' te huren van 'operators' zoals belgacom.

Meer specifiek kan een **LAN** bestaan uit een aansluiting op de ISP via een router. Bv. een 'home netwerk' zoals site A.

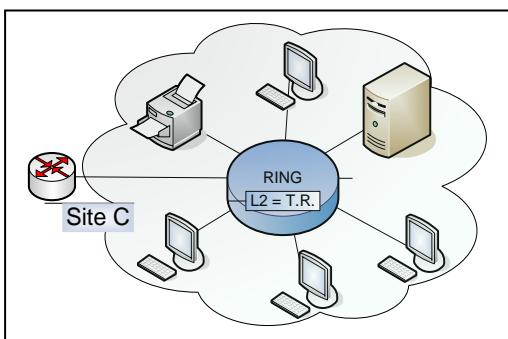


Een andere mogelijkheid, site B, is een bedrijfsnetwerk dat bestaat uit een aantal **subnetwerken**, ook verbonden met (een) **router(s)**.

In alle gevallen verbind je netwerken (wolken) met elkaar via een router. Bv. het home-netwerk van site A met het netwerk van zijn ISP. Of subnet1 naar subnet2 op site B...

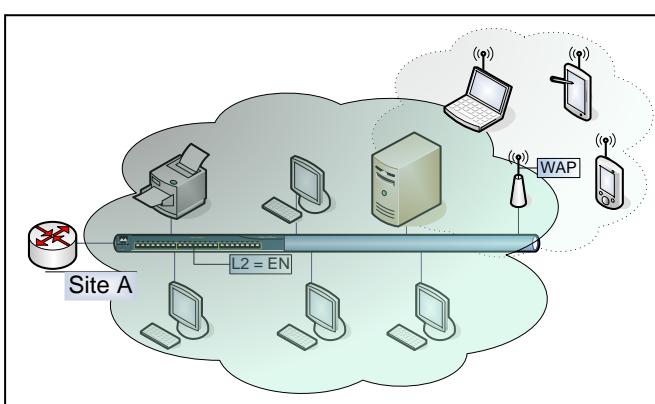
Figuur 257 Verbinding tussen (sub)netwerken.

**Binnenin een (sub-) netwerk** worden de verschillende apparaten verbonden (meestal) met een switch (L2) of soms nog via een hub (L1). Veel is hierbij de gebruikte L2 = EN, EtherNet.



Dit is echter geen must : de verschillende LAN's kunnen binnenv 1 wolk van een verschillend datalink-type zijn : bv. naast Ethernet ook token ring, token bus, wireless, ... → de tussenliggende routers/gateways hebben een **specifieke uitgang per netwerktype** om op dit netwerk aan te sluiten, hier bv. T.R. = token Ring.

Figuur 258 Een Tokenring netwerk.



Een andere mogelijkheid is dat binnenv 1 netwerk 2 verschillende L2's met elkaar verbonden worden m.b.v. een 'bridge' (=L2)

Een veelvoorkomend voorbeeld hiervan is om een ethernet-netwerk uit te breiden met een 'wireless' toegang via een WAP : 'Wireless Access Point'. Dit WAP vormt de brug tussen de draadloze- en draadgebonden gebruikers, **binnenin 1 netwerk**.<sup>1</sup>

Figuur 259 Een Wireless uitbreiding op een ethernet netwerk.

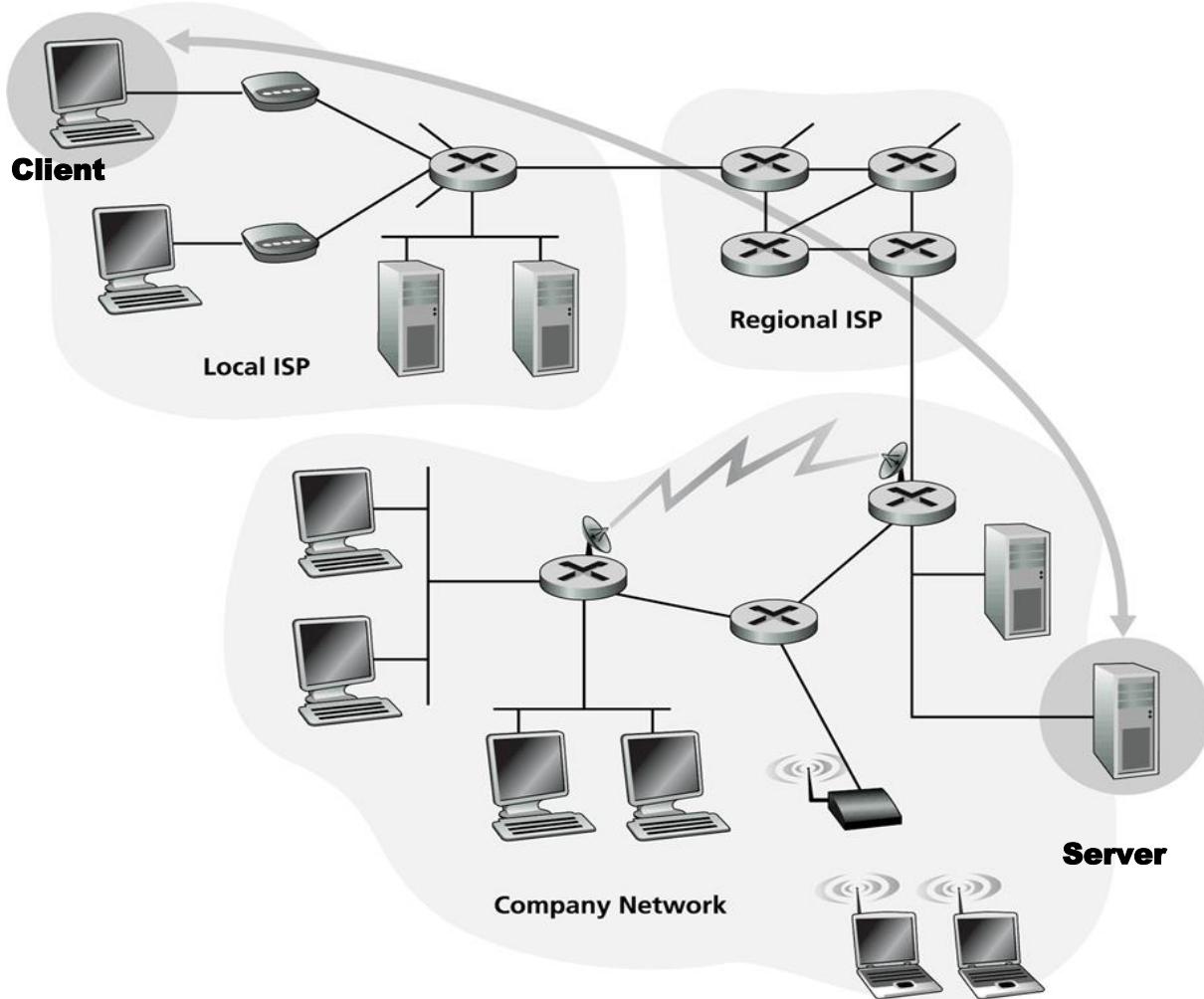
In de praktijk is tegenwoordig het wireless netwerk meestal gescheiden van het bedraad netwerk, en dus gekoppeld door een router, een L3 WAP. Ook al omwille van de verschillen in securiy-eisen.

<sup>1</sup> Het is mogelijk dat een 'wireless router' de functie van WAP en router combineert, maar functioneel zijn dit aparte bouwstenen.

Samengesteld kunnen we stellen dat home users vrij veel met een enkelvoudige computer aansluiten op een locale ISP, terwijl bedrijven aansluiten op snellere ISP's.

Ook het netwerk in een bedrijf is meestal opgedeeld in subnetwerken. Tussen elk (sub)netwerk wordt een router geplaatst.

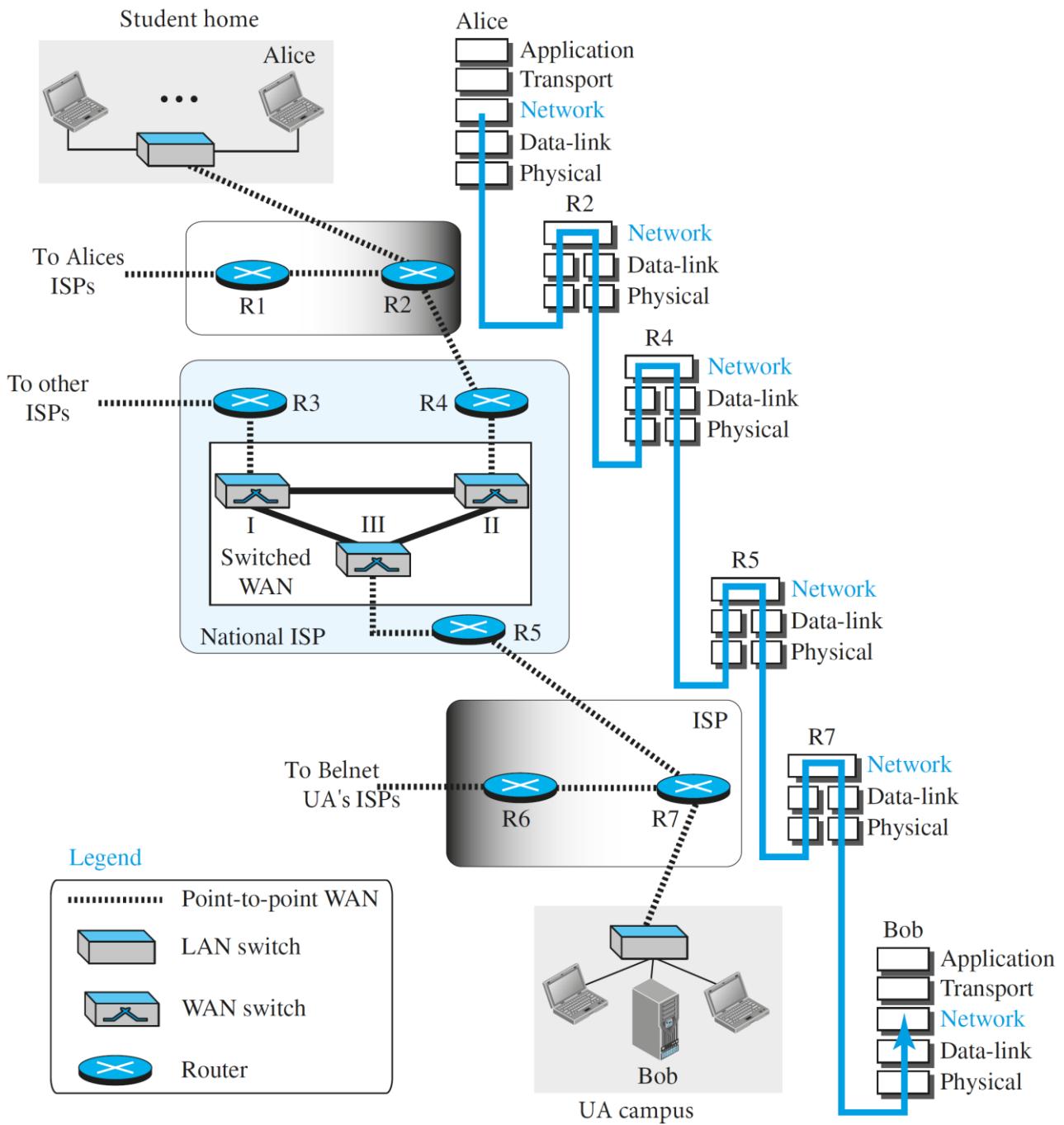
Sommige verbindingen zijn dubbel uitgevoerd ('redundant'): mogelijk om capaciteitsredenen, maar vooral om de betrouwbaarheid te verhogen.



Figuur 260 Totaalbeeld van een internetverbinding.

Hierboven zien we een thuisgebruiker op de lokale ISP, Internet Service Provider, die communiceert met een server van het bedrijf. Deze server is in de praktijk een computer waarop een 'server-programma' draait zoals bv. 'apache webserver'. Het kan ook een PLC, IP-camera, chemisch meettoestel,... zijn die van op afstand bestuurd worden.

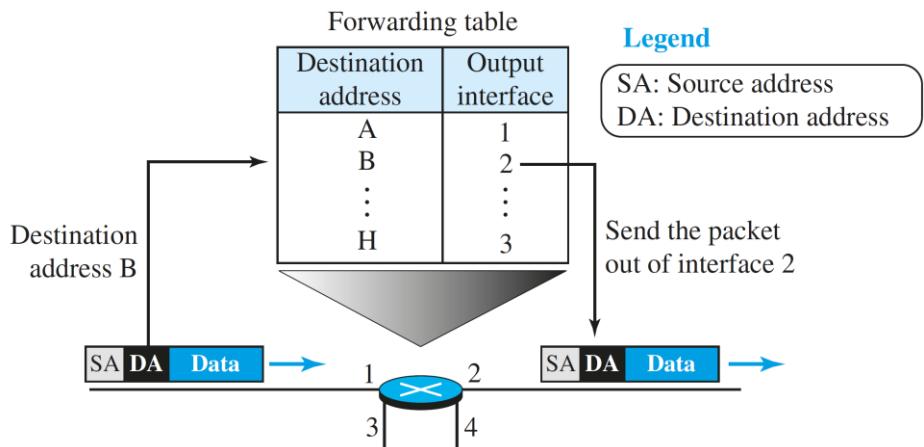
Het doel van internettwerken is een universele dienst te leveren via heterogene netwerken met mogelijk verschillende frameformaten en (laag2-) adresseringssystemen. Het netwerk bekommt zich niet om de inhoud van de boodschap, maar om het foutloos, **wereldwijd** verzenden ervan.



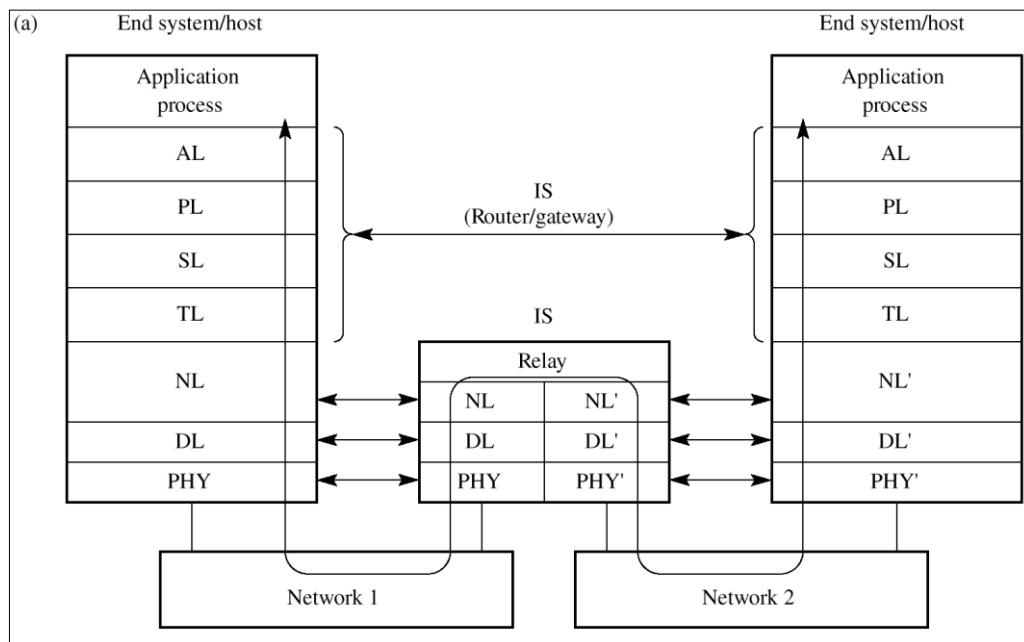
Figuur 261. Routering over ISPs

Tussen elk netwerk gebeurt de koppeling met **routers** die een **L3** forwarding uitvoeren op basis van het IP adres.

Figuur 262. Router forwarding.



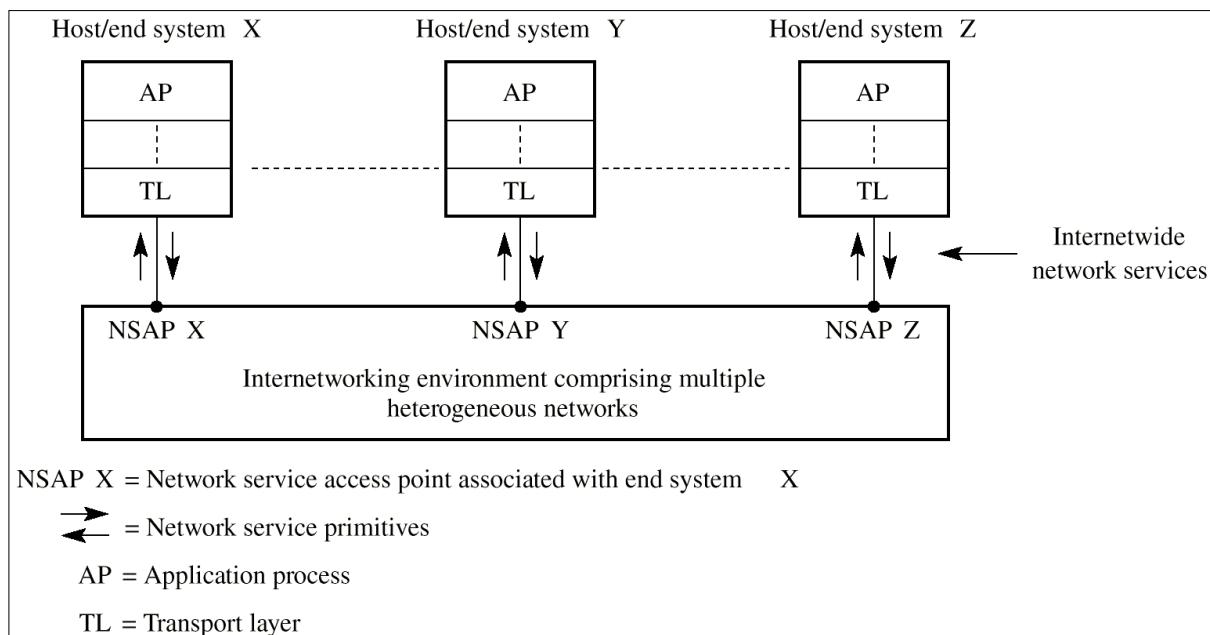
Volgens het OSI model (zie Figuur 13 op p.28) kan men de functie van deze **router/GW** dus situeren als interface t/m laag3. Als de hosts of 'End Systems' (ES) **eenzelfde protocol stack** gebruiken, bvb. TCP/IP, dan volstaat het voor het IS (Intermediate System) om een routerings- (=relay-) functie uit te voeren op laag 3, de **netwerklaag**. Vandaar dat hij dikwijls 'router' wordt genoemd.



Figuur 263 Intermediate Systems : a) router, b) protocol converter.

### 3.1.2 ONTWERPASPECTEN VAN DE NETWERKLAAG.

Vanuit het standpunt van de internetgebruiker moet de netwerklaag hem een aantal diensten ter beschikking stellen die hem toelaten te communiceren met gelijkaardige gebruikers op een ‘remote system’. Dat onderliggend hierbij mogelijk verschillende netwerktypes dienst doen als transport-middel moet voor die gebruiker volledig transparant zijn : de netwerklaag levert in de interface netwerk/transportlaag diensten aan de transportlaag.



Figuur 264 Internetwerk services

Begin- en eindbestemming worden aangegeven door hun resp. **NSAP** (Network Service Access Point), wat voor het Internet overeenkomt met het **IP-adres**. Ze volgen een uniform schema zowel in LAN's als WAN's.

Een netwerklaag-ontwerp moet verschillende topics beschouwen : netwerk service, adressering, routering, QOS (Quality Of Service), maximum pakket size, flow- en congestion controle en foutmeldingen.

### TYPE NETWERK SERVICE.

Herinner dat in een **LAN** de datalink LLC een **CL** (ConnectionLess) service verkiest (type 1) om zijn frames te verzenden aangezien de BER zeer laag is en de vertragingen kort zijn.

**WAN's** daarentegen zullen veel opteren voor een **CO** (Connection Oriented) service aangezien de BER daar beduidend slechter is en de vertragingen groter.

De NS\_users (Netwerk Service-gebruikers), i.e. de transportlagen (bv. TCP, UDP), kunnen mogelijk communiceren over LAN's én WAN's (wat voor hen transparant is). → Er moet eerst een keuze gemaakt worden welke service zal gebruikt worden en hoe deze service kan in overeenstemming gebracht worden met de verschillende onderliggende netwerken.

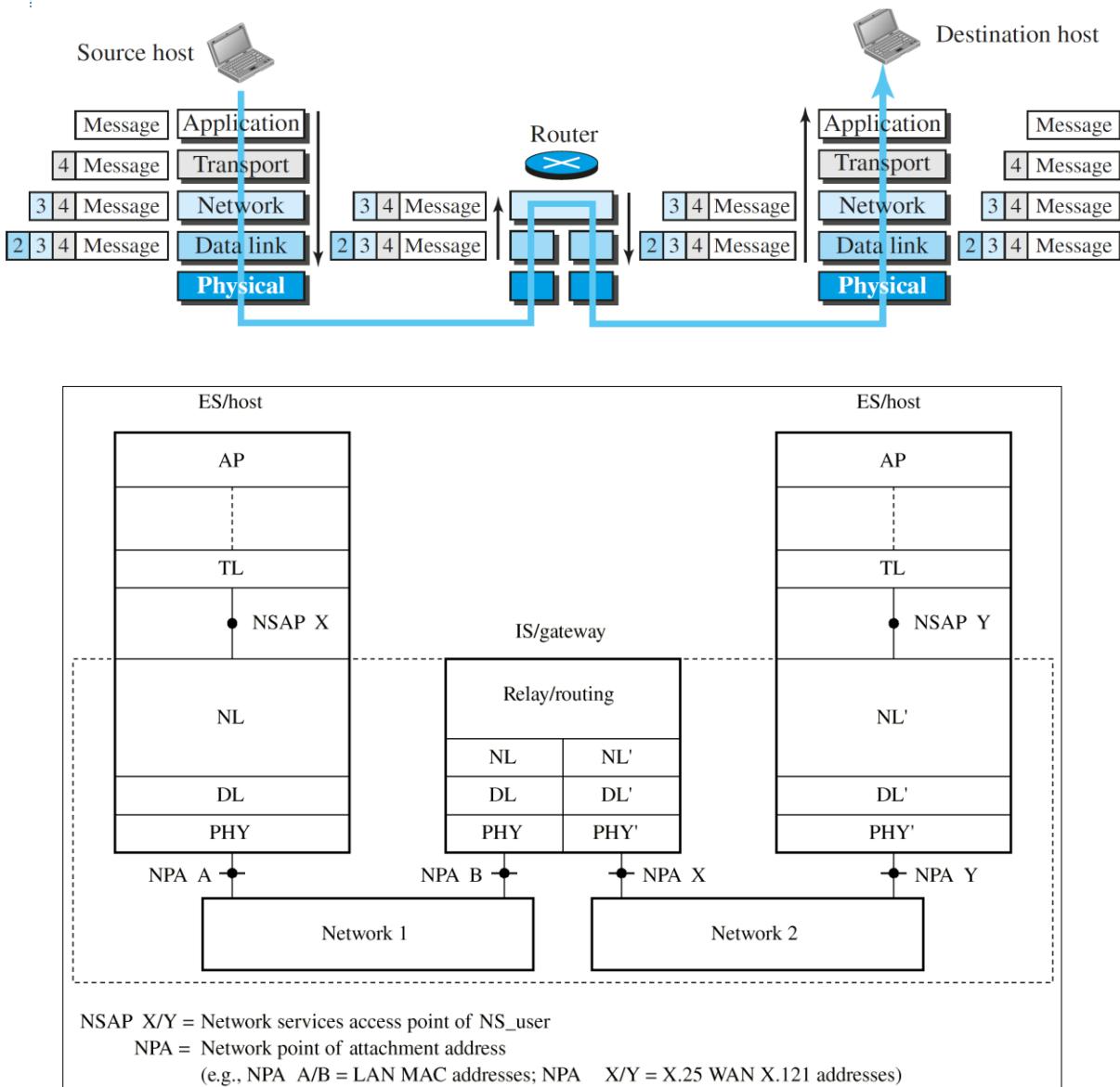
Voor **Internet** zal dit **CL** zijn wat leidt tot primitieven als ZEND PAKKET en ONTVANG PAKKET en weinig meer. Het end system **ES** moet er dus van uit gaan dat het subnet **onbetrouwbaar** is en **zelf foutenafhandeling** en **stroomregulering** verzorgen → 'error- en flow-control' in **laag 4**. De netwerklaag bekommert zich niet om deze problemen omdat er weinig wordt gewonnen als dit 2 keer gebeurt.

Het komt er eigenlijk op neer dat de complexiteit, in geval van **CLNS** (Connection-Less-Network-Service), in de **hosts** wordt gestopt, i.p.v. in het *netwerk* zoals bij **CONS**. De filosofie hierachter is dat rekenkracht voor gebruikers zo goedkoop is geworden dat dit financieel interessanter is dan het netwerk ermee te belasten. Bovendien hebben sommige soorten verkeer meer baat bij **snelle** dan bij **accurate** aflevering. Zie ook QoS (Quality of Service).

Aangezien, CL, er niet eerst een verbinding wordt opgezet, moeten **alle** pakketten het **netwerkadres** (IP) van de bestemming bevatten aangezien ze allen een andere weg door het netwerk kunnen volgen, m.a.w. onafhankelijk van elkaar kunnen worden gerouteerd. Aangezien IP een **CL** protocol is zullen pakketten opgestuurd worden in een **unacknowledged – best try** – mode.

In dit hoofdstuk zal vooral IP aan bod komen, een **CLNS**. Voor CONS we verwijzen naar de cursus 5 infrastructuur waar we zullen zien dat een dergelijke opzet mogelijk ook voordelen biedt, bv. bij MPLS.

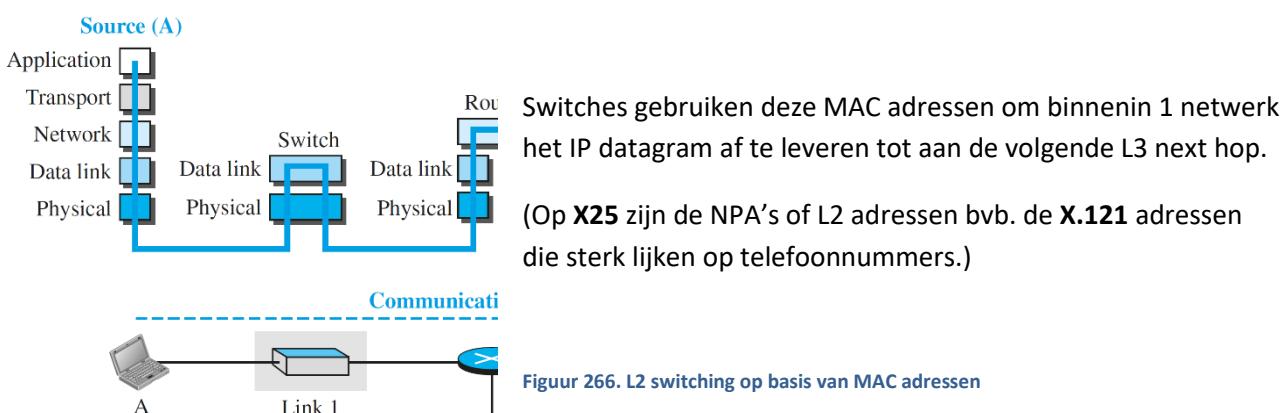
## ADRESSEERING



Figuur 265 Verband tussen NSAP en NPA.

Het **NSAP**-adres (vb. **IP**) wordt gebruikt om een **NS\_user** (vb. **TL** = **TCP**) in een **ES** te identificeren, wereldwijd. L3 routers of gateways gebruiken enkel dit IP adres om hun weg te vinden.

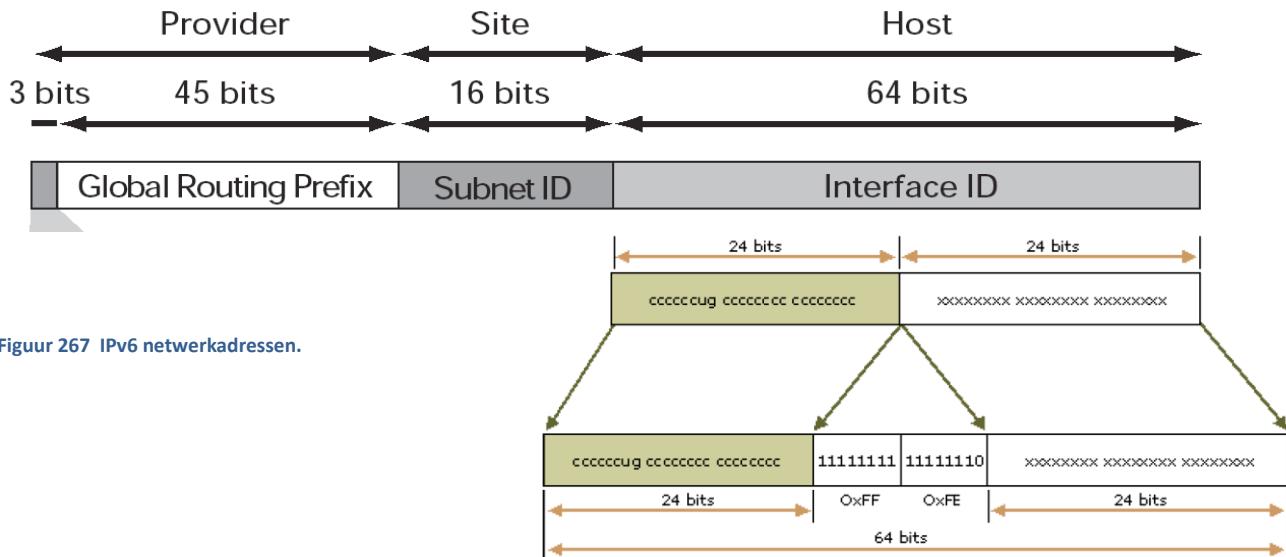
Binnenin 1 LAN, vb. ethernet, kennen we ook reeds de **MAC**-adressen die op de **datalink** gebruikt worden om elke NIC te identificeren. Algemeen worden dit **NPA's** genoemd, Network Point of Attachment.



Figuur 266. L2 switching op basis van MAC adressen

In een internet, bestaande uit meerdere netwerken van een verschillend type, kunnen deze NPA's niet gebruikt worden als NSAP omdat ze **niet universeel** zijn (MAC <> X.121). Daarom wordt als NSAP een volledig ander adresseringssysteem gebruikt, onafhankelijk van de NPA's. De **NPA's** blijven dienen als **datalinkadres** binnenin 1 netwerksegment, de **NSAP's** als **wereldwijd** unieke 'identifier' van een NS\_user in een ES. Er zijn dan processen die de NSAP's vertalen in NPA, bvb. **ARP** op ethernet.

OPM. IPv6 kan een techniek om het 48 bit MAC (L2) adres te verwerken in het onderste deel van zijn IP adres. Hostid of Interfaceid is hiervan afgeleid : '**FF FE**' er temidden in...



Figuur 267 IPv6 netwerkadressen.

Aangezien dit echter geen must is, en IPv6 een volledig andere hostid kan gebruiken, blijven we (nog even) bij de situatie van IPv4 waar er dus **geen enkel verband** is tussen **IP** en **MAC** adressen.

### ROUTERING.

Aangezien elk (IP-)pakket door het netwerk zijn weg moet zoeken moeten de IS (maar ook de ES) een tabel opbouwen om binnenvkomende pakketten te 'forwarden' naar de 'next hop'. Aangezien dit mechanisme iets te ingewikkeld is om in deze inleiding uiteen te zetten verwijzen we naar pt. 3.4 IP routing.

### QUALITY OF SERVICE. QOS

Algemeen zal bij elke service request primitive een parameter meegegeven worden die de QoS specificeert, m.a.w. welke performantie de NS\_user verwacht van het netwerk. Bvb. zijn dat de verwachte vertraging of 'transit delay', de protectie tegen afluisteren, de maximale kost, kans op fouten, ...

Voor een **CONS** wordt bij het opzetten van de call tussen beide systemen **overlegd** over de te gebruiken QoS. De nodes in het netwerk moeten deze afspraken ook respecteren.

Bij **CLNS**, de weg van de datagrammen ligt **niet** vast, zal de NS\_user een aantal QoS parameters **aannemen**. Voor internetten waar pakketten verschillende types netwerken passeren moet het ES kennis verwerven over de te verwachten QoS naar een specifieke bestemming.

Deze parameters zijn normaal afh. v. het **soort dataverkeer** dat de netwerklaag zal versturen :

1. **Interactief**,: vb. tussen server en user : de gebruiker verwacht een **snelle respons** →  $t_{acc}$ . Dit stelt hoge eisen aan het dataverkeer. Het wordt soms geclasseerd als **synchroon** verkeer, (→ directe interactie tussen zender en ontvanger) en kan ‘bursty’ zijn.
2. **File-transfer** : is vooral ‘bursty’, maar niet noodzakelijk met interactie van gebruiker, bv. een backup →  $t_{acc} \pm$ . Het type verkeer is **±synchroon**.
3. **E-mail**: store & forward, geen directe interactie met gebruiker, **lage eisen** aan het dataverkeer. → **asynchroon**
4. **Spraak en video** : vereist korte maar vooral **konstante tijdsvertragingen** → **isochroon**. Hoge én speciale eisen aan het dataverkeer :  $t_{acc} < 200ms$  en jitter hierop  $< 100ms$  voor spraak.

## FLOW EN CONGESTION CONTROL

Ook een netwerk in zijn geheel kan lijden onder congestions met name als er meer pakketten in het netwerk toekomen dan dat er vertrekken. Concreter : als een router meer pakketten toekrijgt dan dat hij kan verwerken.

Voor een **CONS**, bvb. X25, zal bij het opzetten van het VC (virtueel circuit) overeengekomen worden wat de ‘window size’ is (typisch 2) en moet de zender wachten op ‘ACK’s alvorens verder te gaan. Dit controleert weliswaar congestie, maar voorkomt het niet.

Voor **CLNS** zijn er geen afspraken op L3 en moet de **transportlaag (L4, TCP)** in de ESs **flow control** toepassen op een ‘**end to end**’ basis : ze moeten stoppen met zenden bij het optreden van congestie.

## FOUTMELDINGEN

Het type fouten die optreden in verschillende netwerken is soms verschillend. Er moet dus een manier gezocht worden om deze meldingen over het internet te transporteren. Typisch gebeurt dit door **ICMP**.

### MAXIMUM PACKET SIZE.

We kennen reeds een aantal **MTU's**, Maximum Transfert Units, van verschillende datalinks met als gevolg de **maximum pakketlengte** die kan worden verpakt (- header en CRC) :

- *ethernet → 1500 bytes* (1518 – 18)
- *IEEE 802.3-802.2-SNAP → 1492* (1518 – 18 – 8)
- *Token ring IEEE 802.5 → 5000*
- *SLIP niet gedefinieerd, normaal tot 1006, enkele IP*
- *PPP → 1500 bytes*

Ze zijn allen (redelijk) verschillend, bepaald door de volgende factoren :

- *BER* : hoe hoger de BER, hoe kleiner de MTU, om een redelijke kans te maken dat een pakket foutvrij aankomt.
- *Transit delay* : hoe langer de max. pakketgrootte, des te langer moeten andere pakketten wachten op een link om te worden doorgestuurd.
- *Buffers*: kleinere max. pakketgrootte reduceert de te reserveren geheugenbuffers
- *Processing overhead* : een kleinere pakketgrootte leidt tot **meer** overhead in het fragmenteren van de totale boodschap.

In één en hetzelfde netwerk is de MTU, en dus de max. pakketgrootte, gekend en zal de **transportlaag** de boodschap opdelen – **segmenteren** – in de ideale grootte.

Over een internet bestaande uit **verschillende netwerken** (en ~MTU's) zal **ofwel** een kleinere, niet ideale lengte, bvb **576<sup>1</sup>** bytes IP, gebruikt worden, **ofwel** wordt de '**Path MTU**'<sup>2</sup> opgezocht, **ofwel** wordt de MTU van de 1<sup>e</sup> link ('naar de 1<sup>e</sup> hop') gebruikt en moet de netwerklaag zonodig **fragmenteren**.

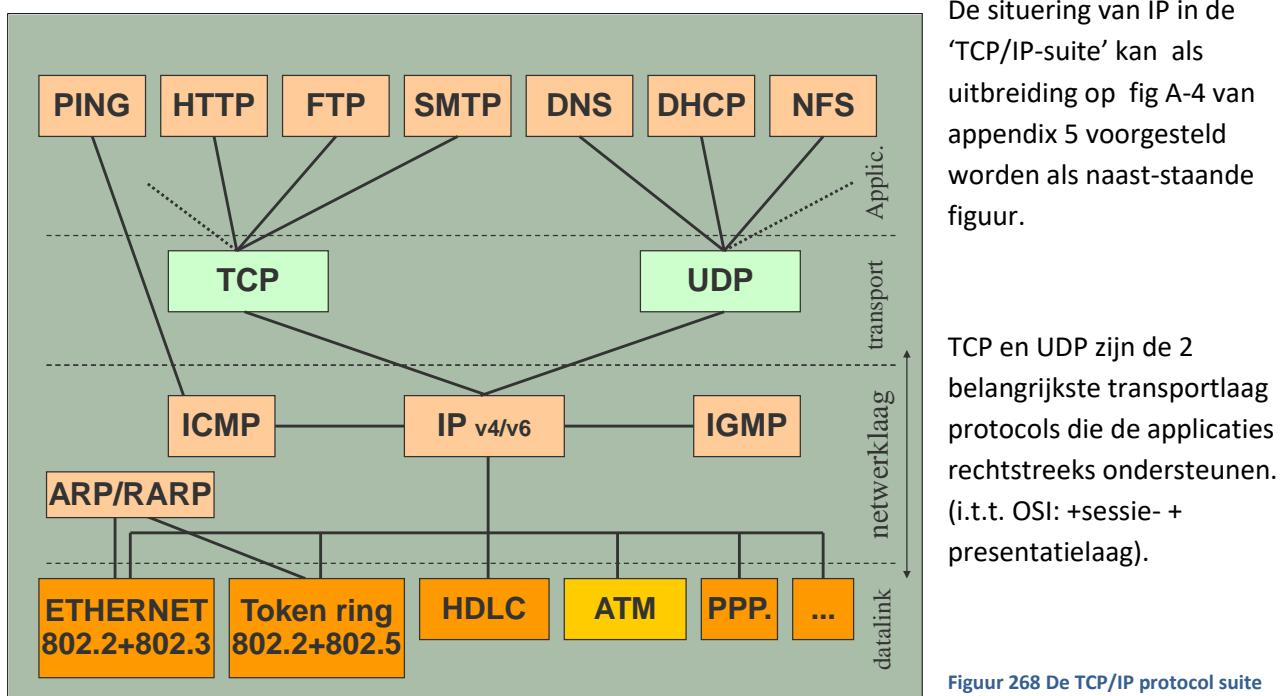
*(Dit laatste is eerder theoretisch en out-dated. Routers worden vandaag de dag zo geconfigureerd dat ze geen fragmentering meer uitvoeren. Indien ze een link tegenkomen die een te kleine MTU heeft voor een bepaald IP pakket, droppen ze het pakket. De hoger gelegen laag, TCP moet dan maar de steken oprapen.)*

<sup>1</sup> Dit is voor IPv4 de **minimale MTU** die netwerken moeten hebben. Een **576** byte IP pakket moet gegarandeerd ongefragmenteerd kunnen vervoerd worden over alle links. Voor IPv6 is dit trouwens opgetrokken tot 1280 bytes.

<sup>2</sup> Dit gebeurt door een pakket op te sturen dat niet mag gefragmenteerd worden en zien of het aankomt (door TCP).

### 3.1.3 HET TCP/IP MODEL

Aangezien hét net steunt op het IP protocol is dit model automatisch de **standaard** geworden i.t.t. OSI.



Figuur 268 De TCP/IP protocol suite

Beiden gebruiken **IP** als **netwerklaag**, onafhankelijk of de onderliggende laag een LAN, WAN of een internetwerk is. (Zeldzaam maar toegestaan is dat een applicatie IP rechtstreeks aanstuurt (vb. traceroute, niet getekend). IP kan v4 of v6 zijn, er is geen wijziging nodig voor de omliggende protocols..., het enige minieme verschil hierbij is dat IGMP v6 zal opgenomen worden in ICMPv6).

**TCP** voorziet een '**reliable**', **CO**, transport ook al zijn de services van **IP** '**unreliable**'. **UDP** daarentegen voorziet '**unreliable**', 'best try' verkeer. Uiteraard eisen niet alle applicaties betrouwbare verbindingen : bij video- of audio- informatie is het niet zo erg dat er al eens een fout optreedt, de snelheid daarentegen is wel belangrijk → '**isochroon**'. Alhoewel de BER in LAN's zeer laag ligt, is ze niet NUL. Daarom mag UDP enkel gebruikt worden voor applicaties die af en toe een fout toelaten. In alle andere gevallen wordt TCP (= CO) gebruikt.

IP zelf zal zijn datagrammen CL, best try, verzenden wat de ballast of 'overhead' minimaliseert aangezien er niet steeds een verbinding moet opgezet worden. IP voert geen error-controle op de data uit.

CL → betekent dat in **elk** IP-datagram het **adres** van bestemming en afzender vervat zit. Dit wordt ook gebruikt om tussenliggende routers de juiste weg te laten kiezen.

**ICMP** (Internet Control Message Protocol) is een toevoegsel aan IP. Het wordt gebruikt door IP om **foutberichten** en andere vitale informatie door te geven. (Vb. bij ontvangst van een fout IP-datagram, hosts (of netwerken) die niet kunnen bereikt worden, ... zal de IP laag **proberen** (best try) de zender op de hoogte te brengen,) Ping en traceroute, 2 populaire diagnose-tools gebruiken ICMP.

**IGMP** (Internet Group Management Protocol) wordt gebruikt bij multicasting om een UDP datagram naar verschillende gebruikers te zenden.

ARP (Address Resolution Protocol) en RARP (Reverse ARP) zijn speciale protocols die gebruikt worden bij sommige netwerkinterfaces (zoals ethernet en token ring) voor de conversie van IP adressen naar MAC adressen (en omgekeerd).

De applicaties uit de figuur zijn :

- **FTP** : File Transfer Protocol, laat gebruikers bestanden uitwisselen
- **HTTP** : Hypertext Transfer Protocol, wat gebruikt wordt voor webservices.
- **Telnet** : Terminal emulation, andere computers kunnen interactief een taak openen
- **NFS** : Network File System, laat toe om het file system uit te breiden naar andere computers
- **SMTP** : Simple Mail Transfer Protocol
- **DNS** : Domain Name System (of Server), om internetnamen om te zetten naar IP adressen.
- **DHCP** : geeft een host zijn IP configuratie van een DHCP-server
- **Ping** : onderzoekt of een host bereikbaar is of niet door een ICMP pakket te zenden en er een terug te verwachten.

De applicaties gebruiken de aangeboden '*protocol stack*'. Het transport verloopt als volgt :

#### Zender

- De applicatie geeft de data aan de transport layer (pointer van mem. Buffer, + IP-adres)
- TCP (bv.) verdeelt de data in TCP-segmenten, voegt een header toe met een **TCP-volnummer** en een '**port**' i.f.v. de applicatie (=SAP), berekent op header en data een checksum, en speelt dit door aan IP
- IP creëert een datagram met als **data** het TCP segment, voegt zelf een IP-header toe met source en dest. IP-adres. IP bepaalt bovendien het fysische (MAC) adres van de volgende node (via ARP) en geeft dit door aan de datalink-laag.
- Datalink verpakt het IP pakket in het datagedeelte van een frame en verzendt het naar de gekregen bestemming (MAC).

#### Ontvanger

- De datalink controleert zijn CRC. OK → in header staat het typeveld (zie Verschil tussen ethernet en IEEE 802.3 frames p. 201) → = IP ? Ja → datagedeelte naar IP-laag.
- IP laag controleert IP-datagram-header en checksum. ∀ OK → protocol ? = TCP → datagedeelte naar TCP-laag.
- TCP berekent de checksum van het TCP segment data en header. OK → zendt een ACKnowledge terug naar de zender (**CO!**). TCP verzamelt alle segmenten op basis van de volgnummers, error- en flowcontrole, en speelt de data door via de 'port' aan de overeenkomstige applicatie.

Een uitgebreidere verwerking is te vinden in de Appendices, 5.2.2 overzicht TCP/IP over EtherNet op p. 353.

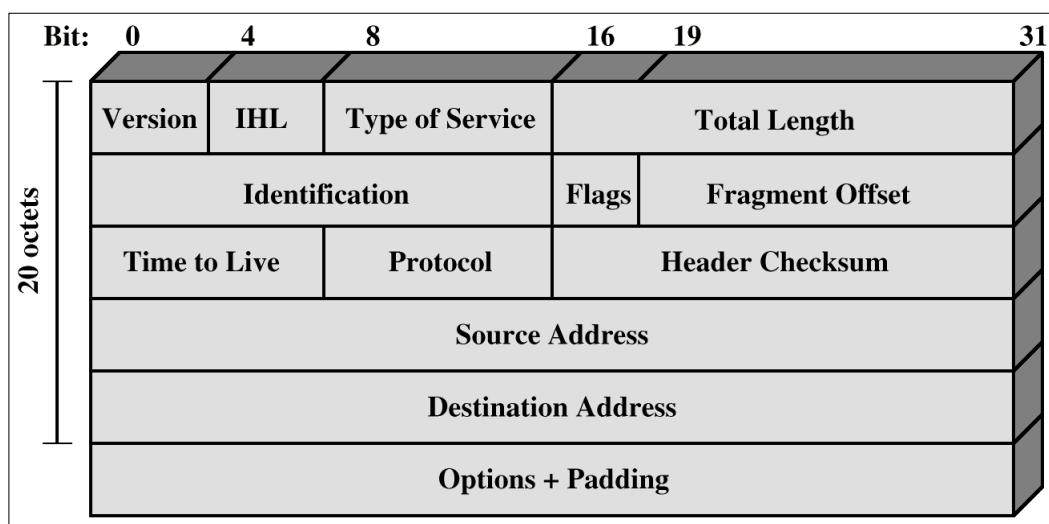
## 3.2 HET IP DATAGRAM

Dit aspect is reeds uitgebreid besproken in een voorgaande cursus: **2-computernetwerken**. Het wordt hier enkel herhaald voor de volledigheid, vanaf deze pagina tot p. 256.

**NIEUW (tov 2CN)** is pt. 3.3.5 IP VLSM subnet – adressering en 3.3.6 CIDR Classless InterDomain Routing

IP pakketten die op deze netwerken worden getransporteerd hebben allemaal een header meegekregen van het verzendende station (source). Dit dient om de werking van IP te verzekeren door bv. de routers toe te laten om te bepalen waarheen dit pakket moet worden verzonden (destination).

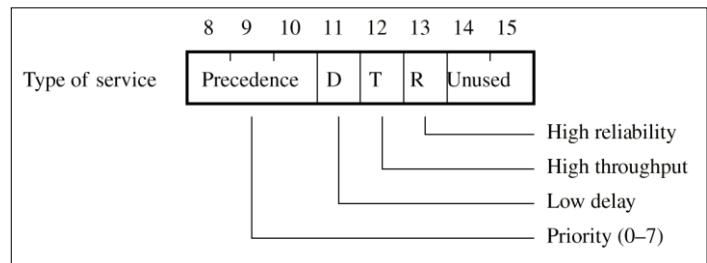
Een keer een pakket aangekomen is bij zijn **bestemming** kan deze laatste de 2 adressen omdraaien om een antwoord te formuleren.



Figuur 269 Het IP datagram formaat.

De header bestaat uit veelvouden van **32bit** woorden, normaal 5, m.a.w. een **20-byte** header.

- Het ‘**version**’ veld bevat de IP versie, hier vb. 4 (0010), en dient om de nodes toe te laten de juiste **indeling** van het datagram te interpreteren. Het zal ook dienen om later IPv6 te herkennen (§ 6).
- De ‘**Header length**’ is normaal (en minimaal) dus **5**, maar er kunnen opties (vb. *record route*) worden meegegeven in veelvouden van 32 bit. Maximaal is de header dus 15 velden (v. 32bit) lang § 60 byte.
- **Type of Services (TOS)** vervult een QoS functie. Het specificeert de gewenste voorkeuren bij het behandelen van dit datagram.
  - De **precedence** bits geven een **prioriteit** aan waarmee het pakket dient behandeld te worden : 3 bits waarmee we in totaal 8 niveau’s kunnen instellen. Routers kunnen dan voorrang geven aan pakketten met een hoger niveau.



Figuur 270 TOS bits in een IP header.

De andere bits definiëren **specifieke service**-vragen.

- D-bit:= Low **Delay**. Typisch gevraagd voor voice (en video) trafiek.
- T-bit: = High **Throughput**. Bv. voor **FTP-data**: (FTP = File transfert Protocol → moet snel gaan  
(Veel minder/niet gebruikt zijn de andere bits)

Routing protocollen zoals OSPF en IS-IS gebruiken deze bits om de beste weg te kiezen, maar steeds minder vanwege het compliceren van de beslissing-algoritmes.

Dit is de allereerste definitie van de TOS byte. Moderner is de nieuwe versie hiervan: diffserv code point, of DSCP. Het heeft dezelfde functie, namelijk het gedifferentieerd behandelen van IP pakketten, maar hiervoor verwijzen we naar een volgend vak: 5-infrastructuur.

- De **'total length'** definieert de totale lengte van het datagram, incl. header. De maximum lengte is 65535 bytes (16 bit veld), meestal beperkt men zich tot de MTU van de datalink, bvb. 1500 bytes voor ethernet om fragmentering te voorkomen. Kent men de MTU niet, dan gebruikt men een default internet MTU = 576 byte als IP pakket.
- Het **identificatie** veld : IP kan grote datagrammen fragmenteren als het een datalink tegenkomt die een MTU heeft, kleiner dan het oorspronkelijk datagram: IP zal grote datagrammen fragmenteren als het een datalink tegenkomt die een MTU heeft, kleiner dan het oorspronkelijk datagram zie ook pt. 3.5.4 op p.286. Om nu het oorspronkelijk datagram terug te kunnen reassembleren zal IP alle datagram-fragmenten die gegenereerd worden eenzelfde identificatieveld geven. De fragmenten (met hetzelfde identificatieveld) worden samengevoegd in volgorde volgens het fragment offset veld, dat de rel. positie aangeeft van het fragment in het datagram.  
Het spreekt voor zichzelf dat IP het identificatieveld zal incrementeren voor opeenvolgende, nieuw gegenereerde, datagrammen zodat dit veld uniek is per datagram.)
- Het **Flag** veld bevat 3 vlaggen waarvan er 2 worden gebruikt :

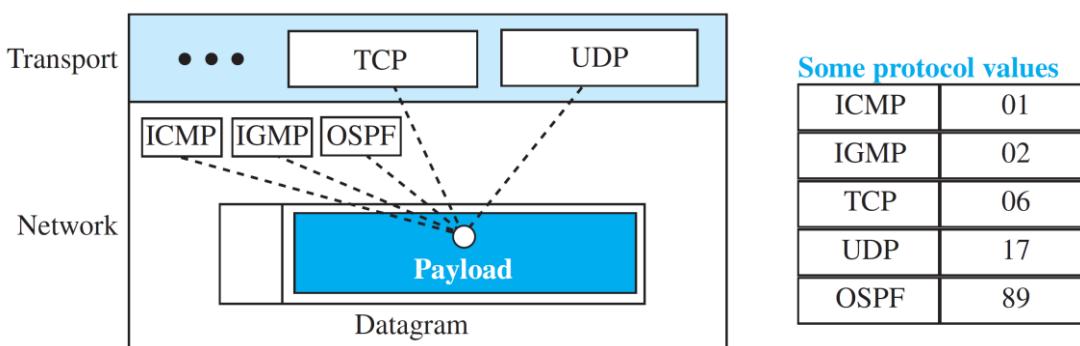
D	M	Fragm. of
---	---	-----------

- D = 1 → **Don't fragment** : een router moet dan een pad kiezen waar het volledige datagram kan passeren, zoniet verwerpt hij het. (D=0 → fragmenteren kan).
- M = more fragments : bij gefragmenteerde boodschappen hebben **alle** datagrammen M=1 (→ 'meer fragmenten' komen), behalve het **laatste** datagram M=0.
- **Time To Live (TTL)** begrenst het aantal routers dat een datagram kan passeren. M.a.w. het beperkt de **leeftijd** van een datagram. Het wordt geïnitialiseerd door de zender op een bepaalde waarde (bv. 64, afh. van het OS) en wordt gedecrementeerd in elke router. Bereikt dit veld = 0, dan wordt het datagram geëlimineerd en de zender wordt op de hoogte gebracht met een ICMP bericht.

Deze procedure verhindert datagrammen om oneindige lussen te maken in het netwerk. In theorie wordt de TTL uitgedrukt in seconden, maar dat betekent dat routers een 'clocking' mechanisme moeten hebben om de tijd te meten tussen 2 hops. Praktisch zullen routers de TTL verlagen met 1 → **TTL = 'hop count'**.

In IPv6 is dit ook meteen vervangen door de term 'hop limit'.

- Het **protocol**-veld laat IP toe om het datagram door te geven aan de gepaste hoger liggende laag, de **NLPID** (Next Layer Protocol Identifier), zie ook Figuur 268. Hieronder de voornaamste voorbeelden.



Figuur 271. IP protocol byte.

IANA, de internet autoriteit heeft elk protocol dat de IP stack gebruikt een uniek nummer toegewezen, 8bit maximaal 256.

- De **Header checksum** dient om evt. transmissiefouten op te sporen, maar wordt alleen berekend op het **IP-header** gedeelte. Fouten die zouden optreden in de header kunnen leiden tot 'misinsertions' : **IP<sub>D</sub>** : toekomen bij een verkeerde host, **IP<sub>S</sub>** : terugzenden naar een verkeerde host, **protocol** : aflevering aan verkeerd protocol, ... . Dat moet worden voorkomen

Het belang hiervan neemt af: er treden steeds minder fouten op en bijgevolg voorziet IPv6 dit veld zelfs niet meer.

De checksum wordt berekend door alle **16-bit** woorden van de header in 1CC op te tellen en hiervan het inverse te nemen. Een ontvanger moet dan als checksum 0xFF FF uitkomen, zoniet verwerpt hij het datagram. Een router die meestal alleen het TTL veld aanpast ( $\rightarrow -1$ ) kan dan de checksum incrementeren i.p.v. te herrekenen.

4	5	0	28	
49.153		0	0	
4	17	0		
10.12.14.5				
12.6.7.9				
4, 5, and 0	→	4	5	0 0
28	→	0	0	1 C
49.153	→	C	0	0 1
0 and 0	→	0	0	0 0
4 and 17	→	0	4	1 1
0	→	0	0	0 0
10.12	→	0	A	0 C
14.5	→	0	E	0 5
12.6	→	0	C	0 6
7.9	→	0	7	0 9
Sum	→	1	3	4 4 E
Wrapped sum	→	3	4	4 F
Checksum	→	C	B	B 0

Voorbeeld van een IP checksum berekening:

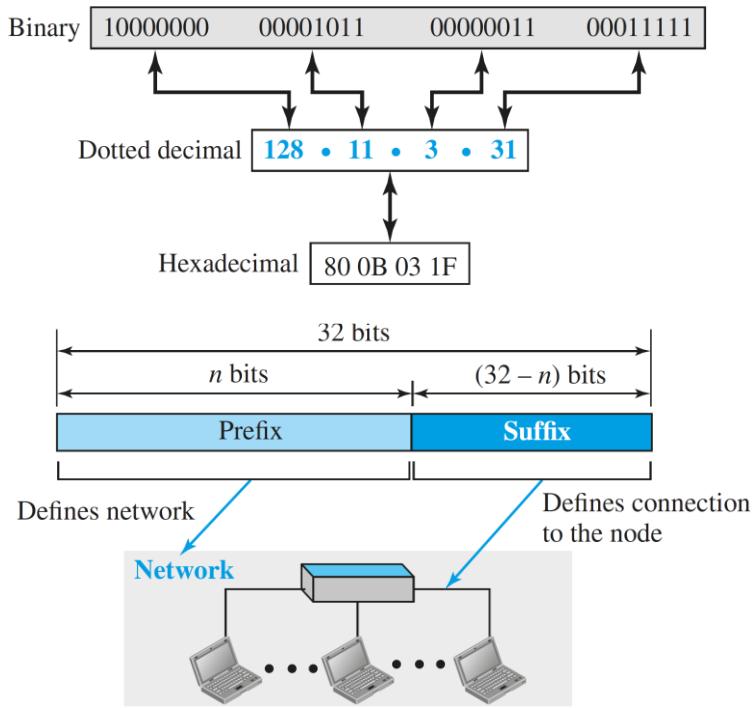
- Alle Velden worden in 16 bit opgeteld
- Bij de initiator is het **checksum** veld = 0
- + Modulo FFFF  $\rightarrow$  carry's vervallen
- De wrapped sum wordt gecomplementeerd  
Bv. 3 = 0011  $\rightarrow$  C = 1100
- Bij de receiver: SOM inbegrepen **checksum**  $\rightarrow$  Moet = FFFF zijn.

Figuur 271. Een 16-bit checksum berekening, bv. van de IP header.

(TCP, UDP, ICMP en IGMP hebben elk hun eigen checksum, berekend op **hun** header én **data**, op dezelfde manier als hierboven beschreven.)

- **Source en destination adres** zijn de **32-bit** internet-wijde IP adressen van zender en ontvanger. Ze definiëren de 2 eindpunten van de communicatie, wereldwijd uniek. Zie ook pt. 3.3. Tot hier gaat de normale IP-header..

Notatie: meestal/altijd “dotted-decimal notation” en bestaat uit 2 delen: **netid – hostid. (prefix – suffix)**.



Figuur 272. IP adressen notatie en indeling.

Tot hier gaat de normale IP-header.

#### UITBREIDING, NIET STRICT NOODZAKELIJK.

Zeldzaam wordt de header uitgebreid met (32 bit) **options**-velden, aangegeven in de header lengte ( $\oplus \neq 5$ ). Mogelijke options zijn :

- *Security* : het dataveld kan bv. geëncrypteerd zijn, of enkel voor specifieke gebruikers.
- *Source routing* : een opsomming van IP-adressen die het datagram moet passeren.
- *Route recording* : elke gepasseerde router moet zijn IP-adres noteren, kan dienen voor ↗.
- *Timestamp* : (idem als ↑) elke gepasseerde router moet zijn IP-adres én tijdstip noteren
- *Stream identification* : hier kan het type data aangegeven worden, vb video, audio, data, ...

Aangezien de header lengte (4 bit) max. 15 woorden (van 32 bit) kan bevatten waarvan er al 5 vast liggen hebben we er nog 10 over ... ➔ relatief zelden gebruikt.

### 3.3 IP ADRESSEERING

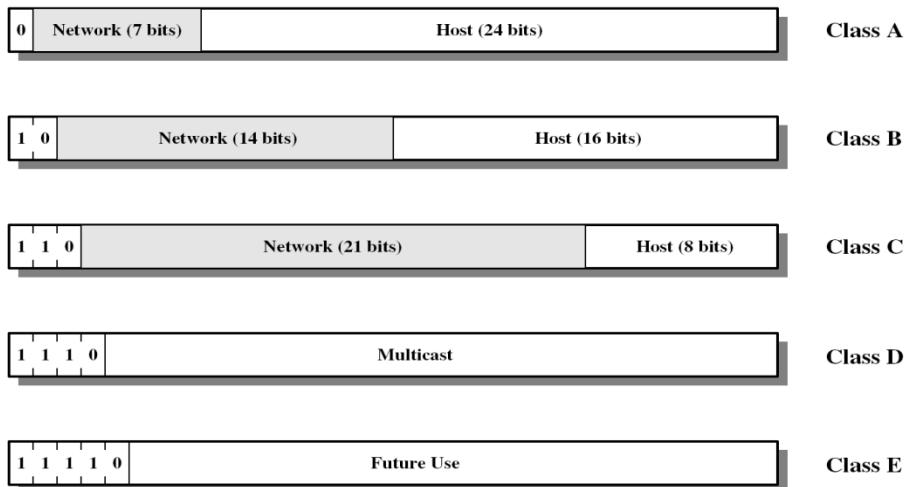
#### 3.3.1 IP ADRES-STRUCTUUR – VOLLE KLASSE ‘CLASS-BASED’.

Dit is de **oorspronkelijke** indeling van IP netwerken waarbij de grens tussen netid en hostid op de bytegrenzen ligt. Het is ondertussen achterhaald door subnet adressering, daarom enkel een kort overzicht.

Elke interface (op hosts én routers) moet wereldwijd een **uniek internet (IP) adres** bezitten als je er mee wil communiceren. Dit IP adres is steeds een **32bit** adres, en bestaat uit **2 delen**:

- een **network identifier (netid)** om het netwerk aan te duiden waartoe dit IP-adres behoort
- en een **host identifier (hostid)** om de host op dit netwerk aan te duiden.

Om te vermijden dat elke router op het net **alle** paden naar **alle** afzonderlijke IP adressen moet opslaan wordt er dus gewerkt met **netwerken** ➔ **routers** kennen alleen de **paden** naar de **netwerken**<sup>1</sup>.



Figuur 273 IP adres klassen

Maar netwerken kunnen veel of weinig hosts bevatten ➔ indeling in grote, middelgrote en kleine netwerken = indeling in 3 ‘klassen’ **netwerken : A, B en C**.

De **klasse** tot dewelke een adres behoort wordt bepaald op basis van de **eerste 0-bit** in de eerste 4 bits

De resterende bits specificeren **2 subvelden**: netid en hostid. De lengte van deze velden is afhankelijk van de klasse. De grens ertussen verschuift op **8bit** (byte)-basis.

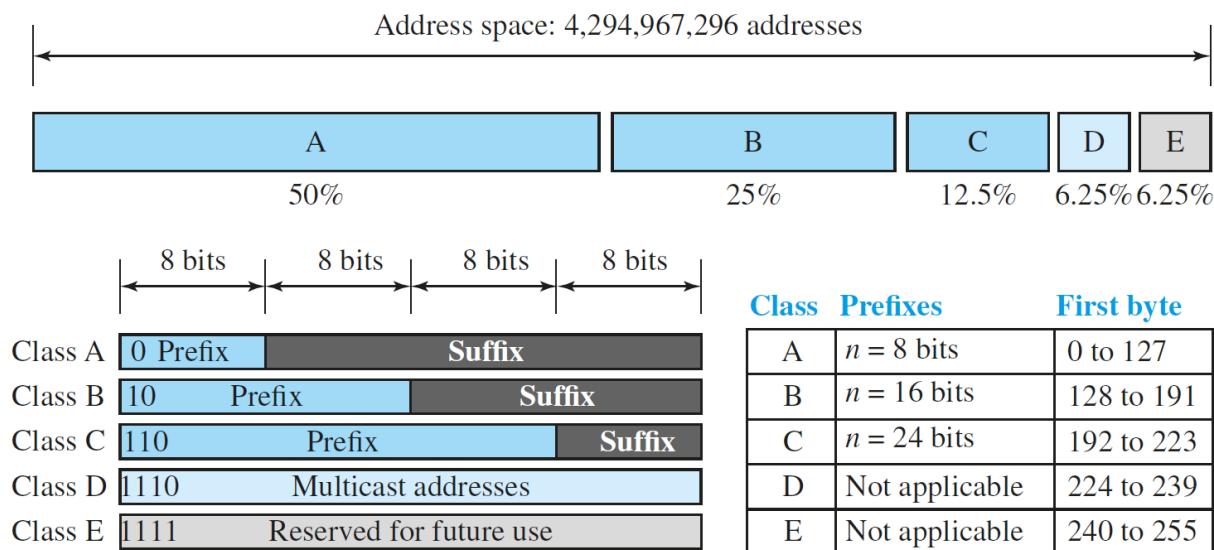
Klasse	netid	# netwerken	hostid	# hosts	netw.	netw.
Klasse A	7 bit	$2^7 = 128$ (-2)	24 bit	$2^{24} = 16\ 777\ 216$ (-2)	1.	126.
Klasse B	14 bit	$2^{14} = 16384$	16 bit	$2^{16} = 65\ 536$ (-2)	128.0	191.255.
Klasse C	21 bit	$2^{21} = 2\ 097\ 152$	8 bit	$2^8 = 256$ (-2)	192.0.0	223.255.25

<sup>1</sup> Merk op dat netwerksegmenten die verbonden zijn d.m.v. een ‘bridge’ of ‘switch’ 1 netwerk vormen. (Bridges en switches koppelen op laag 2, netwerken zijn laag 3)

Voorbeeld: (de netwerken zelf worden herkend door enkel de netid, de hostid =  $\forall 0$ )

- klasse A: netwerk van MIT (18.0.0.0), ZEER grote bedrijven of universiteiten
- klasse B: 156.12.0.0, geschikt voor grote bedrijven met maximaal **65 000** hosts.,.
- klasse C: 195.162.205.0 LAN's tot **254** hosts.

#### INDELING 'ADDRESS SPACE' IN IPV4.



Figuur 274 IPv4 adres space

We zien hier direct de reden waarom deze classfull adressering heeft geleid tot het uitputten van de 4 miljard mogelijke IP adressen: de 128 (126) klasse A adressen namen veel te veel plaats in beslag, 50%, en waren grotendeels ongebruikt: 16 miljoen IP adressen in 1 netwerk... megalomaan!

#### MULTICAST & BROADCAST

Dit zijn adressen die tot doel hebben om een **groep hosts** (interfaces) aan te spreken, in het geval van een **broadcast**: *allemaal* (meestal binnen 1 netwerk, niet *allemaal* op het Internet).

**Klasse D** adressen zijn gereserveerd voor **multicasting** over het gehele internet. (Op ethernet-niveau is een gelijkaardige werking voorzien binnenin één LAN. Ze kunnen gebruikt worden voor het volgen van bv. mediestromen (internet-TV) of door de netwerkapparatuur zelf, bv. routers communiceren met elkaar door bv. OSPF.) Merk ook op dat hier géén indeling is in netid – hostid.

**Klasse A, B of C** adressen met een **hostid =  $\forall 1$**  zijn **broadcasts**, bedoeld om alle hosts **binnenin hetzelfde netwerk (A, B of C)** te bereiken. Vb van voorgaande netwerken zijn de broadcastadressen :

- Kl A : **18.255.255.255**
- Kl B : **156.12.255.255**
- Kl C : **195.162.205.255**

## RESUME

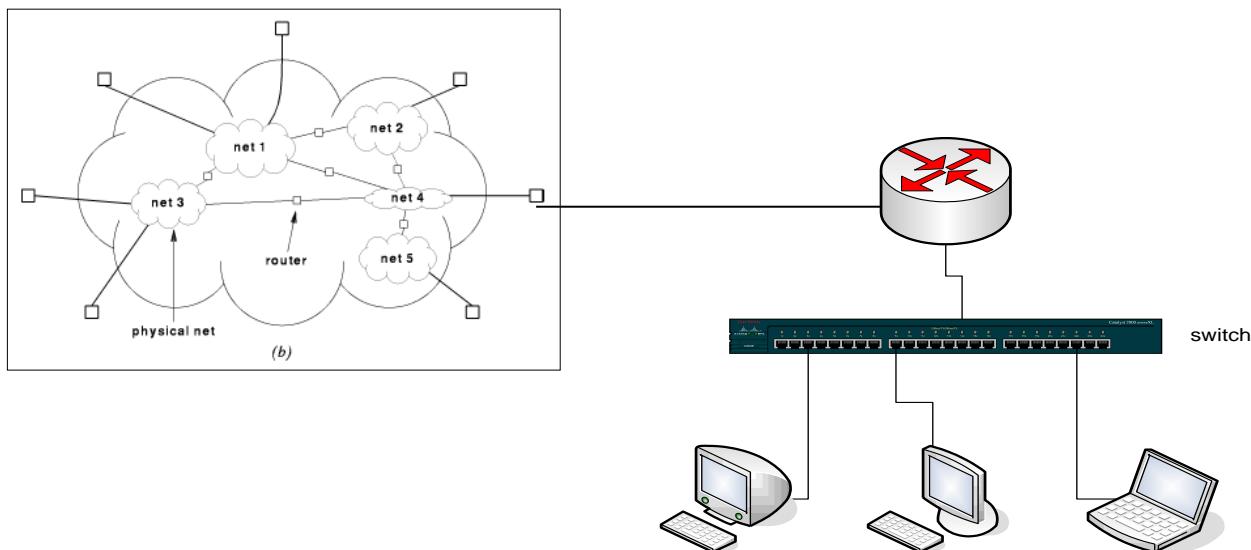
Er zijn 3 types van IPv4 adressen :

- **unicast** (1 host) , de adressen van klasse A, B en C. Slechts 1 bestemming mogelijk per adres.
- **multicast** (klasse D = een gedefinieerde groep gebruikers op het net, zie IGMP).
- **broadcast** : 255.255.255.255<sup>1</sup> of *netid*.255. ... (alle hosts binnennin 1 netwerk)

De overige adressen, = klasse E, (1111...) zijn gereserveerd voor de toekomst en voor testprocedures.

Aangezien elke interface op het internet een **uniek IP adres** moet hebben is er een centrale autoriteit die de **netwerken of- 'netid's** toekent : IANA of Internet Assigned Numbers Authority, <http://www.iana.org/>.<sup>2</sup>

De **hostid's** moeten toegekend worden door de **plaatselijke systeemverantwoordelijke**. Uiteindelijk dient de **plaatselijke organisatie niet wereldwijd gekend** te zijn door de (internet-) routers : boodschappen worden **binnenin 1 netwerk** afgeleverd door de **switches**.



Figuur 275 hosts op een netwerk.

De **hostid's** specificeren een host op een netwerk, of beter nog : een **interface** van een host. Hosts met meerdere interfaces kunnen dus meerdere IP adressen bevatten (bv. wired & wireless) en dus bereikt worden langs verschillende wegen/netwerken.

<sup>1</sup> Een broadcastadres 255.255.255.255 passeert normaal niet door een router, m.a.w. blijft binnennin een netwerk. Anders zou het Internet snel verzadigd geraken.

<sup>2</sup> Dit wordt wel gedelegeerd naar plaatselijke RIR's , Regional Internet registry's, zoals voor europa : Réseaux IP Européens Network Coordination Centre , [www.ripe.net](http://www.ripe.net).

### 3.3.2 OEFENINGEN OP VOLLE KLASSE NETWERKEN (TER INFO!!)

## OEFENING 1:

Definieer **klasse**, **netwerk-id** en **host-id**.

- 054.172.045.032.....
  - 195.026.204.002.....
  - 138.002.004.152.....
  - 190.254.013.241.....

Vroeger werden adressen met hostid =  $\forall 0$  wel eens gebruikt voor broadcasts. Nu praktisch niet meer en bijgevolg worden ze gebruikt om een **netwerk** aan te duiden. Bvb. 172.16.254.003 is een host van netwerk ... 172.16.0.0 (Klasse B). Nochtans is een netwerk als dusdanig niet bereikbaar (er zit geen intelligentie op een kabel), het is enkel een voorstelling.

Routers, die verschillende internetsegmenten verbinden, hebben een ‘relay-functie’ op laag 3. Ze koppelen 2 (of meer) netwerken met elkaar. Ze moeten direct aanspreekbaar zijn voor elke host op de gekoppelde netwerken (om de pakketten met bestemmingen op andere netwerken te ontvangen en verder te sturen, ‘relay’)

→ een router heeft een IP adres per interface (= 1 per gekoppeld netwerk).<sup>1</sup>

<sup>1</sup> Merk op dat ‘multihomed hosts’ ook **meerdere IP adressen** bezitten, 1 per interface.

## OEFENING 2.

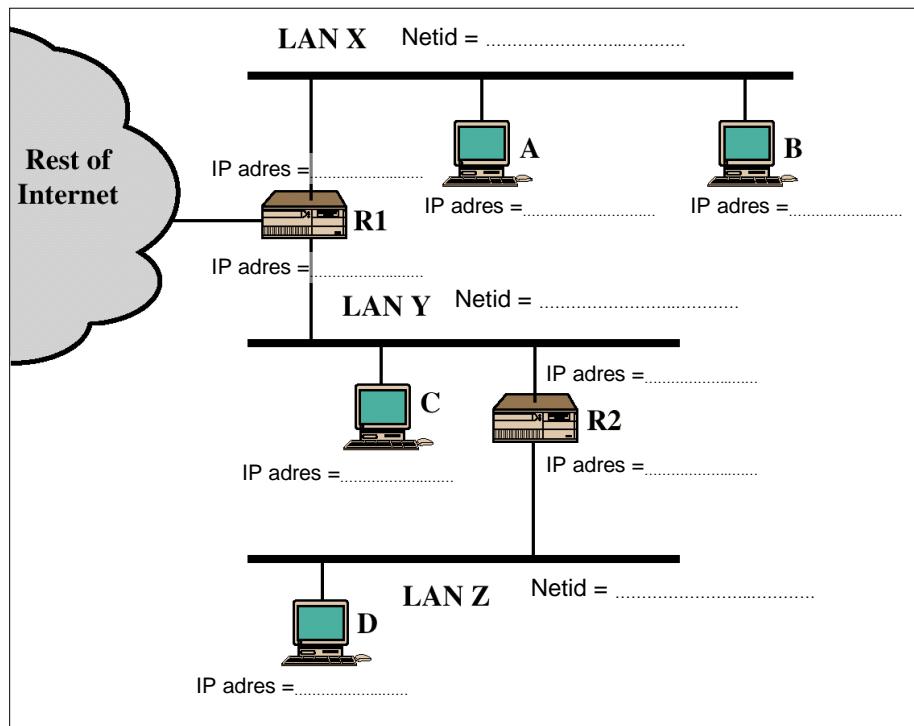
Als systeemverantwoordelijke voor een intranetwerk bestaande uit 3 LAN's moet u IP-adressen aanvragen.

LAN X bevat 320 hosts,

LAN Y : 120 hosts,

LAN Z : 12 hosts.

Geef een mogelijke economische oplossing.



Figuur 276 Toekenning van IP adressen.

Wat wordt de oplossing als Router R2 vervangen wordt door een **switch/bridge**?.....

Wat als R2 → een hub/repeater ?.....

### OEFENING 3. VOLLE KLASSE

Los de figuur op met 'volle klasse' netwerken.

Plaats op de figuur de gepaste IP adressen

Site A : A1 bezit 600 hosts

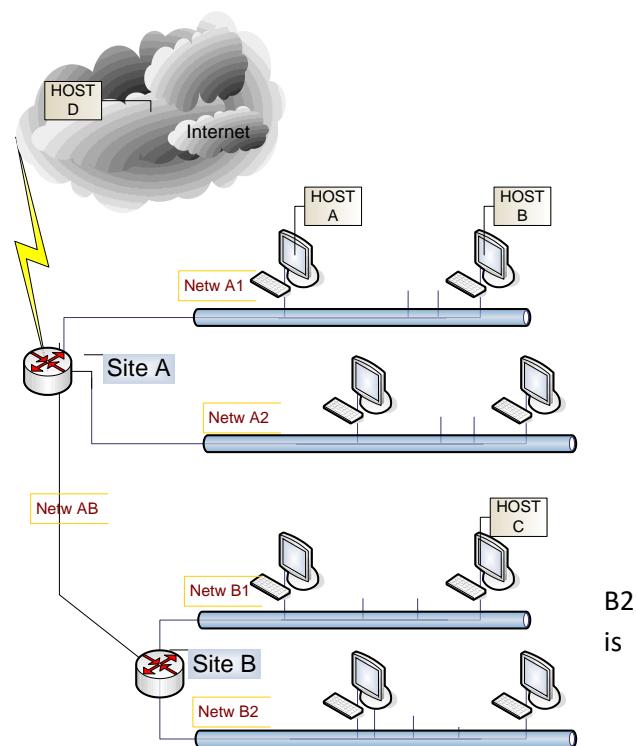
A2 bezit 200 hosts

Site B : B1 bezit 300 hosts

B2 bezit 150 hosts

### OPLOSSING.

Een router scheidt netwerken, dus A1, A2, B1 en zijn allen verschillende netwerken, en bovendien ook de P2P (Point to Point) link een verschillend netwerk → AB.



- A1 : 600 → kl B, bv. 140.121.0.0 (=netwerk aanduiding,  $\geq 128.x, \leq 191.x$ )
  - Router A intf A1 = 140.121.0.1, host A = 140.121.0.2, host B = 140.121.255.254
  - Broadcasten op net A1 doet u door te sturen naar IP = 140.121.255.255
- A2 : 200 → kl C, bv. 200.21.0.0 (=netwerk,  $\geq 192.x.x, \leq 223.x.x$ )
  - Router A intf A2 = 200.21.0.1, hosts = 200.21.0.2, tot = 200.21.0.254
  - Broadcasten op net A2 doet u door te sturen naar IP = 200.21.0.255
- B1 : 300 → kl B, bv. 140.122.0.0 (=ander netwerk,  $\geq 128.x, \leq 191.x$ )
  - Router B intf B1 = 140.122.0.1, host = 140.122.0.2, tot host C = 140.122.255.254
  - Broadcasten op net B1 doet u door te sturen naar IP = 140.122.255.255
- B2 : 150 → kl C, bv. 200.21.1.0 (=netwerk aanduiding,  $\geq 192.x.x, \leq 223.x.x$ )
  - Router B intf B2 = 200.21.1.1, hosts = 200.21.1.2, tot = 200.21.1.254
  - Broadcasten op net B2 doet u door te sturen naar IP = 200.21.1.255
- AB : 2 IP adressen → kl C, bv. 200.21.2.0 (=netwerk aanduiding,  $\geq 192.x.x, \leq 223.x.x$ )
  - Router A intf AB = 200.21.2.1, Router B intf AB = 200.21.2.2,

De enige (controleerbare) interface die nu nog geen IP adres heeft is de link naar het internet van router A. Die moet een IP adres krijgen van het netwerk van de service provider, bv. telenet.

Vraag. Hoe stuurt **host A** een IP pakket naar **host B** ?

- Host A bekijkt het dest. IP adres van B = 140.121.255.254.  
A weet dus dat dit een **kl B** adres is van netwerk **140.121.0.0**.  
A bekijkt zijn **eigen IP adres** en ziet dat dit behoort tot **hetzelfde netwerk 140.121.0.0**.  
A speelt dit door aan **laag 2** om dit af te leveren **BINNENIN** het eigen netwerk.

Vraag. Hoe stuurt **host A** een IP pakket naar **host C** ?

- Host A bekijkt het dest. IP adres van C = 140.122.255.254.  
A weet dus dat dit een **kl B** adres is van netwerk **140.122.0.0**.  
A bekijkt zijn **eigen IP adres** en ziet dat dit **niet** behoort tot **hetzelfde netwerk (140.121.0.0)**.  
A stuurt het dan maar naar **router A** voor verdere routering. Router A heeft een interface in netwerk A1 en is dus bereikbaar voor host A via **laag 2** (afleveren BINNENIN het eigen netwerk).
- Merk wel op dat het **dest. IP adres (L3)** het uiteindelijke bestemmingsadres van **host C** is.
- Router A **zoekt** de weg naar (het netwerk van~) host C (bestemmingsadres) en vindt dat dit te bereiken is **via router B**. Router A stuurt het via netw AB naar router B. (aflevering via **L2**)
- Router B zoekt de weg naar host C en ziet dat een van zijn interfaces ook in dit netwerk **140.122.0.0** ligt. Hij laat het door L2 afleveren via deze interface.

OPM. de manier waarop routers hun routingstabellen opstellen en dus uiteindelijk de ligging van alle netwerken kennen valt buiten het bestek van dit werk.

Vraag. Hoe stuurt **host A** een IP pakket naar **host D** ?

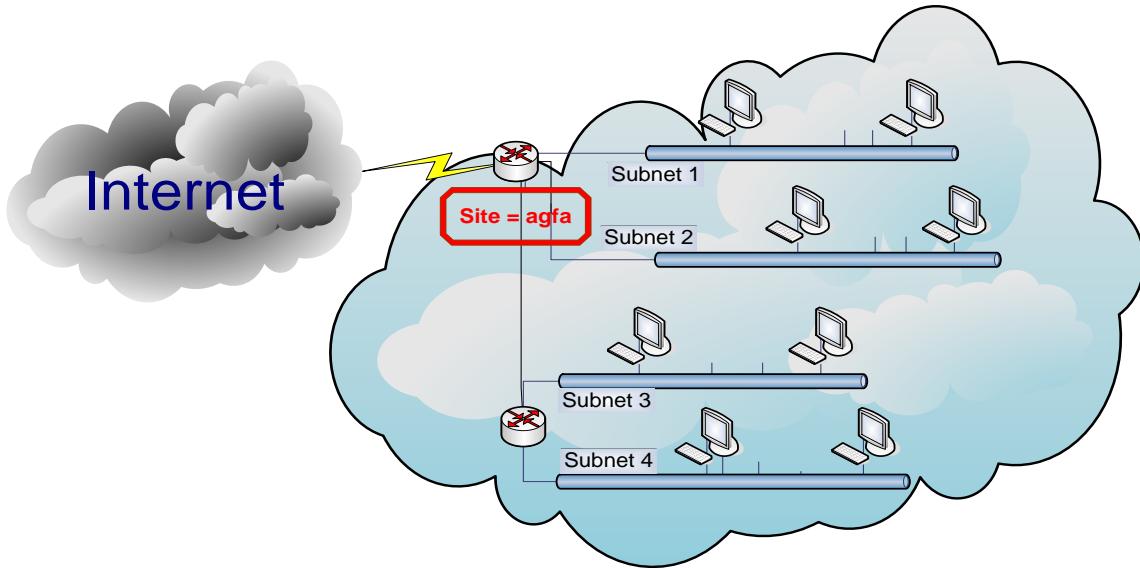
- Host A bekijkt het dest. IP adres van D bv. = 198.12.25. 4.  
A weet dus dat dit een **kl C** adres is van netwerk **198.12.25.0**.  
A bekijkt zijn eigen IP adres en ziet dat dit **niet** behoort tot **hetzelfde netwerk (140.121.0.0)**.  
A stuurt het dan maar naar **router A** voor verdere routering. Id. (Router A heeft een interface in netwerk A1 en is dus bereikbaar voor host A via **laag 2**, afleveren BINNENIN het eigen netwerk).
- **het dest. IP adres (L3)** = het uiteindelijke bestemmingsadres van **host D**.
- Router A **zoekt** de weg naar (het netwerk van~) host D (=bestemmingsadres) en vindt dat dit te bereiken is **via zijn ISP** verbinding, waar hij een adres van een **router X** heeft.
- Router X zoekt de weg naar host D en levert het af aan de router die volgens hem op de kortste weg ligt naar host D. Dit proces herhaalt zich verschillende keren op de tussenliggende routers. Uiteindelijk ziet de router die voor het netwerk van host D staat dat een van zijn interfaces ook in dit netwerk 198.12.25.0 ligt.
- Hij laat het door L2 afleveren via deze interface.

### 3.3.3 IP SUBNET – ADRESSEERING

Na een tijdje bleek de indeling in 3 klassen veel te ruw te zijn : geen mens die 65000 computers in 1 netwerk beheert (klasse B), laat staan de 16M van klasse A. In vele gevallen is een klasse C dan weer te klein met 254 hosts.

Er is sinds **RFC 950, 1985** een aanpassing van het IP-adresseringssysteem d.m.v. ‘subnetten’: alle hosts (= IP stacks) moeten **subnetting** ondersteunen. De toepassing hiervan is een ‘**lokale** verantwoordelijkheid’.

Na het verkrijgen van een netwerk-ID van **Ripe**, bepaalt de **systeemverantwoordelijke** zelf welk subnet hij aanbrengt.

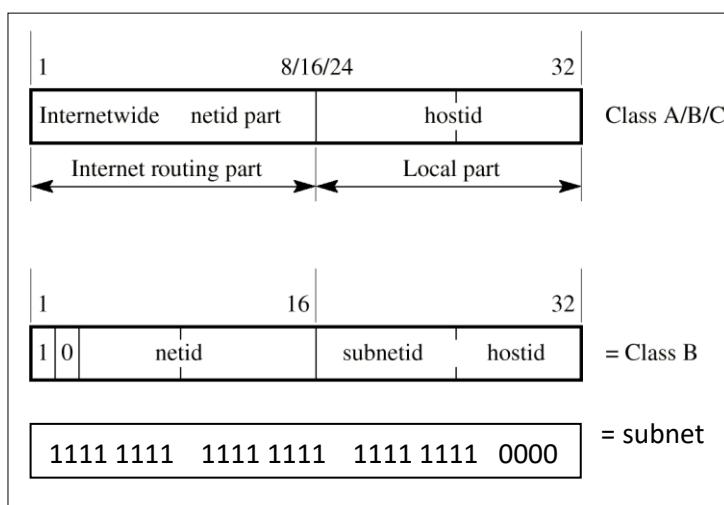


Figuur 277 Principe van subnetting .

Voor het internet zijn alle IP adressen van bv. de site Agfa te vinden achter 1 specifieke router, m.a.w. alle hosts van agfa hebben een adres uit 1 ‘volle’ klassebereik, bv. een kl B.

De **interne** organisatie binnenin agfa was vroeger ook al de verantwoordelijkheid van de netwerk-ing. Deze kan nu echter die grote massa computers (bv. kl B → 65000) verder opsplitsen in **subnets** om het geheel beheersbaar te houden. Elk subnet wordt verbonden via routers. M.a.w. een router scheidt nog steeds netwerken maar nu ook **subnetwerken**.

De volledige ‘site’ krijgt (nog steeds) een eigen **netid**. Bovenaan de figuur is de ‘full class’ indeling te zien in



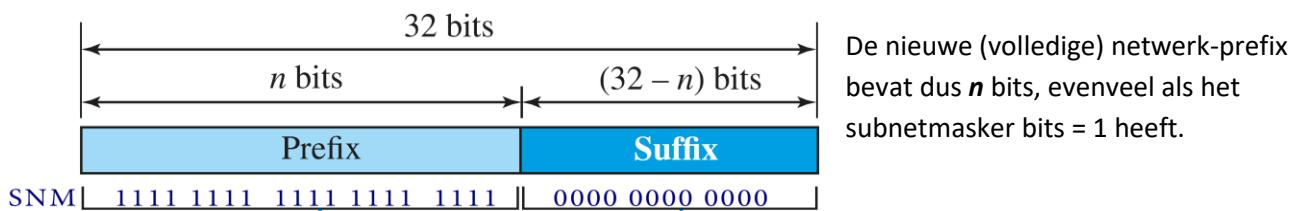
klasse A, B of C, op de **byte-grens** 8/16/24.

De subnets worden verder (lokaal) gespecificeerd door de **eerste bits** van de vroegere (‘full class’) **hostid**.

Onderaan is als vb. een klasse B adres te zien (netid tot bit 16) waarvan het **host-gedeelte** onderverdeeld is in een **subnetid** en **hostid**.

Figuur 278 Subnetting van een Klasse B adres

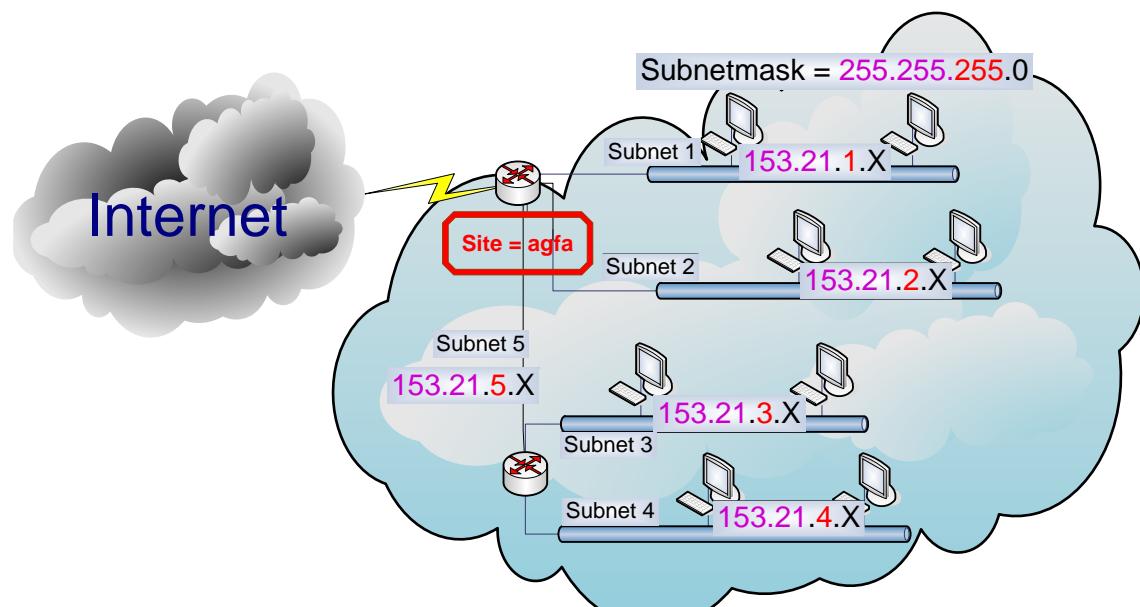
De subnetgrens is in te stellen **per bit** en gebeurt door een **subnet-masker (SNM)**. Het **subnet-masker** bevat  $\forall 1$  voor het **netid**- en **subnetid** gedeelte,  $\forall 0$  voor de hostid. (contiguous 1's  $\rightarrow$  contiguous 0's )



Figuur 279. Subnetmasker lengte

In onderstaand vb. ligt het subnetid op de bytegrens, dat is gemakkelijk om in de IP adressen direct het subnet te herkennen, nl het 3<sup>e</sup> getal. Zie ook onderstaande figuur, maar dit is niet verplicht.

Hier bv. = 1111 1111. 1111 1111. | 1111 1111. | 0000 0000 of decimal dotted : 255.255.255.0.<sup>1</sup>



Figuur 280 Subnetting van een Klasse B site op de bytegrens

Dit masker is een '**lokale aangelegenheid**' en moet gekend zijn voor **alle routers op de site** én voor de **lokale 'internet gateway'**. Het voordeel aan dit systeem is dat **wereldwijd** de internet gateways enkel het klasse B netid hoeven op te slaan in hun tabellen. Alle datagrammen naar deze klasse B 'site' worden gestuurd naar de 'internet gateway' die ze op zijn beurt moet verdelen over de lokale netwerken op basis van het subnet.

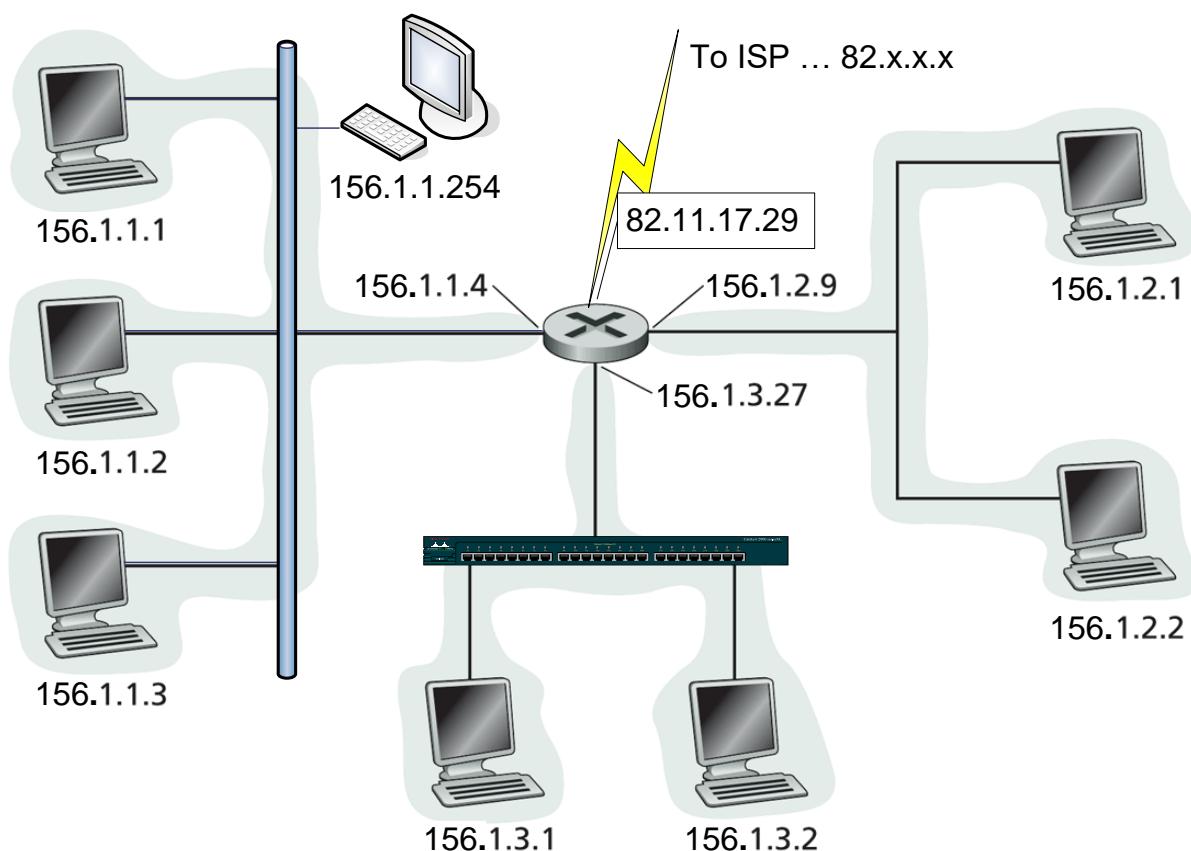
De **subnetindeling** van de agfa site is dus **niet gekend buiten** de grenzen van deze site. Zo zal je dus ook van een bestemming die je contacteert op het internet het subnetmasker niet kennen, enkel het IP adres.

<sup>1</sup> Een andere voorstelling is een / met het aantal bits netid+subnetid, hier /24. Op die manier krijgt met een sneller beeld van de grootte van het netwerk.

In volgend voorbeeld is een deel van een klasse B netwerk (156.1.0.0) besproken. Alle IP adressen beginnen dus met **156.1**, het **netid** van de volledige site. Het subnetmasker ligt op de byte grens zoals hierboven.

De interne router koppelt de verschillende subnets met elkaar, alsook voorziet hij toegang tot het internet via de ISP. Aan deze lijn krijgt hij een IP adres van de ISP.

De routers op het internet weten dat (alle hosts op) het netwerk 156.1.x.x te bereiken is via 82.11.17.29.. (I.p.v. bv. 30 entry's voor 30 subnetten → internet routing tabellen verkleinen.) Ze hoeven ook het plaatselijk subnetmasker **niet** te kennen aangezien dit een **plaatselijke** aangelegenheid is. De **lokale** routers en hosts moeten dit subnetmasker dus **wél** kennen!



klasse B : 156.1.0.0 netwerk met subnetmask 255.255.**255.0**

Figuur 281 Een klasse B netwerk met subnets op de byte-grens

Merk op dat :

- Alle interfaces van de hosts een IP adres bevatten (als ze bereikbaar willen zijn)
- Ook alle interfaces van de router een IP adres bezitten
- **All** deze interfaces IP-adressen bevatten **IN** het **subnet** waar ze zijn aangesloten.
- Deze IP adressen **UNIEK** zijn, zelfs wereldwijd ! (aangezien ook het netid slechts 1 keer is toegekend)
  - subnetid en hostid mag je lokaal ook maar 1 keer (= uniek) toekennen.

- Een subnet bestaat eigenlijk uit 1 switch (of hub) die op L2 (resp L1) de apparaten verbindt, hier ook symbolisch voorgesteld door een ‘ethernetsegment’<sup>1</sup>, of gewoon een lijn.
  - Computers op 1 subnet met elkaar kunnen communiceren ZONDER de router.
  - Om van 1 subnet naar een ander te gaan moet men via de router passeren → ‘next hop’
  - Een host die wil communiceren heeft minimaal 2, normaal 3 parameters:
    - Een IP adres (**IP**)
    - Een subnetmasker (**SNM**)
    - *Een ‘default gateway’ (**DGW**).*

(de 3<sup>e</sup> parameter, DGW, heeft hij enkel nodig als hij ook buiten zijn eigen subnet wil communiceren.)

## IP – HOSTWERKING BIJ SUBNETTING

Indien een host een datagram moet versturen moet hij eerst bepalen of de bestemming **op zijn eigen netwerk** te vinden is, dan wel op een ander. Afh. van de uitkomst hiervan is de behandeling namelijk verschillend:

1. In het eerste geval moet hij het versturen 'rechtstreeks', = via L2 naar zijn buur. (de 'next hop' = buur).
  2. Indien de bestemming **niet** op zijn **eigen netwerk** is gelegen, zal hij een router moeten passeren om naar dit andere netwerk te gerakten. Meestal<sup>2</sup> is een host niet erg intelligent en heeft hij maar 1 uitweg voor bestemmingen op een ander netwerk en dit noemt men de '**default gateway**'<sup>3</sup> (in dit geval DGW = 'next hop').

Merk op dat de router in vorige Figuur 281: **4 interfaces** heeft, en dus ook **4 IP adressen**, 1 per interface. Je stelt als DGW een adres in van de router dat voor de hosts rechtstreeks bereikbaar is: bv. voor alle hosts in subnet 1 (=156.1.1.x) is de DGW = 156.1.1.4! → het IP adres IN het subnet van de host. Dit is dus bereikbaar via L2.

Terug naar de vraag: hoe weet een host of de bestemming op zijn eigen netwerk ligt of niet ? (Of m.a.w. ... Hoe kiest een host de 'next hop' voor een bestemming? ).

Een host heeft sinds RFC 950 (=subnetting) minstens 2 IP-parameters :



Hij kan hiermee zijn (eigen) **netwerk-id** berekenen door een **AND functie** uit te voeren op deze 2 parameters.

<sup>1</sup> Meestal wordt dit segment niet meer getekend : men neemt op layer 3 abstractie van de L2 en L1 aansluiting.

<sup>2</sup> Dit is niet verplicht, er kunnen meerdere routers aan een netwerk aangesloten zijn, en de hosts kunnen de kortste paden bijhouden naar al hun bestemmingen, maar doen dit meestal niet. Dit betekent een beetje meer werk voor de routers.

<sup>3</sup> Gateway is een ander woord voor routers. Wordt tegenwoordig als synoniem voor elkaar gebruikt.

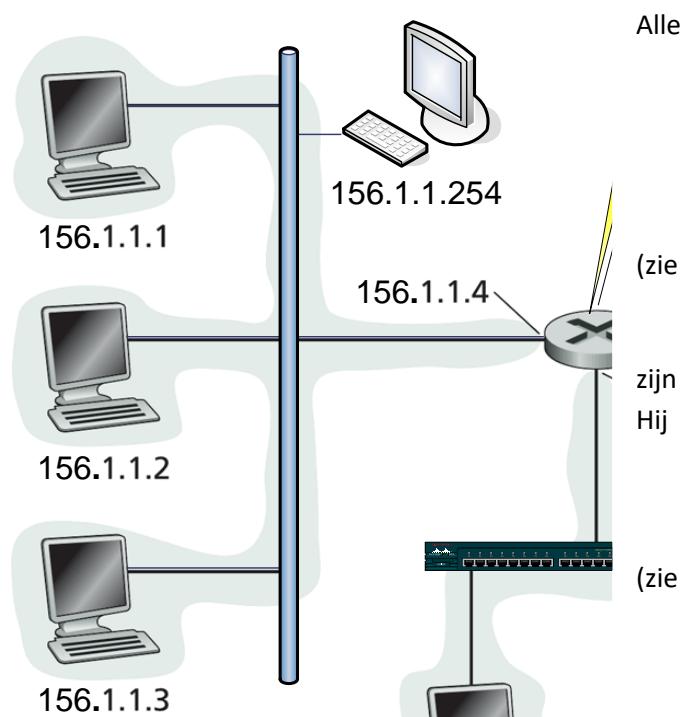
Vb. het linkse subnet van vorige figuur.  
hosts (van subnet1) hadden een subnetmasker (SNM) = 255.255.255.0. Ze verkrijgen als uitkomst van de vorige AND-functie hun **netid** (&subnetid) :

- **IPs & SNM = 156.1.1.0.**  
ook onder, Binaire voorstelling.)

Hoe weet een host dan of de **bestemming** op eigen netwerk ligt of niet ?  
doet **opnieuw** een AND functie , nu met het **bestemmingsadres: IPd**.

- **IPd & SNM = ??? → = ? 156.1.1.0. ?**  
ook onder, Binaire voorstelling.)

Is deze uitkomst identiek dan ligt de bestemming op zijn eigen netwerk!.



Figuur 282 subnet 1 van klasse B netwerk met SNM = 255.255.255.0

→ Het datagram is bestemd voor :

1. Een host op zijn **eigen subnet** → inpakken en via L2 **direct** versturen
2. Een host op een **ander** (sub)netwerk → via L2 sturen naar router, meestal **DGW**

#### PRAKTISCH VOORBEELD.

Vb. de host met **IPs = 156.1.1.2** wenst een aantal datagrammen te versturen.

#### 1. Berekening van zijn **eigen subnet**. **IPs & SNM**

Eigen subnet ?	Dotted decimal	Binaire voorstelling
IPs adres	156.1.1.2	<u>1001 1100</u> <u>0000 0001</u> <u>0000 0001</u> <u>0000 0010</u>
Subnet mask	255.255.255.0	<u>1111 1111</u> <u>1111 1111</u> <u>1111 1111</u> <u>0000 0000</u>
AND → subnet ?	<u>156.1.1.0</u>	<u>1001 1100</u> <u>0000 0001</u> <u>0000 0001</u> <u>0000 0000</u>

2. Zendt een datagram naar host IPd = **156.1.3.2**. Is deze bestemming op mijn eigen subnet? **IPd & SNM**

Eigen subnet ?	Dotted decimal	Binaire voorstelling
IPd adres	156.1.3.2	<u>1001 1100</u> <u>0000 0001</u> <u>0000 0011</u> <u>0000 0010</u>
Subnet mask	255.255.255.0	<u>1111 1111</u> <u>1111 1111</u> <u>1111 1111</u> <u>0000 0000</u>
AND → subnet ?	<u>156.1.3.0</u>	<u>1001 1100</u> <u>0000 0001</u> <u>0000 0011</u> <u>0000 0000</u>

NEEN !

Oplossing : versturen via L2 naar de **DGW** = router int. **156.1.1.4** .

Let op, het ‘destination IP address’ blijft **156.1.3.2**, (=stempel op L3, IP pakket, wereldwijde aflevering). Maar op **L2**, binnen het eigen subnetwerk, wordt het verstuurd naar het L2 adres van de DGW. Er zal hiervoor nog een adres-mapping moeten plaatsvinden tussen L3 en L2 adressen : ARP.

3. IPs zendt een ander datagram naar host **156.1.1.254**. Is deze bestemming op mijn eigen subnet? .

Eigen subnet ?	Dotted decimal	Binaire voorstelling
IPd adres	156.1.1.254	<u>1001 1100</u> <u>0000 0001</u> <u>0000 0001</u> <u>1111 1110</u>
Subnet mask	255.255.255.0	<u>1111 1111</u> <u>1111 1111</u> <u>1111 1111</u> <u>0000 0000</u>
AND → subnet ?	<b>156.1.1.0</b>	<u>1001 1100</u> <u>0000 0001</u> <u>0000 0001</u> <u>0000 0000</u>

JA !

Gevolg : direct versturen naar de bestemming met op L3 het IP adres 156.1.1.254 **én** op **L2** het L2 adres van dezezelfde interface. ( ook te bekomen via ARP)

4. IPs zendt een datagram naar een internet host **193.130.240.5**. Bestemming op mijn eigen subnet? .

Eigen subnet ?	Dotted decimal	Binaire voorstelling
IP adres	193.130.240.5	<u>1100 0001</u> <u>1000 0010</u> <u>1111 0000</u> <u>0000 0101</u>
Subnet mask	255.255.255.0	<u>1111 1111</u> <u>1111 1111</u> <u>1111 1111</u> <u>0000 0000</u>
AND → subnet ?	193.130.240.0	<u>1100 0001</u> <u>1000 0010</u> <u>1111 0000</u> <u>0000 0000</u>

NEEN ! Net zoals pt. 2, versturen (op L2) naar de router. Deze zoekt de bestemming op in zijn routing tabellen en stuurt het verder naar de ISP. Daarna zal het op dezelfde manier een reeks routers passeren die het zullen opzoeken en uiteindelijk afleveren bij de bestemming.

### GEREERVEerde IP ADRESSEN BIJ SUBNETTING

Merk op dat, net zoals bij de volle klasse, sommige IP adressen een **speciale functie** hebben. Nu ook per subnet! ... en sommige van deze definities nu gewijzigd zijn .

Zijn niet te gebruiken als unicast :

- **Broadcastadressen** : (enkel in het destination veld, staan uiteraard nooit in het source veld)

- **Netid & hostid =  $\forall 1$** . ➔ een beperkte broadcast : hij wordt niet ge'forward' door routers ➔ = **locale broadcast**. ➔ op het subnet waar ze vertrekken.

NETid	Hostid
1111 ...	1111 1111 ... 1111

- **subnet & hostid =  $\forall 1$** . ➔ 'gerichte' broadcast naar alle subnetten van een volle klas netwerk, bvb. de volledige site van een klasse B: 140.252.**255.255**. of bvb. een volledige klasse C : 192.150.26.**255**.

NETid	subnetid	Hostid
...	1111 ... 1111	1111 ... 1111

- **hostid =  $\forall 1$** . ➔ 'gerichte' broadcast naar 1 **subnet** van een netwerk, bvb. op subnet **3** : de hostid =  $\forall 1$  ➔ 140.252.**3.255**

NETid	subnetid	Hostid
...	0000 ... 0011	1111 ... 1111

- **netwerkaanduidingen** : Adressen met **hostid =  $\forall 0$**  (= aanduiding van de netwerken, nu ook van de subnetwerken) .

Bvb.

- de site van het **volledige** klasse B netwerk : **subnet & hostid =  $\forall 0$**  ➔ 142.25.**0.0** of van een klasse C : 192.150.26.**0**.

NETid	subnetid	Hostid
...	0000 ... 0000	0000 ... 0000

- subnet **3** van klasse B : de hostid =  $\forall 0$  ➔ 142.25.**3.0**

NETid	subnetid	Hostid
...	0000 ... 0011	0000 ... 0000

**Resumé** : In de oorspronkelijke richtlijnen is opgenomen dat volgende adressen **niet als unicast** toegekend worden :

1. **hostid =  $\forall 1$**  : uiteraard, het is het **broadcastadres** (per subnet).
2. **hostid =  $\forall 0$**  : het is de (sub-) **netwerkaanduiding** (was broadcast bij BSD 4.0 unix zie oef1).
3. **subnetid =  $\forall 1$  en subnetid =  $\forall 0$**  : ... mogelijk verwarring broadcasts (zie oef1)

Praktisch leidt dit laatste punt tot behoorlijk veel verlies aan bruikbare adressen en wordt ze 'bijgestuurd' in de meeste routers door "allow all 0's and all 1's as **subnetaddress**".

Op die manier worden deze adressen toch toegekend als geldige adressen.

**GELDIGE SUBNETMASKERS.**

Subnetmaskers zijn per definitie instelbaar tot op de bitgrens, én dotted decimal genoteerd. Aangezien ze steeds “*contiguous 1's and 0's*” bevatten (van links naar rechts) zijn er **per byte** slechts 8 mogelijkheden

Bovendien beginnen ze steeds met 255. ... omdat de ‘smalste’ netid, die van een klasse A, de volledige 1<sup>e</sup> byte netid bevat, dus  $\forall 1$ . De volgende bytes kunnen enkel de waarden aannemen uit deze tabel.

Subnetmasker waarden per byte, dotted decimal.

<b>binair</b>	<b>decimaal</b>	<b>berekening</b>
... .0000 0000. ...	... .0. ...	
... .1000 0000. ...	... .128. ...	= 1 * 2 <sup>7</sup>
... .1100 0000. ...	... .192. ...	= 1 * 2 <sup>7</sup> + 1 * 2 <sup>6</sup> = 128 + 64
... .1110 0000. ...	... .224. ...	= 2 <sup>7</sup> + 2 <sup>6</sup> + 2 <sup>5</sup> = 128 + 64 + 32
... .1111 0000. ...	... .240. ...	= 2 <sup>7</sup> + 2 <sup>6</sup> + 2 <sup>5</sup> + 2 <sup>4</sup> = 128 + 64 + 32 + 16
... .1111 1000. ...	... .248. ...	= 2 <sup>7</sup> + 2 <sup>6</sup> + 2 <sup>5</sup> + 2 <sup>4</sup> + 2 <sup>3</sup> = 128+64+32+16+8
... .1111 1100. ...	... .252. ...	= 2 <sup>7</sup> + 2 <sup>6</sup> + 2 <sup>5</sup> + 2 <sup>4</sup> + 2 <sup>3</sup> + 2 <sup>2</sup> = 128+64+32+16+8+4
... .1111 1110. ...	... .254. ...	=2 <sup>7</sup> +2 <sup>6</sup> +2 <sup>5</sup> +2 <sup>4</sup> +2 <sup>3</sup> +2 <sup>2</sup> +2 <sup>1</sup> =128+64+32+16+8+4+2
... .1111 1111. ...	... .255. ...	=2 <sup>7</sup> +2 <sup>6</sup> +2 <sup>5</sup> +2 <sup>4</sup> +2 <sup>3</sup> +2 <sup>2</sup> +2 <sup>1</sup> +2 <sup>0</sup> =128+64+32+16+8+4+2+1

### 3.3.4 OEFENINGEN OP SUBNETTING VAN NETWERKEN

#### OEFENING 1.

Een bestaand netwerk, 194.17.150.0, heeft een subnetmask = 255.255.255.224. (= ... **1110 0000**)

Duidt alle subnets / hosts aan en de resp. broadcastadressen.

subnet	Hosts :begin		Hosts : Einde		broadcastadres	
0	... <b>0000 0001</b>	194.17.150.1	... <b>0001 1110</b>	194.17.150.30	... <b>0001 1111</b>	194.17.150.31
1	... <b>0010 0001</b>	194.17.150.33	... <b>0011 1110</b>	194.17.150.62	... <b>0011 1111</b>	194.17.150.63
2	... <b>0100 0001</b>	194.17.150.65	... <b>0101 1110</b>	194.17.150.94	... <b>0101 1111</b>	194.17.150.95
3	... <b>0110 0001</b>	194.17.150.97	... <b>0111 1110</b>	... .126	... <b>0111 1111</b>	... .127
4	... <b>1000 0001</b>	... .129	... <b>1001 1110</b>	... .158	... <b>1001 1111</b>	... .159
5	... <b>1010 0001</b>	... .161	... <b>1011 1110</b>	... .190	... <b>1011 1111</b>	... .191
6	... <b>1100 0001</b>	... .193	... <b>1101 1110</b>	... .222	... <b>1101 1111</b>	... .223
7	... <b>1110 0001</b>	... .225	... <b>1111 1110</b>	... .254	... <b>1111 1111</b>	... .255

Houdt u rekening met **opm. 3** dan moet u subnet 0 en subnet 7 schrappen.

Doet u dit niet dan hebt u problemen met de laatste broadcast : 194.17.150.255.

Is dit een broadcast op **subnet 7**, of een broadcast voor **alle** subnets?.

Dit is de reden waarom het ganse subnet in de definitie werd geschrapt. In de praktijk zullen routers geen broadcast naar alle subnetten toelaten en bijgevolg wordt dit een broadcast op **subnet 7**.

We zullen dit verder dus steeds 'omzeilen' en toch toelaten door "*allow all 0's and all 1's as subnetaddress*".

(Later, bij VLSM, wordt het effect geminimaliseerd, zie 2BA3).

## OEFENING 2.

Gegeven is van Ripe een klasse C adres 195.193.5.0.

Er moeten in totaal 28 subnetwerken aangesloten worden die elk maximum 5 hosts ondersteunen.

Wat moet als subnetmasker worden ingesteld? <sup>1</sup>

## OEFENING 3.

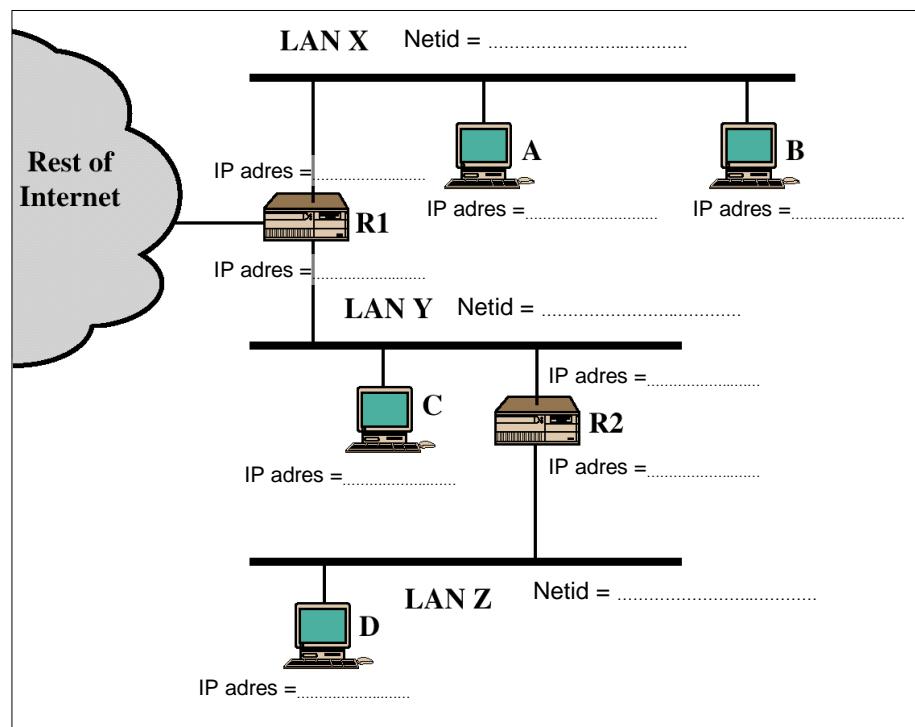
Als systeemverantwoordelijke voor een intranetwerk bestaande uit 3 LAN's moet u IP-adressen aanvragen.

LAN X bevat 60 hosts,

LAN Y : 35 hosts,

LAN Z : 12 hosts.

Geef een mogelijke economische oplossing.



Welke klasse vraagt u aan? ...<sup>2</sup>.

Figuur 283 Toekenning van IP adressen.

Welk subnetmasker gebruikt u ? <sup>3</sup>

Vul een reeks mogelijke adressen in. (zie ppt slides)

<sup>1</sup> 255.255.255.248 = **1111 1111. 1111 1111. 1111 1111. 1111 1000** zijnde 5 bits voor max. 32 subnets en 3 bits voor max. 8 – 2 = 6 hosts, router inbegrepen.

<sup>2</sup> Klasse C

<sup>3</sup> 255.255.255.192 : Er zijn max. 60 hosts op 1 subnet → 6 bits hostid → 2 bits subnetid → 1...1 1100 0000 .

♦ Oefening 2.

Gegeven het IPadres en subnetmasker. Wat is het netwerk-, subnet-, hostnummer en broadcastadres?<sup>1</sup>

IP – adres	Subnetmasker	Netwerknr	Subnetnr	Hostnr.	Local broadcast
194.7.50.200	255.255.255.224				
172.63.193.28	255.255.248.0				
195.6.45.156	255.255.255.192				
139.56.50.227	255.255.240.0				

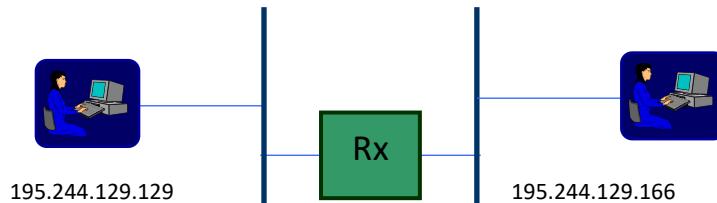
<sup>1</sup> IP = 194.7.50.200 = ... .200 = ... **1100** 1000

en subnetmasker = 255.255.255.224 = ... .224 = ... 1110 0000 → subnet = 6, host = 8

→ **locaal broadcast adres** = 194.7.50.??? = ... ??? = ... **1101 1111** → = 194.7.50.223 !

IP – adres	Subnetmasker	Netwerknr	Subnetnr	Hostnr.	broadcast
194.7.50.200	255.255.255.224	194.7.50	6	8	194.7.50.223
172.63.193.28	255.255.248.0	172.63	24	284	172.63.199.255
195.6.45.156	255.255.255.192	195.6.45	2	28	195.6.45.191
139.56.50.227	255.255.240.0	139.56	3	739	139.56.63.255

- ♦ Oefening 3. Een bestaand netwerk heeft hosts met onderstaande IP adressen.



- Het netwerk heeft als natuurlijke klasse ...<sup>1</sup>
- Moet deze router een subnetmasker toepassen ?<sup>2</sup>
- Zo ja, wat zijn de grenzen van het subnetmasker dat hij moet toepassen ?<sup>3</sup>

IP adres 1	195.244.129.129	<b>1100 0011. 1111 0100. 1000 0001. 1000 0001</b>					
IP adres 2	195.244.129.166	<b>1100 0011. 1111 0100. 1000 0001. 1010 0110</b>					
Min subnet							
Max subnet							

Als u het minimum subnet toepast,

- Wat zijn dan de subnetwerken langs beide kanten ?<sup>4</sup>
- Hoeveel subnetten kan hij installeren met dit masker ?<sup>5</sup>
- Hoeveel hosts kan hij installeren ?<sup>6</sup>
- Hoeveel PC's kan hij op elk subnet plaatsen?<sup>7</sup>

- ♦ Oefening 4.

<sup>1</sup> Klasse C

<sup>2</sup> Ja : hij scheidt tot laag 3 ➔ 2 netwerken

<sup>3</sup>

Min subnet	255.255.255.224	<b>1111 1111. 1111 1111. 1111 1111. 1110 0000</b>
Max subnet	255.255.255.252	<b>1111 1111. 1111 1111. 1111 1111. 1111 1100</b>

MIN : hij moet een onderscheid kunnen maken tussen beide zijden / subnetid's ➔ 4 en 5.

MAX : dit is sowieso het max. subnetadres met slechts 2 hosts per subnet. Nog verder gaan zou betekenen dat alleen maar broadcasts kunnen gegeven worden.

<sup>4</sup> 195.244.129.128 en 195.244.129.160

<sup>5</sup> 3 bits ➔ 8 [-2, evt. niet subnet 111 en 000 ➔ 6 : "allow all 0's and all 1's as subnetaddress" ]

<sup>6</sup> 5 bits ➔ 32 - 2 (niet host 11111 noch host 00000) ➔ 30

<sup>7</sup> 29 : de router heeft ook een interface nodig! ➔ duidt een mogelijke adres aan op de figuur

Gegeven is van interNIC een klasseC adres 195.193.5.0. Er moeten in totaal 28 subnetwerken aangesloten worden die elk maximum 6 hosts ondersteunen.

Wat moet als subnetmasker worden ingesteld? <sup>1</sup>

♦ Oefening 5.

Als systeemverantwoordelijke voor een intranetwerk bestaande uit 3 LAN's moet u IP-adressen aanvragen.

LAN X bevat 60 hosts,

LAN Y : 35 hosts,

LAN Z : 12 hosts.

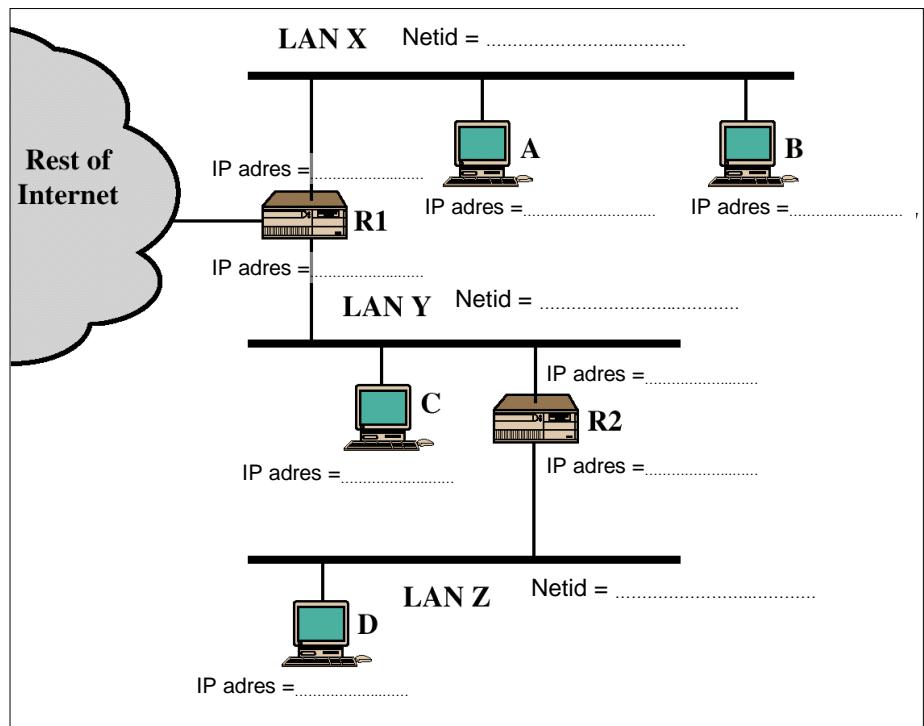
Geef een mogelijke economische oplossing.

Welke klasse vraagt u aan? ...<sup>2</sup>.

Figuur 284 Toekenning van IP adressen.

Welk subnetmasker gebruikt u? <sup>3</sup>

Vul een reeks mogelijke adressen in. (zie ppt)



Dit lokaal netwerk gedraagt zich voor de rest van het internet als een standaard klasse C netwerk met 3 bytes netid en 1 byte hostid. Alle IP pakketten die bestemd zijn voor dit netwerk worden naar R1 gerouteerd. Deze bezit WEL het subnetmasker. Bij aankomst van een pakket zal hij het destination address ANDen met het subnetmasker. Als het bestemd voor...

- subnet X dan levert hij het af (via L2) langs zijn **bovenste** interface aan de bestemming.
- subnet Y → idem langs zijn **onderste** interface.
- subnet Z → aflevering aan 'next hop' R2, die op zijn beurt terug het subnetmask toepast, en zal besluiten om het af te leveren aan de bestemming via zijn onderste interface.

<sup>1</sup> → 255.255.255.248 = **1111 1111. 1111 1111. 1111 1111. 1111 1000** zijnde 5 bits voor max. 32 /(-2→30) subnets en 3 bits voor max. 8 –2 = 6 hosts, router inbegrepen.

<sup>2</sup> Klasse C

<sup>3</sup> 255.255.255.192 : Er zijn max. 60 hosts op 1 subnet → 6 bits hostid → 2 bits subnetid → 1...1 1100 0000

♦ Oefening 6.

De figuur hiernaast geeft een beeld van een bedrijfsnetwerk bestaande uit 3 geografisch gescheiden subnetten.

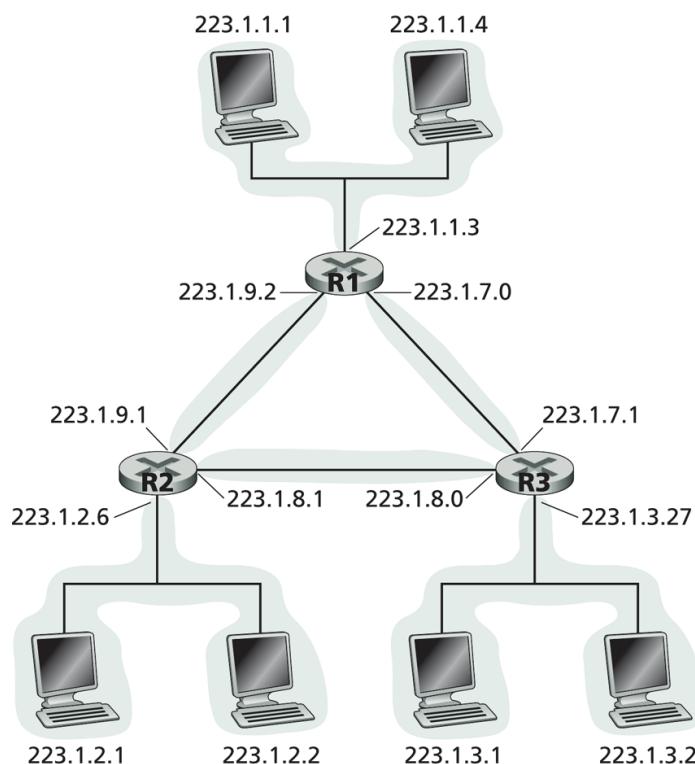
De routers zijn verbonden met elkaar via WAN links. Dit zijn normaal P2P (point to point) verbindingen van router naar router waar geen verdere hosts zich op bevinden.

Bovendien maakt men hiermee een 'redundante' driehoek om te vermijden dat er geen verbinding meer zou zijn als 1 van de WAN links het begeeft.

De auteur heeft echter nogal kwistig omgesprongen met het aanvragen van klasse C netwerken.

Hoeveel stuks zijn dat er? <sup>1</sup>

Dat moest ook wel : hij hield zich aan de regels van de 'volle klas' netwerken.



Figuur 285. Een bedrijfsnetwerk, klasse C.

Als je dit wil verbeteren met **1 klasse C** dat wordt **ge'subnet**'t, hoeveel hosts kan je dan maximaal op 1 subnet plaatsen?<sup>2</sup>

<sup>1</sup> 6 klasse C netwerken.

<sup>2</sup> Merk op dat routers netwerken verbinden. M.a.w. , net zoals het origineel, zijn de 3 P2P links andere netwerken dan de 3 hostnetwerken. Nu worden dit andere subnets. Er moeten dus ook **6 subnets** zijn.

Bijgevolg heb je **3 bits** nodig om de **subnets** aan te duiden. → subnetmask = 255.255.255.224 (...1 **1110 0000**)

Je hebt dan nog, in een klasse C, 5 bits over voor de hostid's →  $2^5 = 32 - 2$  ( $\forall 1^L$  en  $\forall 0^L$ ) → 30 hosts. We gebruiken nu bv. niet subnet **000** en subnet **111**.

Bv. netwerk = 223.1.1.0 met subnetmask 255.255.255.224

- Subnet 1 = 223.1.1.32, (...1 . **0010 0000**) met host 1 = 223.1.1.33 (...1 . **0010 0001**), laatste host = 223.1.1.62 (...1 . **0011 1110**)
- Subnet 2 = 223.1.1.64, (...1 . **0100 0000**) met host 1 = 223.1.1.65, (...1 . **0100 0001**) laatste host = 223.1.1.94, (...1 . **0101 1110**)
- rest net zoals oef 1...merk op dat de P2P links slechts 2 adressen gebruiken van de 30 mogelijke ...
- Subnet 6 = 223.1.1.192, (...1 . **1100 0000**) met host 1 = 223.1.1.193, (...1 . **1100 0001**) laatste host = 223.1.1.223, (...1 . **1101 1110**)

### OEFENING 5.

Los de figuur op, nu met **sub-netwerken**.

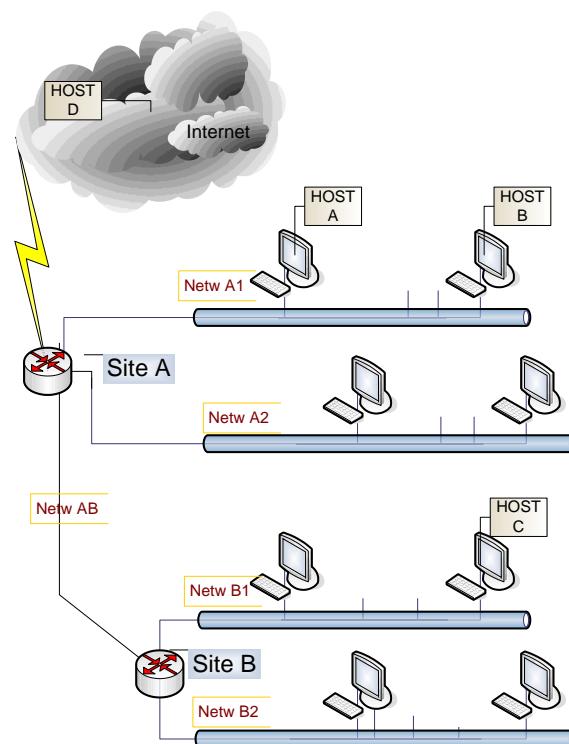
Plaats op de figuur de gepaste IP adressen

Site A : A1 bezit 50 hosts

A2 bezit 20 hosts

Site B : B1 bezit 30 hosts

B2 bezit 15 hosts



Figuur 286. Een bedrijfsnetwerk, te subnetten.

### OPLOSSING.

Een router scheidt (sub-)netwerken, dus A1, A2, B1 en B2 zijn allen verschillende netwerken, en bovendien is ook de P2P (Point to Point) link een verschillend subnetwerk → netw AB.

We hebben in totaal dus **5 subnetten**, met daarin max. **50 hosts**.

- Voor de 5 subnetten hebben we min. 3 bits nodig,
- voor de 50 hosts min 6 bits → 1 klasse C is onmogelijk. (met vast SNM).

→ 1 kl B: 140.121.0.0 .

We hebben dan nogal wat keuze voor de subnetten, we kiezen de gemakkelijkste oplossing : op de bytegrens → subnetmasker = 255.255.255.0.

Dit maakt de subnetten direct leesbaar als **3<sup>e</sup> byte** in een IP-adres.

- A1 : 50 → SN1, bv. 140.121.1.0
  - Router A intf A1 = 140.121.1.1, host A = 140.121.1.2, host B = 140.121.1.254<sup>1</sup>
  - Broadcasten op subnet A1 doet u door te sturen naar IP = 140.121.1.255
- A2 : 20 → SN2, bv. 140.121.2.0
  - Router A intf A2 = 140.121.2.1, hosts = 140.121.2.2, tot = 140.121.2.254
  - Broadcasten op subnet A2 doet u door te sturen naar IP = 140.121.2.255

<sup>1</sup> We duiden het eerste en laatst mogelijke IP adres aan op dit net. Kies unieke adressen tussen deze 2 waarden.

- B1 : 30 → SN3, bv. 140.121.3.0
  - Router B intf B1 = 140.121.3.1, host = 140.121.3.2, tot host C = 140.121.3.254
  - Broadcasten op net B1 doet u door te sturen naar IP = 140.121.3.255
- B2 : 15 → SN4, bv. 140.121.4.0
  - Router B intf B2 = 140.121.4.1, hosts = 140.121.4.2, tot = 140.121.4.254
  - Broadcasten op net B2 doet u door te sturen naar IP = 140.121.4.255
- AB : 2 IP adressen → SN5, bv. 140.121.5.0
  - Router A intf AB = 140.121.5.1, Router B intf AB = 140.121.5.2,

De enige interface die nu nog geen IP adres heeft is de link naar het internet van router A.  
Die moet een IP adres krijgen van het netwerk van de ISP, service provider, bv. telenet.

Vraag. Hoe stuurt **host A** een IP pakket naar **host B** ?

- Host A berekent zijn **eigen subnet** door de AND functie tussen zijn IP adres en subnetmasker  
 $140.121.1.2 \#AND\# 255.255.255.0 = 140.121.1.0$
- A bekijkt het dest. IP adres van B = 140.121.1.254 en AND het met zijn subnetmasker  
 $140.121.1.254 \#AND\# 255.255.255.0 = 140.121.1.0 \rightarrow \text{idem} !! \rightarrow$
- A speelt dit door aan **laag 2** om dit af te leveren **BINNENIN** het eigen subnetwerk.

Vraag. Hoe stuurt **host A** een IP pakket naar **host C** ?

- Host A heeft reeds zijn subnet berekend : 140.121.1.0
- A bekijkt het dest. IP adres van host C = 140.121.3.254 en AND het met zijn subnetmasker.  
 $140.121.3.254 \#AND\# 255.255.255.0 = 140.121.3.0 \rightarrow \text{NIET idem} !! \rightarrow$   
A weet nu dat dit IP adres **niet** behoort tot **hetzelfde subnetwerk** (140.121.1.0).
- A stuurt het dan maar naar **router A** voor verdere routering. Hoe? Router A heeft een interface in netwerk A1 en is dus bereikbaar voor host A via **laag 2** (afleveren BINNENIN het eigen netwerk). Merk hierbij wel op dat het **dest. IP adres** (L3) het uiteindelijke bestemmingsadres van **host C** is.
- Router A **zoekt** de weg naar (het netwerk van~) host C (bestemmingsadres) en vindt dat dit te bereiken is **via router B**. Router A stuurt het via netw AB naar router B. (aflevering via **L2**)
- Router B zoekt de weg naar host C en ziet dat een van zijn interfaces ook in dit netwerk 140.121.3.0 ligt. Hij laat het door L2 afleveren via deze interface.

OPM. de manier waarop routers hun routeringstabellen opstellen en dus uiteindelijk de ligging van alle netwerken kennen valt buiten het bestek van dit werk.

Vraag. Hoe stuurt **host A** een IP pakket naar **host D** ?

- Host A heeft reeds zijn subnet berekend : 140.121.1.**0**
- A bekijkt het dest. IP adres van host D bv. = 198.12.25.4 en AND het met zijn subnetmasker.  
 $198.12.25.4 \text{ AND } 255.255.255.0 = 198.12.25.0 \rightarrow \text{NIET idem !!} \rightarrow$
- A stuurt het dan maar naar **router A** voor verdere routering. (Router A heeft een interface in netwerk A1 en is dus bereikbaar voor host A via laag 2, afleveren BINNENIN het eigen netwerk).  
Het **dest. IP adres (L3)** = het uiteindelijke bestemmingsadres van **host D**.
- Router A **zoekt** de weg naar (het netwerk van~) host D (=bestemmingsadres) en vindt dat dit te bereiken is **via zijn ISP** verbinding, waarvoor hij een adres van een **router X** heeft.
- Router X zoekt de weg naar host D en levert het af aan de router die volgens hem op de kortste weg ligt naar host D.
- Dit proces herhaalt zich verschillende keren op de tussenliggende routers. Uiteindelijk ziet de router die voor het netwerk van host D staat dat een van zijn interfaces ook in dit netwerk 198.12.25.0 ligt.
- Hij laat het door L2 afleveren via deze interface.

### OEFENING 6.

Los de figuur op met sub- netwerken van **zo klein mogelijke volle klasnetwerken**.

Plaats op de figuur de gepaste IP adressen

Site A : A1 bezit 50 hosts

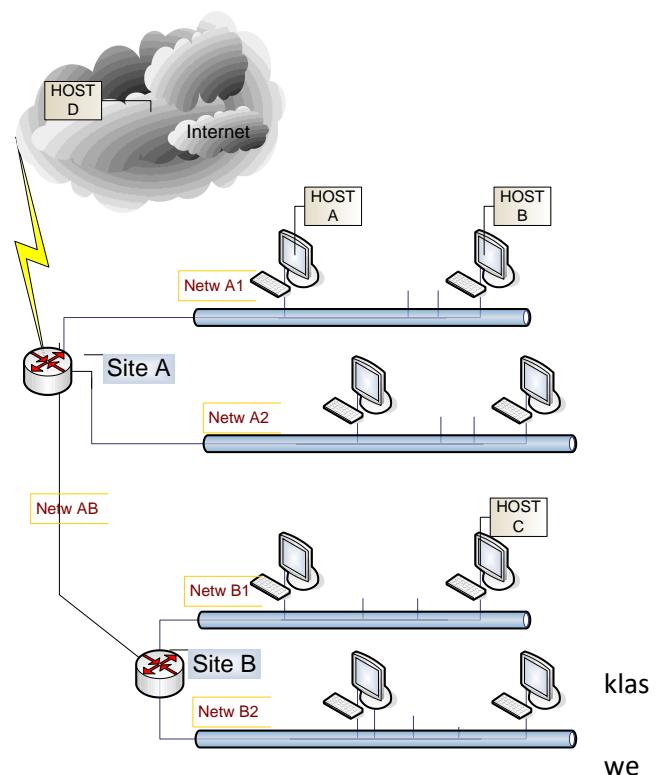
A2 bezit 20 hosts

Site B : B1 bezit 30 hosts

B2 bezit 15 hosts

### OPLOSSING.

Een oplossing met een klasse B netwerk is nogal 'vrijgevig'. We kunnen ook zonder meer volle netwerken met subnetwerken naast en door elkaar gebruiken. 'Ordelijk' zou kunnen zijn dat voor A1 en A2 1 kl C nemen, en voor B1 – B2 een 2<sup>e</sup>.



klas  
we

Wat dan met subnet AB? Bv. in 1 van de 2 klassen.

**Technisch** een stuk **beter** is dat we de grootste subnetten (bv. 50 en 30) het 'grootste subnetmasker' geven zodat ze nog een pak ruimte krijgen voor uitbreidingen. Je hoeft de subnetten van 1 klasse C niet per se (geografisch) te groeperen. Je kan die toewijzen aan gelijk welk subnet. De routers vinden de weg wel naar de subnets, dat zien we later in 3NWA. We gaan voor deze oplossing.

We kiezen 1 kl C voor A1 (50) en B1 (30) , een tweede kl C voor al de rest.

**1<sup>e</sup> KL C** .We gebruiken een max. subnet voor A1 en B1 : 255.255.255.**128**, waardoor er 7 bits over zijn voor de hosts in A1 en B1 → tot 125 hosts mogelijk ( $2^7=128 - 2 - 1$  v. router).

**2<sup>e</sup> KL C** .In de andere klasse C moeten we **3 subnets** voorzien van max 20 hosts.

→ Dat betekent minstens **2 bits** voor de **subnets**, minstens 5 bits voor de hosts.

We hebben in een klasse C 8 bits ter beschikking (volle klasse hostid) → bv. SNM=255.255.255.**192** (11...11 1100 0000) → **2 bits subnetid, 6 bits hostid**.

(Ook mogelijk was bv SNM=255.255.255.224 (...11 1110 0000) → **3 bits subnetid, 5 bits hostid**).

- A1 : 50 → kIC(1), bv. 200.21.0.**0**, SNM = 255.255.255.128, (...11 1000 0000)  
hierop kiezen we bv. SN1 → 200.21.0.**0** (...00 0000 0000) <sup>1</sup>

<sup>1</sup> We gebruiken dus wel degelijk **subnetid = √ 1<sup>L</sup>** en **subnetid = √ 0<sup>L</sup>**

- Router A intf A1 = 200.21.0.1, (...00 0000 0001)  
host A = 200.21.0.2 (...00 0000 0010) , host B = 200.21.0.126 (...00 0111 1110)<sup>1</sup>
- Broadcasten op subnet A1 doet u door te sturen naar IP = 200.21.0.127 (...00 0111 1111)
- A2 : 20 → kIC (2), bv. 200.21.1.0, SNM = 255.255.255.192, (...11 1100 0000)  
hierop kiezen we bv. SN1 → 200.21.1.0 (...01 0000 0000)
  - Router A intf A2 = 200.21.1.1 (...01 0000 0001) ,  
hosts = 200.21.1.2 (...01 0000 0010), tot = 200.21.1.62 (...01 0011 1110)
  - Broadcasten op subnet A2 doet u door te sturen naar IP = 200.21.1.63 (...01 0011 1111)
- B1 : 30 → kIC (1), van A1 200.21.0.0, SNM = 255.255.255.128, (...11 1000 0000)  
hierop kiezen we bv. SN2 → 200.21.0.128 (...00 1000 0000)<sup>2</sup>
  - Router B intf B1 = 200.21.0.129, (...00 1000 0001)  
hosts = 200.21.0.130 (...00 1000 0010) , tot = 200.21.0.254 (...00 1111 1110)<sup>3</sup>
  - Broadcasten op subnet B1 doet u door te sturen naar IP = 200.21.0.255 (...00 1111 1111)
- B2 : 15 → kIC (2), van A2 bv. 200.21.1.0, SNM = 255.255.255.192, (...11 1100 0000)  
hierop kiezen we bv. SN2 → 200.21.1.64 (...01 0100 0000)
  - Router B intf B2 = 200.21.1.65 (...01 0100 0001) ,  
hosts = 200.21.1.66 (...01 0100 0010), tot = 200.21.1.126 (...01 0111 1110)
  - Broadcasten op subnet B2 doet u door te sturen naar IP = 200.21.1.127 (...01 0111 1111)
- AB : 2 IP adressen → kIC (2), 200.21.1.0, SNM = 255.255.255.192, (...11 1100 0000)  
hierop kiezen we bv. SN3 →
  - Router A intf AB = 200.21.1.129 (...01 1000 0001)
  - Router B intf AB = 200.21.1.191 (...01 1011 1111)

Vraag. Hoe stuurt **host A** een IP pakket naar **host B** ?

- Host A berekent zijn eigen subnet door de AND functie tussen zijn IP adres en subnetmasker  
 $200.21.0.2 \text{ AND } 255.255.255.128 = 200.21.0.0$  (...00 0000 0010) #AND# (...11 1000 0000)
- A bekijkt het dest. IP adres van B = 200.21.0.126 en AND het met zijn subnetmasker  
 $200.21.0.126 \text{ AND } 255.255.255.128 = 200.21.0.0$  (...00 0111 1110) #AND# (...11 1000 0000) → idem !! → speelt dit door aan **laag 2** om dit af te leveren **BINNENIN** het eigen subnetwerk.

Vraag. Hoe stuurt **host A** een IP pakket naar **host C** ?

---

<sup>1</sup> We duiden het laatst mogelijke IP adres aan op dit net. Kies unieke adressen tussen deze 2 waarden.

<sup>2</sup> We gebruiken dus wel degelijk **subnetid =  $\forall 1^L$**  en **subnetid =  $\forall 0^L$**

<sup>3</sup> We duiden het laatst mogelijke IP adres aan op dit net. Kies unieke adressen tussen deze 2 waarden.

- Host A heeft reeds zijn subnet berekend : 200.21.0.0
- A bekijkt het dest. IP adres van host C = 200.21.0.254 en AND het met zijn subnetmasker.  
 $200.21.0.254 \#AND\# 255.255.255.128 = 200.21.0.128$  (...00 1111 1110#AND# (...11 1000 0000))  
→ NIET idem !! → A stuurt het naar **router A** voor verdere routering.
- Router A **zoekt** de weg naar (het netwerk van~) host C (bestemmingsadres) en vindt dat dit te bereiken is **via router B**. Router A stuurt het via netw AB naar router B. (aflevering via **L2**)
- Router B zoekt de weg naar host C en ziet dat een van zijn interfaces ook in dit netwerk 200.21.0.128 ligt. Hij laat het door L2 afleveren via deze interface.

Vraag. Hoe stuurt **host A** een IP pakket naar **host D** ?

- Host A heeft reeds zijn subnet berekend : 200.21.0.0
- A bekijkt het dest. IP adres van host D bv. = 198.12.25.4 en AND het met zijn subnetmasker.  
 $198.12.25.4 \#AND\# 255.255.255.128 = 198.12.25.0$  → NIET idem !! →  
A stuurt het dan maar naar **router A** voor verdere routering... enz.

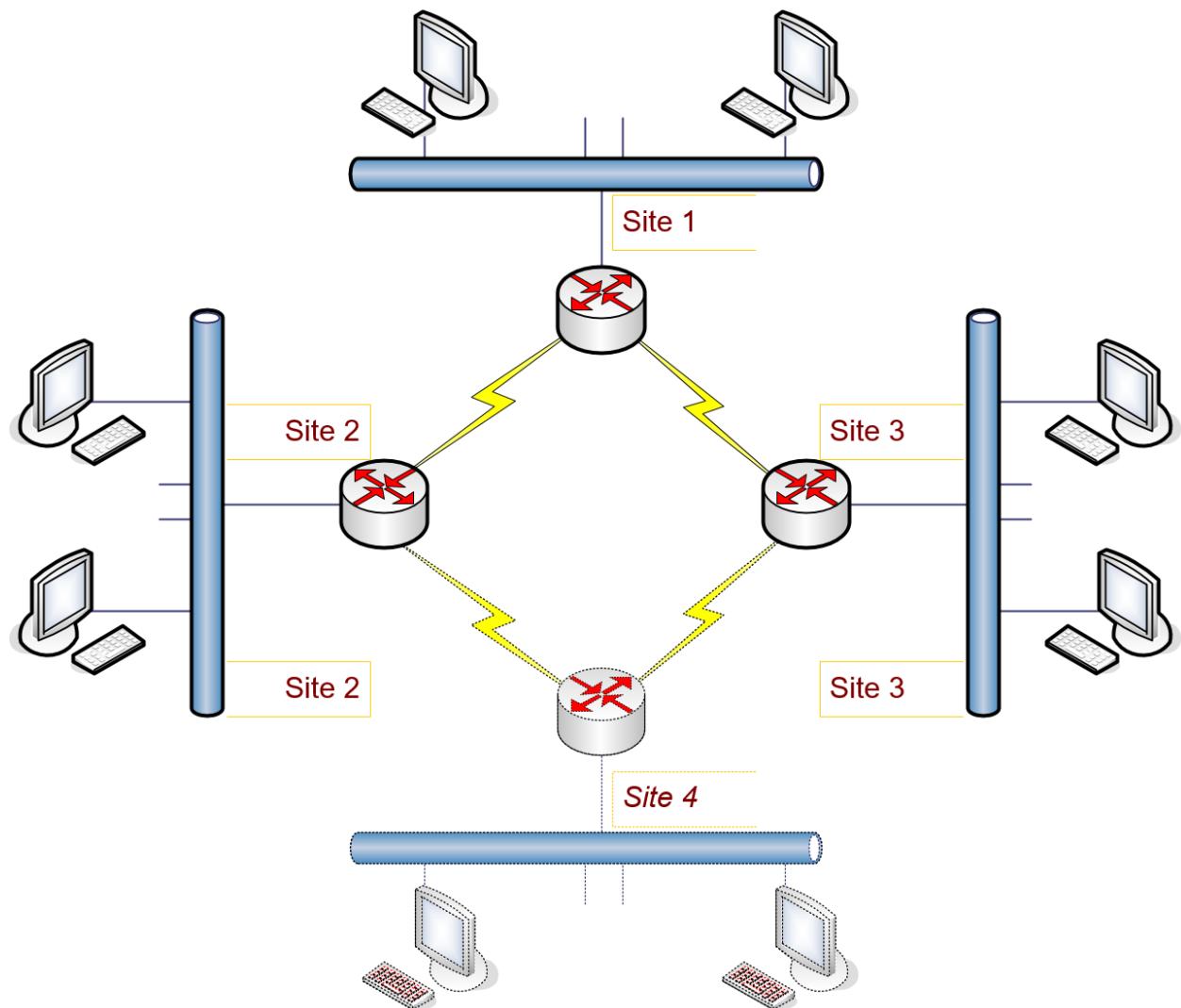
### 3.3.5 IP VLSM SUBNET – ADRESSEERING

#### ♦ Oefening 7.

Sinds RFC 1878 (1996) moet het subnetmasker **niet identiek** zijn op alle subnetten in het totale netwerk!!!  
 → VLSM : ‘Variable Length Subnet Masks’. (Niet ondersteund door RIP-1 ... → RIP-2, OSPF)

Stel : uw bedrijf heeft 3 sites verspreid over België die via TCP/IP moeten verbonden worden.

- Site 1 : heeft 24 hosts, (22 PC's en 2 servers) waarvan geen groei verwacht wordt,
- Site 2 : heeft 30 hosts, maar men plant een uitbreiding van deze site op korte termijn ... 2 jaar.
- Site 3 : heeft 49 hosts die op korte termijn een 4-tal hosts toegevoegd krijgt, maar dan zeker 5 jaar stabiel zal blijven.
- *Optie* : op middellange termijn (<3j) kan een bedrijf worden opgekocht wat een 4<sup>e</sup> site zal betekenen met 20 PC's. Deze site heeft op lange termijn max. 26 hosts.



Figuur 287 Oefening op IP-subnetting

Gegeven is **1 !!! klasse C** adres, bvb. 195.204.73.0. Stel een adresseringsschema voor dat zo flexibel mogelijk en toekomstgericht is, volgens de normale richtlijnen, t.t.z. subnet's & hosts ≠ √1's, ≠ √0's.

**Oplossing oefening 7.**

Merk op dat elke PPP link een subnet voorstelt, met enkel 2 hosts per link, de routers. Begin dus vooraan in uw adresseringsgebied met de kleine subnetten toe te kennen, dit om het verlies van subnet =  $\forall 0^L$  te minimaliseren. Het kleinste mogelijke subnetmasker is ... . 252 (= ... . 1111 1100) of /30.

Hierop kunnen juist **2** bruikbare IP adressen worden geconfigureerd.

- Klasse C : 195.204.73.0, met subnetmask 255.255.255.252 op de PPP links :

SNM: 1... ... 1.1111 1100 = /30

- PPP1 = ... . 0000 0100 = 195.204.73.4<sup>1</sup>, met hosts 195.204.73.5 en 195.204.73.6
- PPP2 = ... . 0000 1000 = 195.204.73.8, met hosts 195.204.73.9 en 195.204.73.10
- PPP3 = ... . 0000 1100 = 195.204.73.12, met hosts 195.204.73.13 en 195.204.73.14
- (PPP4 = ... . 0001 0000 = 195.204.73.16, met hosts 195.204.73.17 en 195.204.73.18)
- [PPP5 = ... . 0001 0100 = 195.204.73.20, met hosts 195.204.73.21 en 195.204.73.22]

PPP5 en volgende (bvb tot PPP7) zijn voorzien voor uitbreidingen, bvb. de optie.

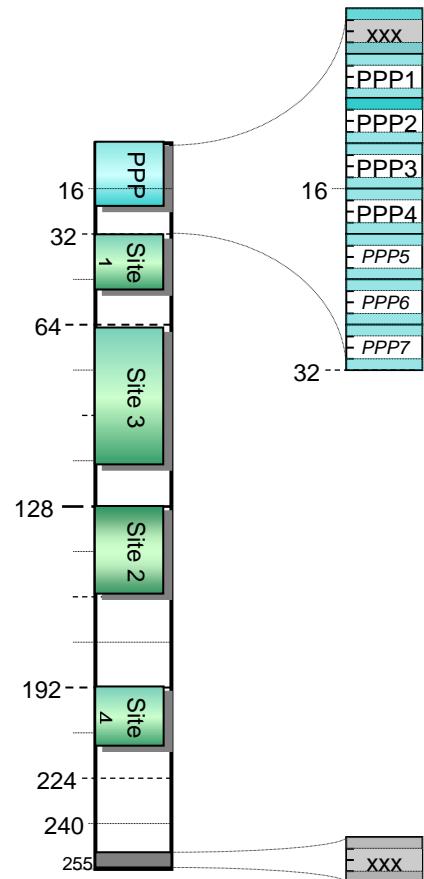
De volgende stap is om onze sites zo intelligent mogelijk in de "IP adres-map" te plaatsen. We bekijken de huidige toestand.

Subnetmaskers moeten (normaal) bitsgewijs toegekend worden. Grafisch betekent dit dat de adresblokken geen 'hogere' bitgrens mogen overschrijden.

Dit betekent dat site 3 (en 2) die meer dan 32 adressen vragen niet na de PPP subnetten mag geplaatst worden. Een betere oplossing is om site 1, die slechts 24 hosts vraagt, hier te plaatsen. Hij kan zodoende nog 6 plaatsen groeien → 30.

Site 3, met 49 hosts en site 2, met 30 hosts, kunnen dan naar keuze vanaf 64 of vanaf 128 geplaatst worden. Ze kunnen beiden groeien tot 62 hosts.

Figuur 288 IP variabel subnetmasker toekenning.



- Klasse C : 195.204.73.0, toewijzing (= start) van de sites.

- SITE1 : SNM = /27 : ... . 1110 0000 = 255.255.255.224  
en IP adres ... . 0010 0000 = 195.204.73.32, met hosts 195.204.73.33 tot .62
- SITE3 : SNM = /26 : ... . 1100 0000 = 255.255.255.192  
en IP adres ... . 0100 0000 = 195.204.73.64, met hosts 195.204.73.65 tot .126

<sup>1</sup> Subnet  $\forall=0^L$  wordt niet gebruikt!

<input type="checkbox"/> SITE2 : SNM = /26 :	... . 1100 0000 = 255.255.255.192 en IP adres ... . 1000 0000 = 195.204.73. <b>128</b> , met hosts 195.204.73. <b>129</b> tot <b>.190</b>
<input type="checkbox"/> SITE4 : SNM = /27 :	... . 1110 0000 = 255.255.255.224 en IP adres ... . 1100 0000 = 195.204.73. <b>192</b> , met hosts 195.204.73. <b>193</b> tot <b>.222</b>
<input type="checkbox"/> (SITE5 : SNM = /28 :	... . 1111 0000 = 255.255.255.240 ) en IP adres ... . 1110 0000 = 195.204.73. <b>224</b> , met hosts 195.204.73. <b>225</b> tot <b>.238</b>
...	

We moeten achteraan de map zeker nog een adresveld vrijlaten voor de 'all subnet' broadcast 195.204.73.**255**. Maar er zijn nog een aantal kleine sites (zoals site 5) toe te kennen in dit gebied, bvb. PPP links.

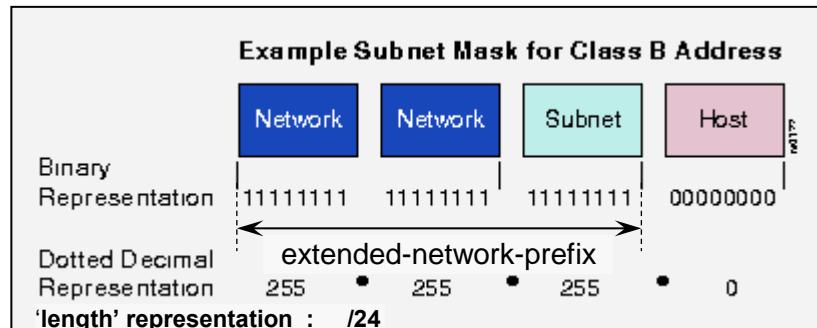
Breng deze gegevens over op Figuur 287 en geef de routers een IP adres.

Teken ook de PPP links met router 4 en ken de IP adressen toe.

**Vraag?** Moest er geen site 4 (20 hosts) verwacht worden, maar daarentegen een sterke groei van site 3, kan dan de onderliggende positie van site 2 en 3 niet beter omgedraaid worden, waardoor site 3 meer groeimogelijkheden heeft? ... site 2 → 62, site 3 → 126! <sup>1</sup>

**OPM 1.** We hebben hier ook het 'extended-network-prefix-length' notatie gebruikt i.p.v. het subnetmasker.

Deze lengte komt overeen met het aantal 1'en in het subnetmasker.



Figuur 289 Extended-network-prefix-length voorstelling.

Dit betekent dat een netwerkadres 134.4.5.36 met een subnetmasker van 255.255.255.0 kan worden voorgesteld als 134.4.5.36/**24**. Het is een compactere voorstelling en voor lezen beter voor te stellen.

Beide voorstellingen worden door elkaar gebruikt. Merk op dat de routing-protocollen nog altijd het 4 byte subnetmasker oversturen, i.p.v. bvb. een 1 byte lengte.

Op deze manier worden de klasse A netwerken ook wel eens "/8" ("slash eights") genoemd, klasse B → /16 en klasse C → /24.

**OPM2.** Je kan ook oefening 5 en oefening 6 beter oplossen door VLSM te gebruiken.

<sup>1</sup> NEEN, site 3 zou dan een subnetmasker ... . 128 nodig hebben, en dit mag niet gebruikt worden : het heeft als subnetadres alleen  $\forall 1^L$  en  $\forall 0^L$  mogelijk.

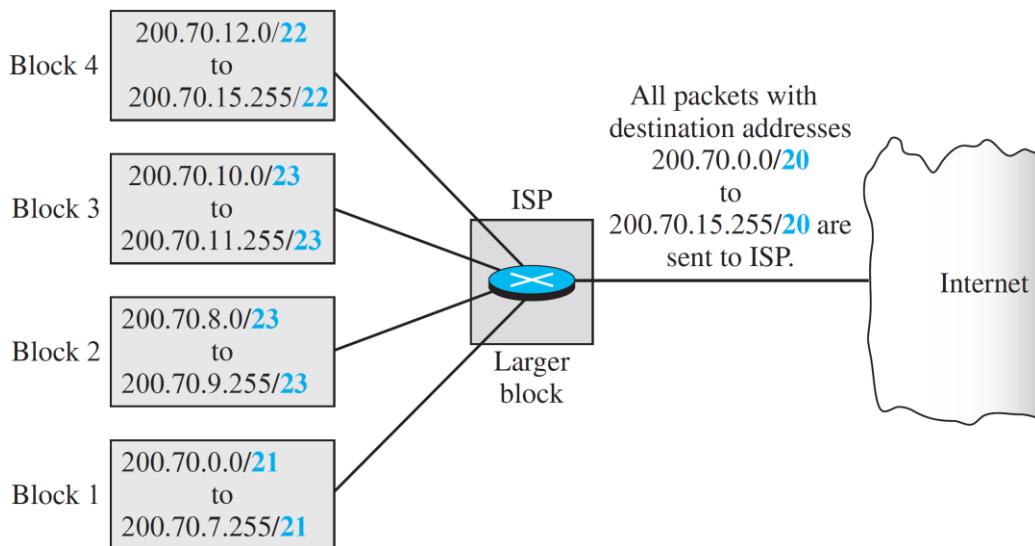
### 3.3.6 CIDR CLASSLESS INTERDOMAIN ROUTING

Een uitbreiding van het subnet-principe is Classless InterDomain Routing, **CIDR** (1993). Hierbij wordt het **omgekeerde** uitgevoerd : bv. meerdere klasse C netwerken worden **samengevoegd** om 1 groter netwerk te bekomen. Dit gebeurt omdat klasse B netwerken ‘uitverkocht’ zijn of heel moeilijk verkrijgbaar. Het wordt ook wel eens ‘**supernetting**’ genoemd.

Meerdere klasse C adressen leiden echter tot enorme routing tabellen op het internet aangezien ze allen een ‘entry’ vereisen. De bedoeling is nu om meerdere IP-klassen (bv. meerdere klasse C’s, er zijn er 2M...) samen te voegen tot een grotere entiteit terwijl ze toch maar 1 ‘entry’ in de routing tabellen vereisen.

Als bv. één enkele site een 2000-tal computers bevat en bestaat uit **8** (opeenvolgende) **klasse C** netwerken die allen via dezelfde internet provider een connectie tot het internet bezitten zouden ze moeten kunnen gerouteerd worden door **1 entry**. De exponentiële groei van de routertabellen kan op die manier worden gestopt. Dit wordt ook wel ‘route aggregatie’ genoemd.

Oplossing : **masker** op het **netid** plaatsen i.p.v. het hostid → ‘**variable sized networks**’, het masker bepaalt de netwerk-grootte. Hieronder worden resp. 8, 2, 2 en 4 klasse C’s samengevoegd. De ISP beheert deze adressen en wordt zelf gerouteerd als een /20.



Figuur 290. CIDR toewijzing van een /20 aan een ISP

Hier zijn wel stricte voorwaarden aan verbonden. Het zijn steeds machten van  $2 + \dots$  :

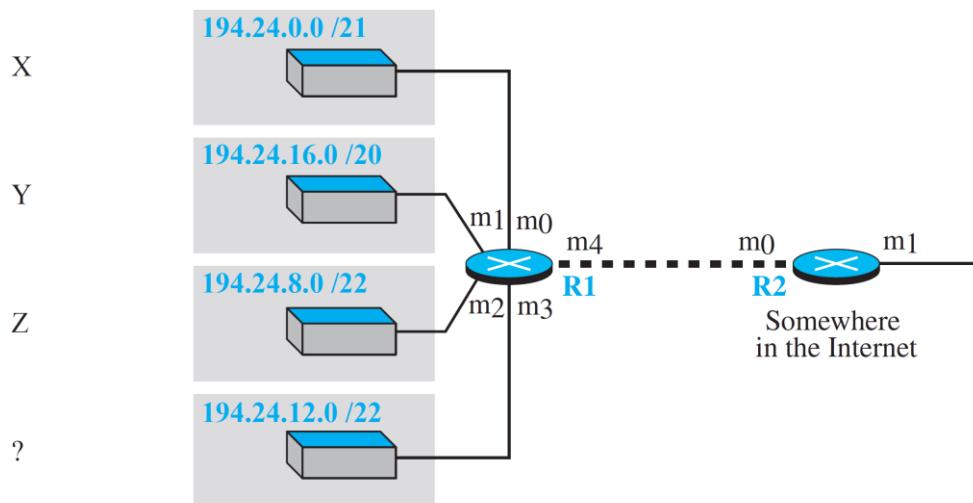
1. de samen te voegen klassen moeten **dezelfde hogere orde bits** hebben in hun IP adres.
2. de routers en hun algorithmes moeten aangepast worden om hun beslissingen te baseren op een 32 bit IP adres mét een 32 bit masker
3. de router-protocols (het onderhoud van de routertabellen) moeten dit masker ook ondersteunen. RIP-2 en OSPF ondersteunen dit.

Bv.

- Stel user X vraagt 2048 adressen aan → **8** klasse C's.  
Hij krijgt toegewezen 194.24.0.0 tot 194.24.7.255 (opeenvolgende adressen).
- Y vraagt 4096 adressen → **16** klasse C's. Ze moeten op bit-niveau een te decoderen **blok** vormen ('gelijke hogere orde bits') daarom krijgt hij **niet** 194.24.8.0 e.v. (tot 194.24.23.255), zie verder, maar 194.24.16.0 t/m 194.24.31.255.
- Z vraagt 1024 adressen (4xC) en krijgt wel 194.24.8.0 t/m 194.24.11.255.

X : 2048 adr. → 194.24.0.0 tot 194.24.7.255	1100 0010. 0001 1000. 0000 0000. 0000 0000 1100 0010. 0001 1000. 0000 0111. 1111 1111 <b>1111 1111. 1111 1111. 1111 1000. 0000 0000</b>
Y : 4096 adr. → 194.24.16.0 tot 194.24.31.255	1100 0010. 0001 1000. 0001 0000. 0000 0000 1100 0010. 0001 1000. 0001 1111. 1111 1111 <b>1111 1111. 1111 1111. 1111 0000. 0000 0000</b>
Z : 1024 adr. → 194.24.8.0 tot 194.24.11.255	1100 0010. 0001 1000. 0000 1000. 0000 0000 1100 0010. 0001 1000. 0000 1011. 1111 1111 <b>1111 1111. 1111 1111. 1111 1100. 0000 0000</b>
Masker: ... /21	
Masker: ... /20	
Masker: ... /22	

(De ISP heeft nog een /22 ter beschikking in zijn 194.24.0.0/19 bereik: 194.24.12.0/22).



Forwarding table for R1

Network address/mask	Next-hop address	Interface
194.24.0.0 /21	-----	m0
194.24.16.0 /20	-----	m1
194.24.8.0 /22	-----	m2
194.24.12.0 /22	-----	m3
0.0.0.0/0	address of R2	m4

Forwarding table for R2

Network address/mask	Next-hop address	Interface
194.24.0.0 /19	-----	m0
0.0.0.0/0	default router	m1

Figuur 291. CIDR router tabel

OPM. Y krijgt **niet** 194.24.8.0 /20 toegewezen ( ➔ tot 194.24.23.255).

Dat zou betekenen:

Y : 4096 adr. ➔ 194.24.8.0 tot 194.24.23.255	1100 0010. 0001 1000. 0000 1000. 0000 0000 1100 0010. 0001 1000. 0001 0111. 1111 1111
Masker	1111 1111. 1111 1111. 1111 0000. 0000 0000

... GEEN eenduidige bits in het NETID !!

VRAAG. Hoe zoekt een router nu het netwerk van een IP pakket met dest.adres 194.24.17.4 ?

De vergelijking met de **netwerkadressen** in zijn routertabellen moet **identiek zijn tot de maskergrens**.

Network address/mask	Leftmost bits in the destination address	Next hop	Interface
194.24.0.0 /21	<b>1100 0010. 0001 1000. 0000 0</b>	C	m0
194.24.16.0 /20	<b>1100 0010. 0001 1000. 0001</b>	C	m1
194.24.8.0 /22	<b>1100 0010. 0001 1000. 0000 10</b>	C	m2
194.24.12.0 /22	<b>1100 0010. 0001 1000. 0000 11</b>	C	m3
0.0.0/0	Default	IP-adres van R2	m4

Figuur 292. Route tabel van R1 in detail.

➔ Hij voert een AND ftie uit op het IP-adres tot netwerkmasker . Bvb.

194.24.17.4 ?	Adres	1100 0010. 0001 1000. 0001 0001. 0000 0100
➔ =X ?	netadres	1100 0010. 0001 1000. 0000 0 ➔ NeeN
➔ =Y ?	netadres	1100 0010. 0001 1000. 0001 ➔ Ja ➔ Y router
➔ =Z ?	netadres	1100 0010. 0001 1000. 0000 10 ➔ NeeN

OPM. De 'C' in de routertabellen onder next hop staat voor 'Connected' , m.a.w. direct af te leveren bij de bestemming.

Bijkomende informatie : RFC **1519**, 1518, 1517, 1520, 1467 en 1466.

Middellange termijn-oplossing : NAT, zie p. 263.

Lange termijn-oplossing : IPnG (next Generation) of IPv6.

Vanaf nu zijn de toewijzingsregels voor de adressen in klasse C ook veranderd → op **geografische** basis worden wereldwijd 4 zones gecreëerd :

- Europa : 194.0.0.0 tot en met 195.255.255.255
- N.Amerika 198.0.0.0 tot en met 199.255.255.255
- Mid. en Zd Amerika 200.0.0.0 tot en met 201.255.255.255
- Azie 202.0.0.0 tot en met 203.255.255.255

Op deze manier heeft elke regio nog **130 000** C-netwerken of 32 miljoen adressen gekregen. Bovendien zijn de adressen 204.0.0.0 tot 223.255.255.255, = 320 miljoen adressen, gereserveerd voor de toekomst. Het voordeel is dat nu een internet-gateway die een pakket krijgt met 195.xxx.yyy.zzz het naar de **standaard gateway** voor **Europa** kan sturen, een idee afkomstig van IPv6.

Voor meer informatie betreffende CIDR en het zorgvuldiger omspringen met IPv4 adressen, zie appendix C.

### 3.3.7 IP INTRANETTEN EN NETWORK ADDRESS TRANSLATION.

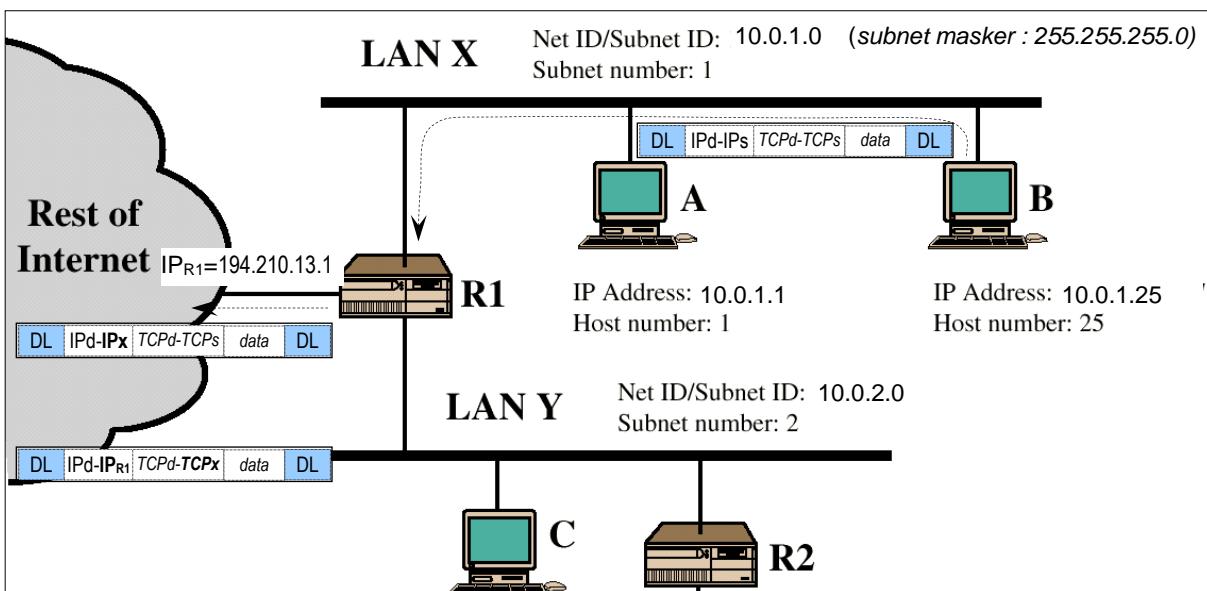
Deze oplossing voor het nijpend tekort aan IPv4 adressen is reeds uitvoerig(er) besproken in de cursus 2 computernetwerken. We geven enkel een korte resume.

Zoals reeds eerder vermeld zijn er een aantal IP adressen voor prive-gebruik : (RFC 1918, 1627)

- **10.0.0.0** tot **10.255.255.255** : een volledig klasse A bereik
- **172.16.0.0** tot **172.31.255.255** : 16 klasse B bereiken
- **192.168.0.0** tot **192.168.255.255** : 256 klasse C bereiken

Ze worden niet gerouteerd door de internetrouters.

Dit principe wordt gebruikt als oplossing voor het **gebrek aan IP adressen** (in afwachting van IPv6?). Stel dat een netwerk van KdG-IWT niet de beschikking krijgt over een klasse B adres (uitverkocht). Het is echter weinig waarschijnlijk dat de meer dan duizend hosts **tegelijk** een connectie willen met het Internet. Door toekenning van 1 klasse C adres (bvb. **194.210.13.x**) zal de internet-gateway een '**adres-translatie-tabel**' bijhouden en de intranetadressen (='**privé**'-adressen, bvb. **10.0.1.25**) in de IP pakketten vervangen door een **wereldwijd uniek** klasse C internet adres (='**publiek**'-adres). Zo kunnen toch 253 hosts tegelijk wereldwijd communiceren. Dit principe wordt NAT, Network Address Translation genoemd (RFC 1631). Indien niet aan de eisen voldaan wordt kan men overgaan naar PAT, Port Address Translation, waar de volledige 'socket' (=IP adres + TCP-port nummer) wordt vertaald.



Figuur 293 IP network address translation

Bvb. host B, met intranet IP = 10.0.1.25, communiceert met het internet bvb. IPd = 204.12.23.2.

Het pakket heeft in het **intranet** als IP header ; IPd=204.12.23.2 – IPs= 10.0.1.25.

Internet gateway R1 stuurt het echter op het **Internet** met een ander source IP adres : bvb. IPx = **194.210.13.201**. Hij houdt hier bovendien een tabel van bij om reply's te kunnen afleveren.

Een speciale mode hiervan is om alle hosts te verbergen achter **1 publiek IP adres** en de uitsplitsing te doen in de **TCP** header d.m.v. het poortnummer, ook hide mode of **PAT** = 'Port Address Translation' genoemd.

(Van de 65536 beschikbare poortnummers worden uiteraard de eerste 1024 niet gebruikt.) In dat geval zal hetzelfde intranetpakket worden ‘heringepakt’ met dat enige publieke IP adres van R1 maar zal de ‘source port’ vervangen worden door een nieuwe port TCPx die de gateway opslaat in een tabel in het geheugen om terugkerende boodschappen te kunnen afleveren. Deze boodschappen zien er identiek uit als de getekende, buiten het feit dat source- en destination-adressen (en -ports) zijn omgewisseld.

NAT (PAT) heeft bovendien een ‘**security**’ voordeel dat niet te verwaarlozen is. Vele mensen zien een connectie met het internet als een 1-richtingsverkeer, maar dat is allesbehalve waar : het internet is net zo goed verbonden met uw computer die open staat voor aanvallen van buiten. NAT laat nu enkel connecties toe die van binnenuit worden opgezet. Bvb. kan een interne client een externe FTP server raadplegen, maar omgekeerd kan niemand aan de interne servers, tenzij dat zo is geconfigureerd voor bvb. welbekende poorten, bvb. FTP, = poort 21 naar een specifiek intern IP-adres.

Een ander gebruik van NAT is ‘traffic logging’. Aangezien alle trafiek naar het Internet deze gateway moet passeren kan hij dit ‘loggen’ naar een bestand.

Een nadeel aan NAT en vooral PAT is dat de headers van IP resp. TCP wijzigen waardoor de gateway opnieuw de checksums moeten herrekenen. Intelligent software kan deze extra processing echter minimaliseren waardoor de ‘throughput’ van de gateway weinig wordt beïnvloed.

Een ander nadeel is dat nu het IP pakket geen herkenning meer heeft naar de gebruikte (source) applicatie : het poortnummer is veranderd. Dit kan problemen geven in FW (Firewalls) of QoS mapping.

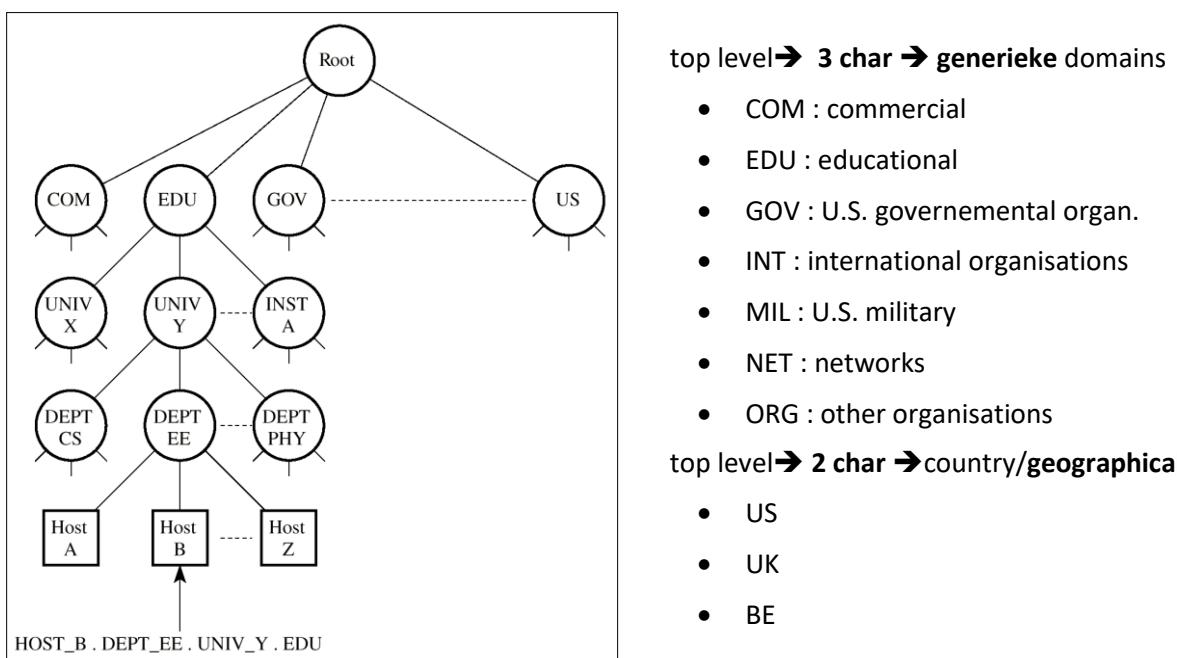
### 3.3.8 DOMAIN NAME SYSTEM.

Ook dit onderwerp is uitvoiger aan bod gekomen in de cursus 2-computernetwerken.

(Dit is eigenlijk een applicatie die draait óp UDP (TCP)/IP, maar aangezien dit de functie heeft om computernamen te vertalen in IP-adressen lijkt het ons gepast dit hier te vermelden.)

Mensen zijn beter vertrouwd met **namen** als hostid i.p.v. getallen. Daarvoor bestaat er een gedistribueerde database, het Domain Name System (DNS) die **hostnamen vertaalt naar IP adressen** en omgekeerd<sup>1</sup>. Gedistribueerd betekent dat niet alle servers de volledige database bevatten, er moet desnoods verder gevraagd worden.

De organisatie is hiërarchisch : de Root is ‘unnamed’ (of ‘.’), daaronder bevinden zich de ‘top level domains’, vervolgens ‘second level domains’, ... . (hoofd- of kleine letters maken geen onderscheid).



Figuur 294 De Domain Name Hierarchie

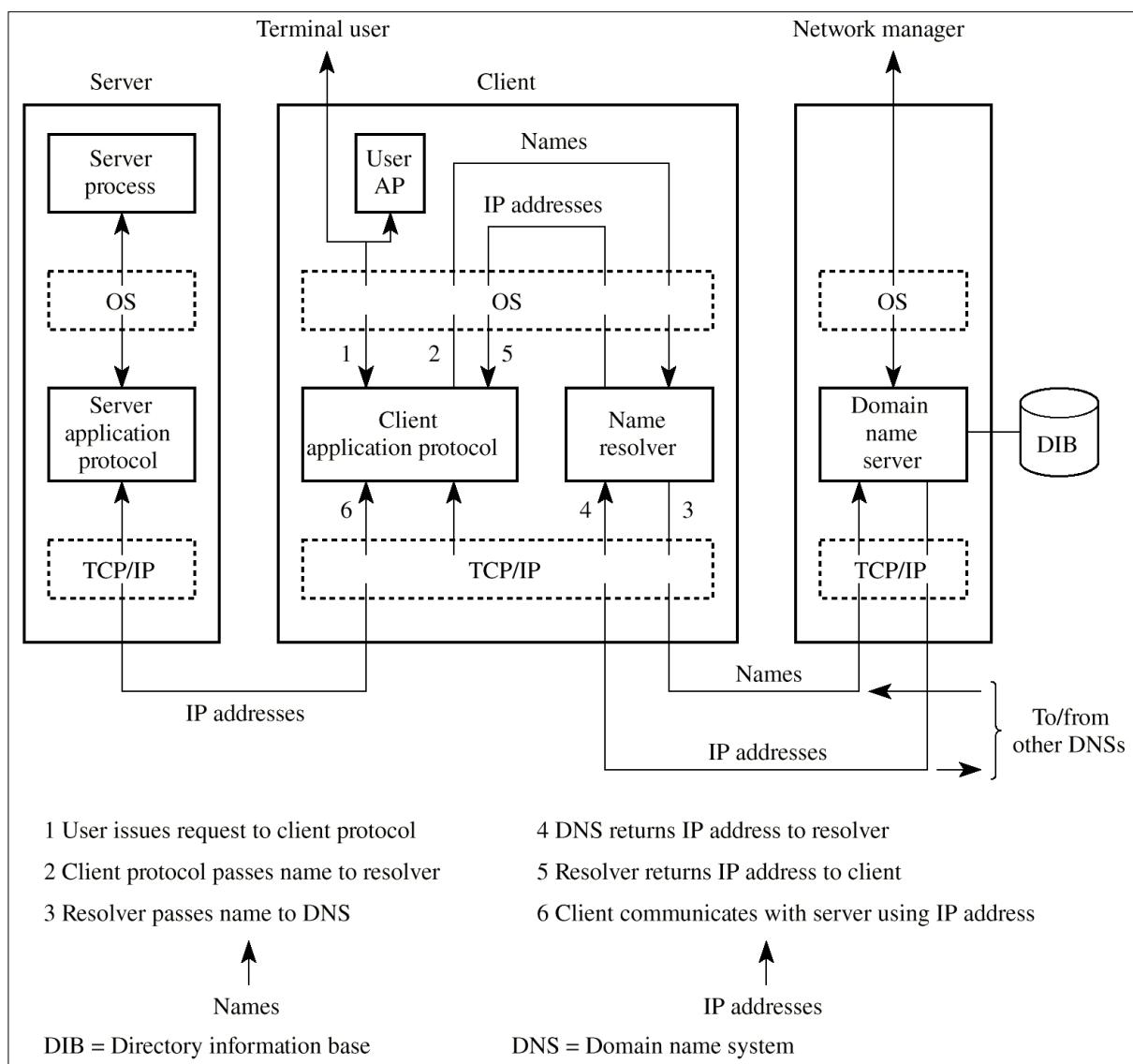
Vele landen vormen onder hun 2-karakter landcode zelf generieke domains : .ac.uk = academische instellingen in UK, .co.uk = commerciële ...

De root-DNS beheert de top levels, en deleert de rest. Zo is het domein .be in handen van de KUL, onder DNS.BE. Elk **netwerk**, vb. van een hogeschool, heeft bovendien (minstens) 1 **eigen name server**<sup>2</sup> (DNS) (en een verantwoordelijke) die de namen beheert onder zijn domain. Wordt een nieuwe host toegevoegd dan moet de database (DIB, Directory Information Base) aangepast worden. De bedoeling is nu van zo verstandig te delegeren dat de meeste aanvragen **lokaal** kunnen worden opgelost.

<sup>1</sup> Bvb. Alcatel’s WWW server : [www.alcatel.com](http://www.alcatel.com) ↔ 198.64.191.11 . (er is geen relatie tussen de ‘.’ en van beide notaties). Of de computer ‘krokus’ van dep. **thti** van **telindus** : [krokus.thti.telinfo.be](http://krokus.thti.telinfo.be) ↔ 194.7.50.221. Elke organisatie bepaalt zelf hoeveel segmenten er worden gebruikt en wat die betekenen.

<sup>2</sup> Kleinere organisaties hebben geen eigen DNS server maar vragen de ISP hun domein op te nemen in zijn DNS.

Als een gebruiker (of AP Applicatie Proces) een netwerktransactie wil uitvoeren met een node waarvan hij de **naam** kent, maar niet het IP adres, dan moet eerst het IP adres worden gevonden.



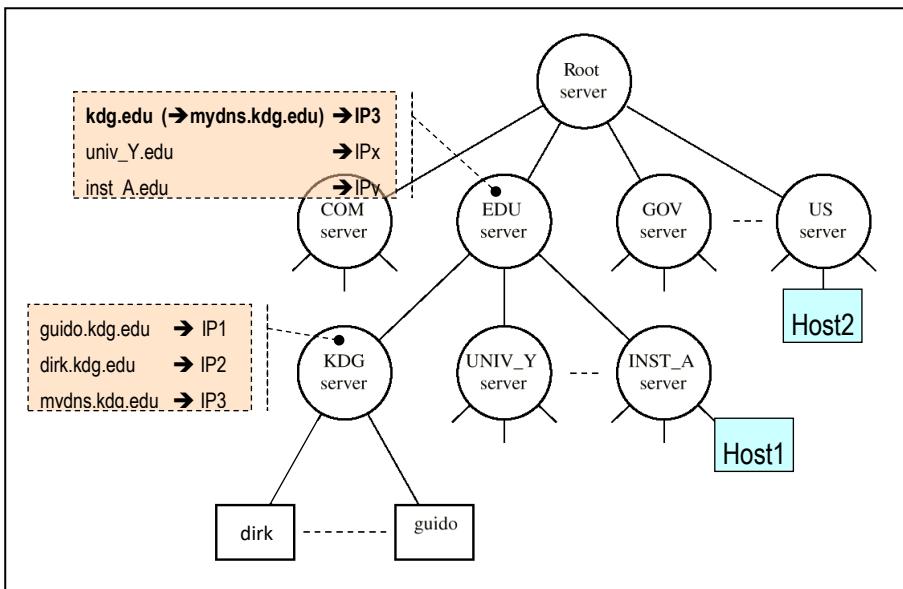
Figuur 295 Name to address resolution protocol

Hieroor draait bij elke host een proces : de ‘name resolver’ die de naam krijgt doorgespeeld. Op zijn beurt zal de resolver uitwendig via TCP/IP de vraag stellen aan de lokale DNS<sup>1</sup>. Gaat het om een gekende host (in de DIB, bv. alle plaatselijke hosts) dan wordt het IP adres hiervan doorgespeeld aan de resolver die het op zijn beurt doorspeelt aan de client die er oorspronkelijk om vroeg. De host kan nu **mét** het goede **IP adres** communiceren → ‘name to address resolution protocol’.

Als een nieuw domein, bvb. KdG, wordt toege-voegd, moet zijn DNS server, mydns.kdg.edu, opgenomen worden in de bovenliggende DNS servers DIB, hier EDU.

Zelf moet de system administrator zijn DNS servers DIB definiëren.

<sup>1</sup> Elke resolver is geconfigureerd met het adres van een lokale domeinnaamserver, en mogelijk een lijst met alternatieve servers voor het geval de eerste niet beschikbaar is. Een client kan kiezen tussen UDP en TCP voor de communicatie met een DNS-server, maar meestal gebruikt hij UDP omdat dit de minste overhead vraagt.



Figuur 296 Een nieuw domein toevoegen.

Als nu host1 een verbinding wil opzetten met guido.kdg.edu dan zal :

1. zijn resolver de vraag stellen aan zijn plaatselijke DNS server, INST\_A
2. Vindt de locale DNS (INST\_A) het adres niet in zijn DIB, dan stelt hij de vraag aan de hiërarchisch hoger gelegen EDU server. Dit noemt men een 'referral'.
3. Deze EDU server kent slechts het adres van de KDG.EDU server (IP3) en speelt dit door aan de INST\_A server.
4. De INST\_A server neemt nu contact op met de KDG.EDU (IP3) server met de oorspronkelijke vraag : wat is het IP-adres van guido.kdg.edu?
5. De KDG.EDU server antwoordt met het adres IP1 in een 'authoritative record' = een record afkomstig van de autoriteit die het record beheert en dus altijd correct is.

Dit authoritative record wordt opgeslagen in een cache geheugen van de locale DNS (INST\_A) om te vermijden dat deze procedure al te veel zou worden herhaald. Deze 'cache-entry's bevatten een 'time to live' veld om hosts toe te laten zich te verplaatsen en ze niet meer zouden bereikt kunnen worden. De TTL beperkt ook de aangroei van de name-cache. Het doel is uiteraard deze referral's te mini-maliseren. Op een vraag wordt eerst de lokale database gecontroleerd en vervolgens de name cache.

Als host2 een verbinding wil opzetten met guido.kdg.edu dan verloopt dit gelijkaardig:

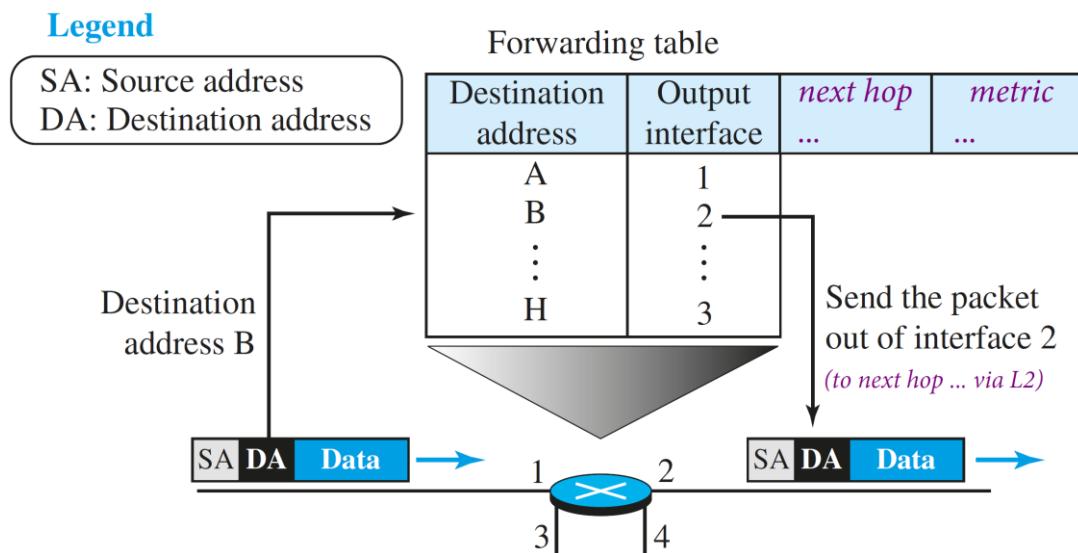
- 2a) Zijn DNS server, bvb. US, stelt de vraag aan de root server, die antwoordt met het adres van de EDU server.
- 2b) de US server stelt de vraag aan de EDU server ...

TEST zelf uit (bvb. op NT)      ...c:> NSLOOKUP    ... ➔

>[www.ping.be](http://www.ping.be), >[www.kdg.be](http://www.kdg.be), >set debug, >[www.mobistar.be](http://www.mobistar.be), >[www.microsoft.com](http://www.microsoft.com)  
(➔ 6 www servers die roteren.)

## 3.4 IP ROUTING.

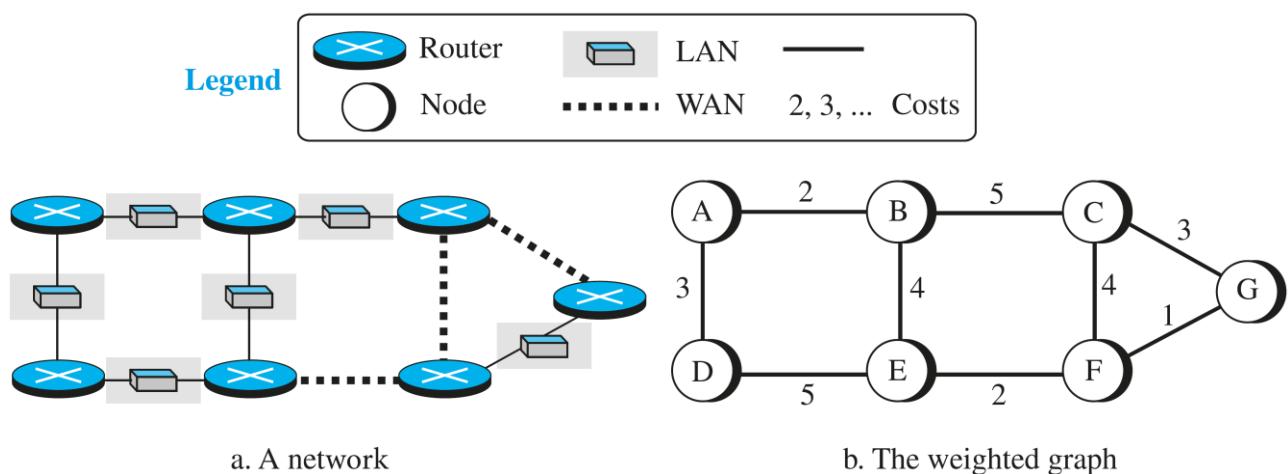
De belangrijkste functie die IP moet voorzien is: **Routing**. IP moet achterhalen welke **weg best** gevolgd wordt tot de bestemming, of m.a.w. '**least cost routing**'. De **routers** die de netwerken met elkaar verbinden berekenen deze **cost** op basis van een '**metric**'.



Figuur 297 Forwarding door routers.

Een IP pakket wordt gerouteerd '**hop by hop**' door een **forwarding table** in de hops of routers waarin een '**metric**' staat die de **cost** aangeeft om via deze interface de bestemming te bereiken. De routers moeten de 'beste' wegen zoeken naar de bestemming, = '**least cost**'.

Om een volledig overzicht te krijgen wordt een netwerk meestal voorgesteld door een '**graph**'.



Figuur 298. Graph voorstelling van een netwerk.

De cost die hierbij gerekend wordt voor een netwerk hangt af van welk protocol gebruikt wordt. Dit kan bv. zijn: *Hop count*, *Bandbreedte/snelheid van de interface*, *Delay*, ...

### 3.4.1 ROUTERINGSPRINCIPES

#### VOOR EEN HOST

... is routing simpel. IP ontvangt een datagram van TCP, UDP, ICMP of IGMP (m.a.w. lokaal gegenereerd in een bovenliggende laag ↓). Afh. v.h. bestemmingsadres zal hij:

1. De bestemming is **direct** verbonden aan **hetzelfde** netwerk → het datagram wordt ‘ingepakt’ op de datalink en naar de bestemming gestuurd. (ARP zorgt voor het MAC-adres) → **‘direct routing’**.
2. Zoniet zendt hij het datagram naar een ‘**default router**’, **de DGW**, die er dan maar voor moet zorgen dat het aankomt → **‘indirect routing’**.

Dit is ook terug te vinden in de **routing tabel** in het geheugen die hij onderzoekt elke keer er een datagram wordt ge‘processed’.

Bv. Met een IP adres van

- 192.168.182.128 /24

Is zijn **eigen netwerk** te vinden op:

- **192.168.182.0 255.255.255.0**  
op deze interface

En zijn ‘**Default gateway**’ :

- 0.0.0.0 0.0.0.0 → **192.168.182.2** !

```
Command Prompt
C:\Users\Guido>route print
=====
Interface List
  3...00 0c 29 90 10 a2 .....Intel(R) 82574L Gigabit Network Connection
  1.....Software Loopback Interface 1
=====

IPv4 Route Table
=====
Active Routes:
Network Destination      Netmask        Gateway       Interface Metric
          0.0.0.0      0.0.0.0    192.168.182.2  192.168.182.128    25
          127.0.0.0    255.0.0.0   On-link        127.0.0.1     331
          127.0.0.1  255.255.255.255  On-link        127.0.0.1     331
          127.255.255.255 255.255.255.255  On-link        127.0.0.1     331
          192.168.182.0    255.255.255.0   On-link      192.168.182.128    281
          192.168.182.128 255.255.255.255  On-link      192.168.182.128    281
          192.168.182.255 255.255.255.255  On-link      192.168.182.128    281
          224.0.0.0      240.0.0.0   On-link        127.0.0.1     331
          224.0.0.0      240.0.0.0   On-link      192.168.182.128    281
          255.255.255.255 255.255.255.255  On-link        127.0.0.1     331
          255.255.255.255 255.255.255.255  On-link      192.168.182.128    281
=====
```

De rest van de entry’s zijn loopback adressen (127.0.0.0 en .1), multicasts (224.0.0.0/4) en de lokale broadcast (255.255.255.255/32) waar hij naar moet luisteren.

. of het ontvangt er een via een netwerkinterface ↑. De IP laag heeft nu.

Bij **ontvangst** van een datagram via de netwerkinterface wordt in dezelfde tabel gecontroleerd of het adres zijn eigen IP adres is of een multi/broadcast adres is waar hij naar luistert.

- **Zoja** dan wordt het datagram afgeleverd aan het protocol dat gespecificeerd staat in de IP header.
- **Zoneen**, dan wordt het datagram ge ‘dropped’.

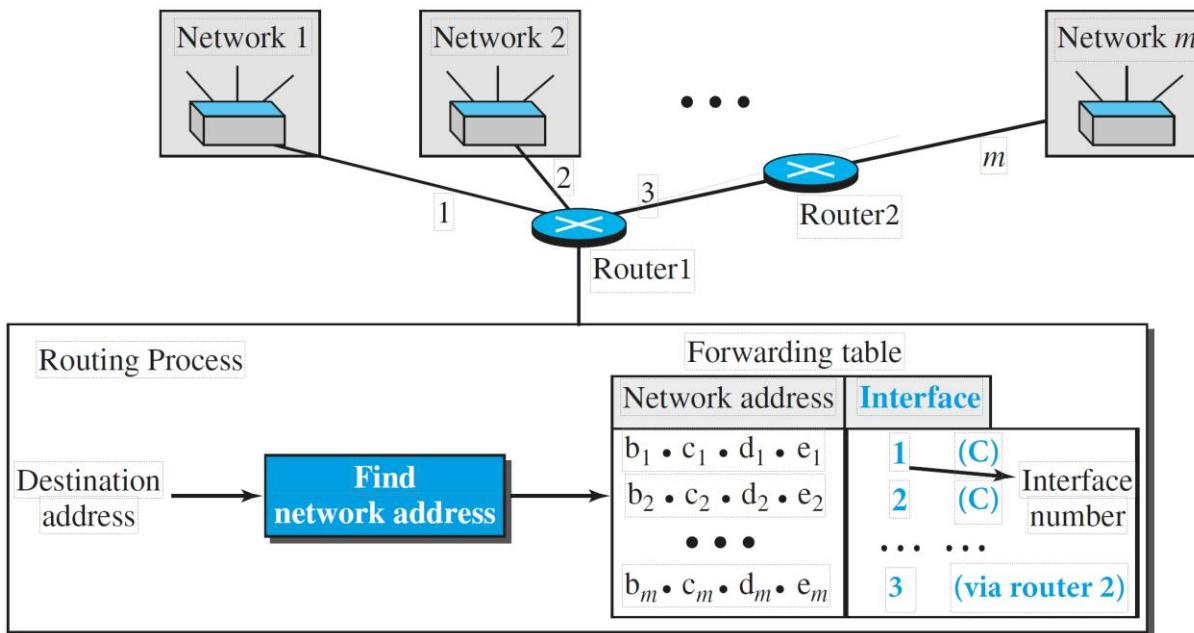
Elke routing tabel ‘entry’ bevat :

- *Destination IP adres* : dit kan zowel een host- als een netwerkadres zijn.
- *IP adres van een next-hop router* : deze kan het datagram verder doorsturen naar zijn bestemming.
- *Langs welke interface moet uitgestuurd worden*.
- *Cost* : de route metric.

## VOOR EEN ROUTER

We weten dat een router enkel de **wegen naar de netwerken** kent / opzoekt.

Hieronder een router die verbonden is aan m (sub-)netwerken. Voor een router maakt het geen verschil uit of dit nu een volle klasse netwerk is dan wel een subnet ervan.



Figuur 299. Hoe vindt een router zijn weg.

De eerste 2 netwerken in bovenstaande figuur zijn 'connected' (directly), het netwerk *m* is te vinden via int. 3 achter router 2.

Voor een 'echte'router ziet dat er overigens hetzelfde uit:

Er zijn:

- '**C**' : Connected
- '**S**' : Statische  
(=manueel  
geconfigureerde)

entry's, en ook een

- **DGW:** 0.0.0.0/0,  
hier → **10.6.0.1**

```

WS-3750E-G#sh ip ro
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2
      i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
      ia - IS-IS inter area, * - candidate default, U - per-user static route
      o - ODR, P - periodic downloaded static route

Gateway of last resort is 10.6.0.1 to network 0.0.0.0

      143.129.0.0/24 is subnetted, 2 subnets
      C        143.129.40.0 is directly connected, Gigabitethernet0/1
      C        143.129.37.0 is directly connected, serial0/0
      C        10.6.0.0 is directly connected, Vlan940
      S        10.6.13.0 [1/0] via 10.6.0.11
      S        10.6.14.0 [1/0] via 10.6.0.11
      S        10.6.15.0 [1/0] via 10.6.0.11
      S*       0.0.0.0/0 [1/0] via 10.6.0.1
WS-3750E-G#
  
```

Figuur 300. Een praktische ruitertabel.

### 3.4.2 ROUTERINGSPROTOCOLLEN

We weten nu dat op elk netwerk dat met de buitenwereld wil communiceren (minstens) één router is aangesloten en dat alle stations zijn IP adres kennen als de 'default gateway'. Hoe weet deze router nu wat het 'beste' pad is om een datagram verder te sturen. Er zijn 2 soorten benaderingen :

- *Statisch* vs. **dynamisch**
- *Gecentraliseerd* vs. **gedistribueerd**

**Statisch** ➔ de routertabellen worden '**manueel**' ingegeven door de **systeemverantwoordelijke**, opgeslagen in een configuratiebestand en bij elke opstart van een router gelezen. Dit is enkel geschikt voor zeer beperkte internetten.

**Dynamisch** ➔ routers **leren** de netwerktopologie door te communiceren met andere routers : ze wisselen informatie uit over de netwerken waaraan ze zijn aangesloten.

Het heeft volgende voordelen (... en nadelen) :

- + Schaalbaarheid : grote netwerken configureren is statisch zeer complex
- + Adaptief : veranderingen in de netwerktopologie worden sneller opgenomen
- Het uitwisselen van routingstabellen bezorgt routers én netwerk een beetje overhead.

In tegenstelling tot een gecentraliseerd systeem, berekenen routers **zélf** het beste pad voor een datagram (➔ **gedistribueerd**), en wisselen deze info uit.

Maar wat betekent het 'beste' pad? ➔ Er wordt een '*link cost*' berekend op basis van een '*metric*'.

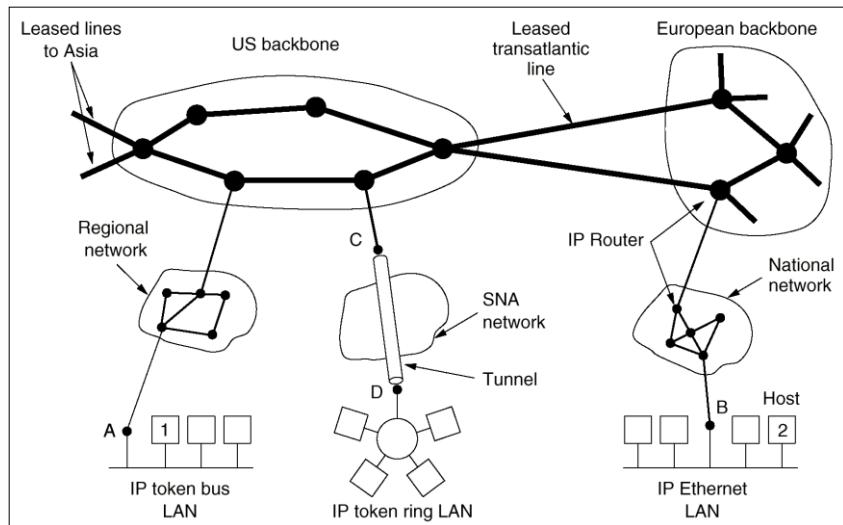
**Metrics** kunnen zijn :

- *Hop count* : elke te passeren router telt 1 maal mee
- *Bandbreedte* van het kanaal
- *Delay v/h pad in seonden*, afh. van de queue-lengte voor routers, congestie, ...
- Netwerk gebruik ➔ load balancing
- ...*Reliability* betrouwbaarheid

## HIERARCHIE

Er is geen echte structuur van het internet, maar bovenaan bestaat het uit een aantal belangrijke continentale backbones die gekoppeld zijn door intercontinentale lijnen.

Onder de continentale backbones bevinden zich **regionale** netwerken die het 'plaatselijk' verkeer proberen af te werken zonder de backbones, of erger nog de intercontinentale lijnen, te beladen.



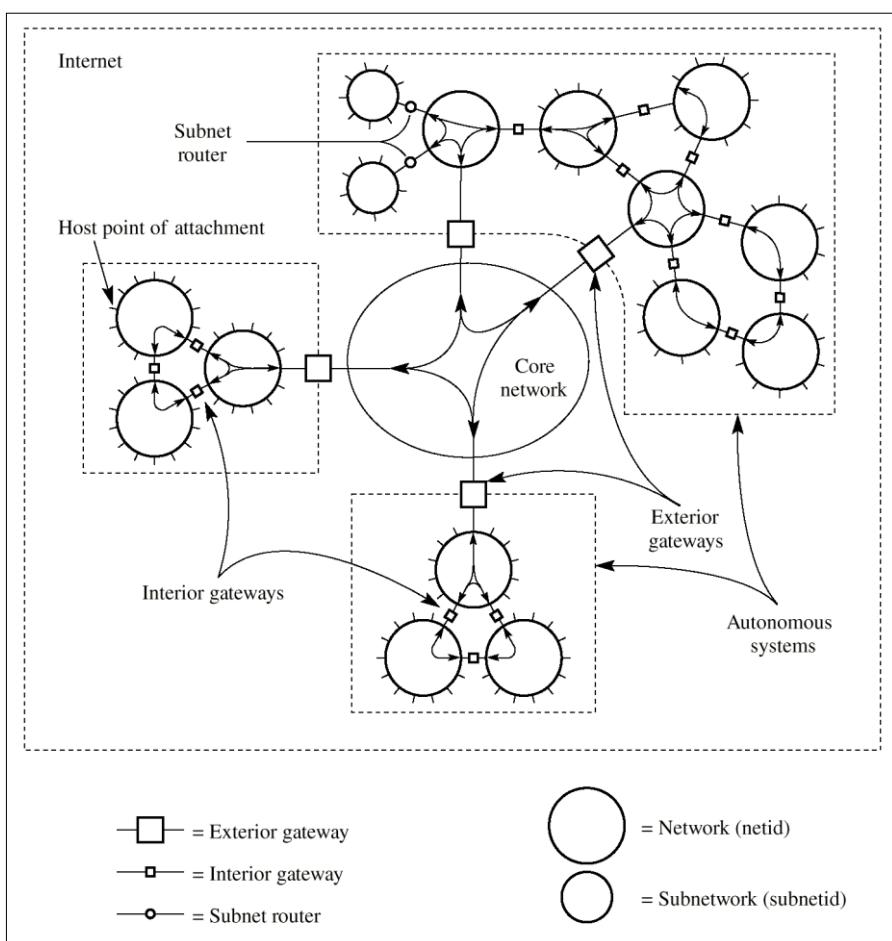
Figuur 301 De fysieke topologie van het Internet.

Het spreekt voor zich dat routing protocollen niet kunnen communiceren over het volledige internet heen. Hun werkingsdomein wordt afgebakend door zogenaamde '**autonomous systems**' border.

De fysieke topologie wordt logisch voorgesteld door een aantal **Autonome Systemen (AS)**, onderling verbonden door een '**core backbone**' netwerk.

Routers binnenin deze autonome netwerken worden '**interne gateways**' genoemd, met hun resp. protocol → **IGP**.

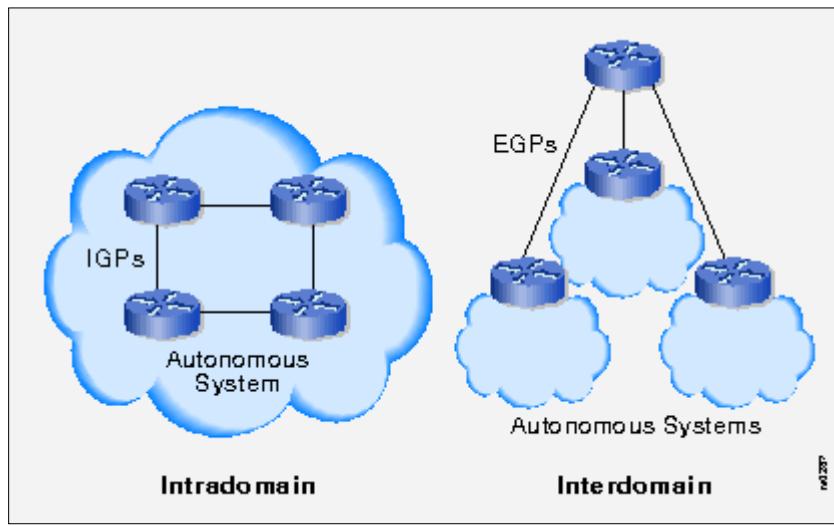
De router die het netwerk verbindt met de 'core' is de '**externe gateway**', met zijn protocol → **EGP**.



Figuur 302 Logische topologie van internetten.

Elk AS (Autonomoos Systeem) kan zijn **eigen IGP** kiezen om zijn routeringsinfo uit te wisselen, desnoods verschillend van zijn buur-AS, maar wel **uniform** voor **alle routers binnenin het AS**.

Het eerste populaire IGP was **RIP**, Routing Information Protocol. Recent zijn **RIP 2** en vooral **OSPF**, Open Shortest Path First, en **IS-IS**, intermediate system to intermediate system belangrijker geworden.



Externe Gateways hebben andere functies dan IG's, wat resulteert in andere protocols → EGP's.

Er is eigenlijk maar 1 algemeen gebruikt EGP: **BGP**, 'Border Gateway Protocol'. De border routers draaien intern een IGP maar extern BGP om hun AS aan de backbone aan te koppelen.

Figuur 303 Intradomain- en Interdomain Routing

Bovendien kan één autonoom netwerk, zoals gezien, dan nog bestaan uit een aantal subnetten die allen verbonden worden door 'subnet routers' (de hosts worden niet getekend).

Niet elke host of router/gateway bevat dus de **volledige** routerings-tabellen voor elk (sub)netwerk van het Internet... **niet realistisch**. Het wordt hiërarchisch georganiseerd.

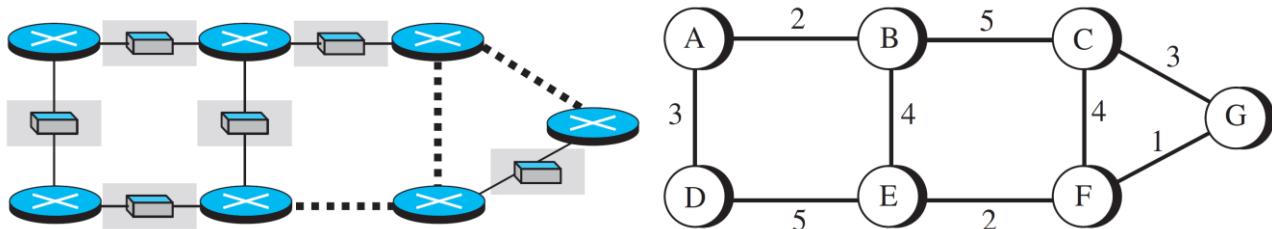
Men onderscheidt verschillende **niveaus** van routers/gateways op het wereldwijde internet.

1. **Hosts** bevatten genoeg routeringsinfo om datagrammen naar andere hosts te sturen op **hetzelfde** (subnet/) netwerk, of naar een interne (subnetrouter) gateway op hetzelfde (subnet) netwerk.
2. Interne gateways weten de weg om te routeren naar hosts of andere gateways **binnenin het netwerk of AS**.
3. Externe gateways 'forwarden' de datagrammen naar de **interne** gateways als het bestemd is voor **dit** netwerk, of naar andere **externe** gateways als dat **niet** zo is.

Hoe 'leren' de routers nu **dynamisch** hun '**least cost**' wegen of routes van hun **domein** waarin ze opereren.

**LEAST-COST ROUTERING**

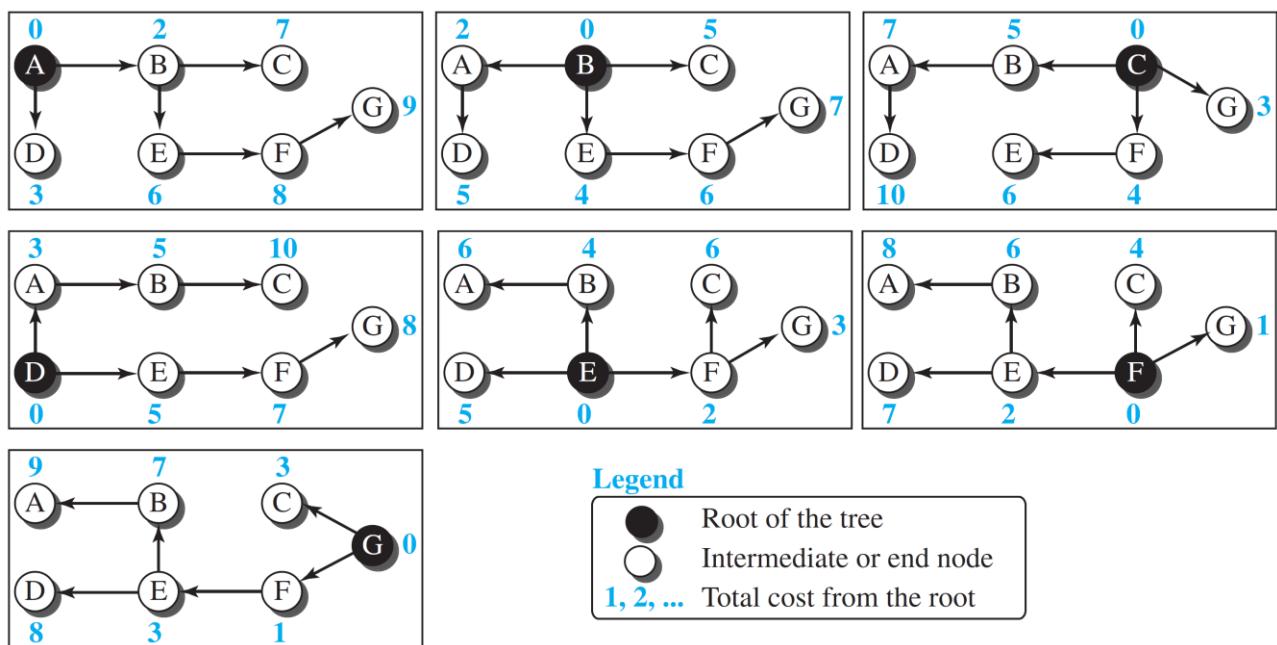
We hernemen fig Figuur 298. Graph voorstelling van een netwerk. Van p.268.



Wat is bv. De laagste kost om van router A naar router E te gaan?

⇒ Over router B met een cost van 6.

Zo zal elke router een **least cost tree** opstellen voor de bereikbaarheid van alle andere nodes (netwerken) binnenvin het AS.



Figuur 304. Least cost trees van routers

Hoe hij hierbij te werk gaat kan volgens een 3-tal principes:

- **Distance vector** routing, bv. RIP
- **Link state** routing, bv. OSPF of IS-IS
- **Path vector** routing, bv. BGP.

### 3.4.3 DISTANCE VECTOR ROUTING.

Bvb. RIP (Routing Information Protocol, RFC 1058 met update voor RIP-2, RFC 1723 in 1994). Dit **IGP** protocol werd in den beginne het meest gebruikte. Het baseert zich op info van de **dichtstbijzijnde** routers (de ‘buren’) waarbij het optimale pad wordt berekend uit ‘*sub optimal paths*’.

Voor de eenvoud gebruikt RIP als *metric* : **hops**<sup>1</sup>.

- Elke router zal **periodiek** (typ. 30s) zijn DV-tabel (=routing tabel) zenden naar zijn **onmiddellijke buren**. (en ook als zijn info wijzigt zowel als de link opkomt of afgebroken wordt).
- Hij **berekent** zelf zijn eigen **DV** door de som van de *metrics* die hij ontvangen heeft van zijn buren + de ‘*link cost*’ naar die buur, en dit vervolgens te **minimaliseren**. Hij zal meerdere paden naar zijn buren ontvangen, maar enkel het pad met de **laagste cost** wordt weerhouden. Dit is ook bekend als het ‘*Bellman-Ford*’-routeringsalgoritme.
- Elke *entry* heeft een **timeout** functie: wordt de route niet regelmatig bevestigd, dan verdwijnt ze uit de tabel. Bv. ‘*invalid route*’ na 180s. Na nog eens 240s zal de entry worden *geflushed*’.
- Routers kunnen dus verschillende routes vinden naar 1 bestemming, maar er wordt **slechts 1 route** weerhouden, zelfs bij gelijke DV (alternatieve equal cost routes worden **niet** weerhouden!). Enkel **betere** routes geven dus aanleiding tot het aanpassen van de tabel, en de tabel wordt opgesteld in de volgorde dat de info werd ontvangen.

OPM.

- RIP ondersteunde **geen subnet maskers** (voor CIDR/VLSM routering) → RIP 2. Dit protocol voorziet bijkomende informatie in het **UDP datagram**: de subnetmaskers bij elk IP dest.adres.
- Het opsturen van al deze routetabellen om de 30s leidt tot een beetje overhead op grote netwerken.

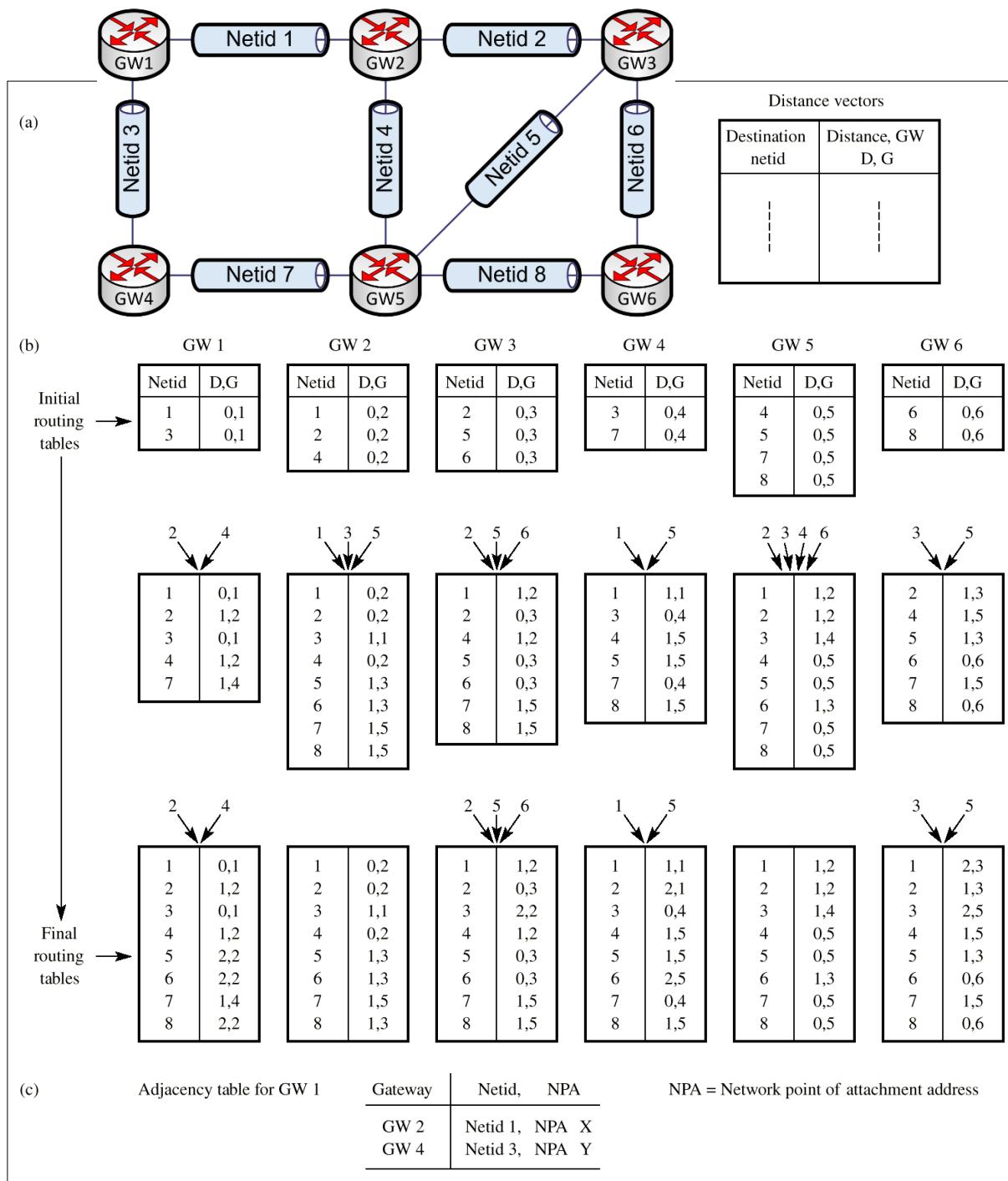
In volgende figuur proberen we ‘Bellman-Ford’ visueel voor te stellen.

---

<sup>1</sup> Het is duidelijk dat *hop counts* aanleiding kunnen geven tot minderwaardige paden (bv. 1 trage modemlijn i.p.v. 2 hypersnelle LAN’s).

Is de *metric* tijd, zoals bij het **HELLO** protocol, dan zullen de GW’s een datagram zenden naar de omliggende GW’s en wachten op een antwoord. De tijd die ondertussen verlopen is (/2) is dan de *link cost*.

Elke router of GW start met een Distance Vector (DV) = **0** naar **zichzelf** en  $\infty$  naar alle **andere** bestemmingen, zie figuur ‘*initial routing tables*’: de eerste lijn van de tabellen : D = DV.



Figuur 305 Distance Vector algoritme.

Op de tweede lijn tabellen hebben de onmiddellijke buren hun DV's uitgewisseld.

Voor **GW1** zijn dat **GW2** en **GW4**. Zo kan GW1 nieuwe, beste paden berekenen :

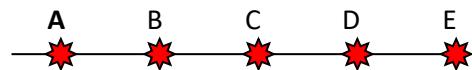
- net 2 te bereiken via GW2, met *distance* = 1 : de *link cost* naar GW2 = 1 hop.
- ... net 4 ... +1
- net 7 te bereiken via GW4, met *distance* = 1 : de *link cost* naar GW4 = 1 hop.

Wordt er een kleinere (betere) afstand naar een netwerk opgegeven als degene die in de tabel staat dan wordt de *entry* vervangen. Na 2 uitwisselingen hebben alle GW's de volledige netwerkinfo: ‘*good news*’ ... .

**Goed nieuws :** ... ‘travels fast’

Zoals hierboven besproken zien we dat ‘goed nieuws’ zich snel voortplant, slecht nieuws echter ... .

Beschouwen we een lineair net van 5 knopen en de **bereikbaarheid van A** (of de netwerk van A):

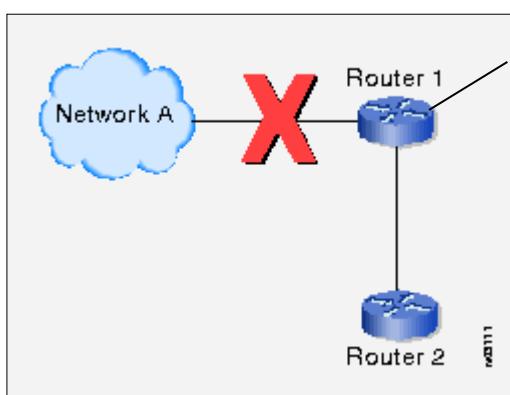


A	B	C	D	E	
0	$\infty$	$\infty$	$\infty$	$\infty$	Initieel
0	1	$\infty$	$\infty$	$\infty$	Na 1 uitwisseling
0	1	2	$\infty$	$\infty$	Na 2 uitwisselingen
0	1	2	3	$\infty$	Na 3 uitwisselingen
0	1	2	3	4	Na 4 uitwisselingen

Na 1 uitwisseling weet B dat A te bereiken is 1 hop naar links, ... na 4 uitwisselingen weet E dat A te bereiken is 4 hops verder.

**STABILITEITSFUNCTIES VAN RIP****Routing Loops**

Stel router 1 ziet zijn link naar netwerk A ‘down’ gaan.



Na 30s, bij het uitwisselen van de routeringsinfo, stelt Router R2 hem echter gerust : hij heeft een weg naar netwerk A met slechts **1** hop afstand. Er wordt niet gespecificeerd dat dit pad over R1 zelf gaat → geen argwaan → R1 past zijn tabellen aan : netwerk A is bereikbaar via R2, nu met **2** hops afstand.

Als router 1 nu een pakket binnenkrijgt met bestemming netwerk A ‘forward’ hij dit naar R2, die zoekt de bestemming op in zijn tabel en... ‘forward’ het naar R1 !

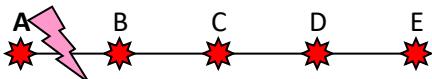
Figuur 306 Routing loop.

Het pakket zal tussen R1 en R2 kaatsen tot zijn TTL=0.

Deze toestand komt voor als een wijziging in het netwerk nog niet is ‘opgenomen’ in de routingtabellen → het protocol moet **convergeren**. Het probleem stelt zich ook in andere protocollen, maar vooral bij RIP vanwege de ‘distance vector’ filosofie. Het wordt opgelost door de ‘**count to infinity**’, wat echter een tijdje duurt.

**'count to infinity' : Slecht nieuws : ... ... 'travels slow'**

Veronderstel voorgaande stabiele toestand, maar A (of de lijn naar A) valt plotseling uit.



A	B	C	D	E	
0	1	2	3	4	Initieel
	3	2	3	4	Na 1 uitwisseling
	3	4	3	4	Na 2 uitwisselingen
	5	4	5	4	Na 3 uitwisselingen
	5	6	5	6	Na 4 uitwisselingen
	7	6	7	6	Na 5 uitwisselingen
	7	8	7	8	Na 6 uitwisselingen
	$\infty$	$\infty$	$\infty$	$\infty$	??

Bij de **eerste pakketwisseling** hoort B niets van A<sup>1</sup>, **maar** ... gelukkig zegt C : 'geen probleem, ik heb een pad naar A, met lengte 2'. B past dus zijn entry naar A aan  $\rightarrow = 3$ , via C. B weet niet dat dit bedoelde pad over zichzelf gaat, C kan een router zijn met 16 uitgangen waarlangs verschillende paden naar A lopen. (DIT is het probleem van distance vectoren : de routers krijgen **geen totaalbeeld** van het netwerk, ze stellen enkel hun minimaal pad op als som van andere minimale paden, i.t.t. *link state*, zie verder.)

D en E wijzigen hun tabel niet na de eerste uitwisseling.

De **tweede pakketwisseling** vertelt C dat elk van zijn buren een pad met lengte 3 heeft naar A  $\rightarrow$  C kiest willekeurig en zijn pad naar A wordt 4. De rest van de tabel is zelf op te stellen. Het is duidelijk dat slecht nieuws zich veel langzamer verspreidt, geleidelijk komen alle routers tot  $\infty$  voor A.

Het aantal uitwisselingen dat daarvoor nodig is is afh. v. de waarde die voor  $\infty$  wordt gebruikt in het protocol. Ideaal is dit de lengte van het langste pad +1. Praktisch gebruikt RIP de waarde 15 als max. hop count metric wat de grootte van het te gebruiken netwerk beperkt: autonome systemen met RIP kunnen maximum 15 tussenliggende routers hebben op hun langste pad. RIP is dus niet aangewezen voor grote netwerken. De waarde 16 wordt als  $\infty$  of 'infinity' beschouwd.

Een oplossing voor deze trage convergentie naar  $\infty$  is '**split horizon**', dat 'default' wordt toegepast bij bvb. cisco routers. Het stelt dat de horizon gesplitst wordt en routeringsinformatie **niet mag teruggezonden** worden. M.a.w. een router stuurt aan een buur enkel dat deel van zijn routeringsinfo door dat niet dezezelfde buur als next hop heeft.

<sup>1</sup> In feite duurt het 3 min. alvorens de route naar A wordt beschouwd als 'down' of 'infinity'.

Het nadeel hierbij is dat de routeringsinfo niet kan ge'broadcast' worden op het netwerk aangezien de boodschap verschillend is per bestemmings-router.

De 'slecht-nieuws-convergentie' bij 'split horizon' wordt dan :



A	B	C	D	E	
0	1	2	3	4	Initieel
	16	2	3	4	Na time out van 180 s.
	16	16	3	4	Na time out + 1 uitwisseling
	16	16	16	4	Na time out + 2 uitwisselingen
	16	16	16	16	Na time out + 3 uitwisselingen

Een ander nadeel aan deze oplossing is dat route entry's timers bevatten die moeten gerespecteerd worden. Door geen informatie te krijgen over een bepaalde route (split horizon) zal het de timer moeten afwachten om de entry te verwijderen. Een alternatief dat hiervoor beter werkt is 'Split Horizon with Poisoned Reverse', waarbij een 'infinity' waarde wordt teruggezondenvoor de onbereikbaar geworden route i.p.v. niks. De entry's in de buurouters kunnen nu onmiddellijk dit slecht nieuws incalculeren.

**Samenvatting** voor distance vector protocollen :

"tell your neighbours about the world"

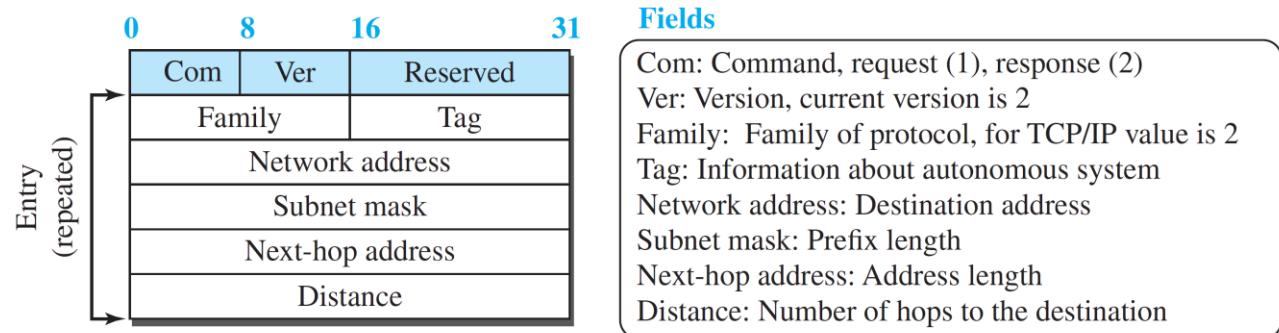
- + een **simpel** protocol, de routers hebben weinig rekenwerk.
- ± Niemand kent de volledige topologie van het netwerk → **gedistribueerd**
- + '**good news travels fast'**
- '**bad news travels slow'**

Vertel aan uw buren hoe uw wereld (routing tabel) eruit ziet.

Omwille van de trage convergentie van DV-protocollen, en de beperkte grootte van het netwerk stapt men over naar 'Link state' protocollen.

### INFORMATIEF: RIP PROTOCOL BERICHT.

Het bericht dat RIP versie 2 verzendt ziet eruit als volgt :



Figuur 307 RIP message

Het wordt verpakt in een IP pakket als **UDP** datagram → protocol veld = **17** met de gebruikte 'well known' UDP ports beiden: **520**.

RIP kan 25 routes (van 20 byte) bevatten in 1 bericht, wat de max. lengte brengt op  $20 \times 25 + 4 = 504$  bytes, en een UDP datagram van 512 byte, een IP pakket van 532 byte. Er zijn meestal verschillende dergelijke berichten nodig om de volledige tabel over te sturen.

De *address family* specificeert het netwerkprotocol → =2 voor IP. RIP kan verschillende protocollen ondersteunen.

De *metric* zit in het '*distance*' veld en is hop count. Het heeft een waarde van 1 tot 15 voor een geldige route, 16 → unreachable.

Als de *address family* van het eerste veld = 0xffff en de *tag* = 2 dan worden de resterende 16 byte gebruikt om een password door te geven in tekstformaat.

→ RIP-2 voorziet **authenticatie** : zowat elke (kwaadaardige) host kon zelf een RIP-1-message opstellen en daardoor de tabellen in de war sturen.

RIP-2 voorziet ook **multicasting** i.p.v. broadcasting om de niet-geïnteresseerde hosts te ontlasten.

Elke 30 seconden worden de routing-table-entry's opgefrist. Na 180s. zonder een normale update wordt de entry = 16 → 'infinity'. Nog 240s. later wordt de entry verwijderd.

### 3.4.4 LINK STATE ROUTING

Een moderner Interior Gateway Protocol (IGP) werd uitgewerkt als opvolger van RIP naargelang de AS (Autonomous Systems) groter werden → OSPF<sup>1</sup>, gebaseerd op ‘link state routing’:

- Elke router test de **link** tot zijn onmiddellijke **buren** en stuurt dit door in een **LSP**, Link State Package (soms **LSA's** : Link State **Advertisements** genoemd).
- Hij stuurt bovendien een LSP telkens er een **directe** link wijzigt : een nieuwe **buur**, de link cost naar een **buur** wijzigt, een link naar een **buur** valt uit, ...

Alle LSP pakketten die hijzelf ontvangt stuurt hij door op **al** zijn **interfaces**, behalve diegene waارlangs het is ontvangen → ‘**flooding**’.<sup>2</sup>

Bijgevolg ontvangt **elke** router **alle** LSP's van elke andere router (en plaatst die in een database) → hij krijgt een **totaal overzicht** van het **volledige netwerk**. Hij **berekent** na ontvangst van **alle** LSP's het ‘**shortest path**’ (i.e. het pad met een **minimale som** van de metrics) naar **elke andere** bestemming volgens het **Dijkstra** algoritme → dat wordt zijn router-tabel.

Er kan gewerkt worden met **verschillende metrics** :

- *Capacity* : een maat voor de ‘throughput’ in bps.
- *Delay* : incl. queue's voor bridges en routers.
- *Expense* : kost van een verbinding, bvb. leased lines.
- *Errorrate* : de BER.

De default is *capacity* en een **hogere** waarde voor de *metric* betekent een **lagere** capaciteit. OSPF kan per bestemming meerdere paden **per metric** definiëren en zo **QoS** (Quality of Service) – routing toepassen op basis van de TOS-bits in het IP pakket, (die vroeger door niemand werden gebruikt).

Hij kan zelfs per metric **meerdere ‘equal cost’-paden** bijhouden voor ‘load balancing’<sup>3</sup>.

Routers wisselen link informatie uit door lid te worden van een ‘dedicated multicast groupaddress’.

Net zoals bij RIP is dit routeringsprotocol beperkt tot de grenzen van zijn AS, autonomous system: geen enkele router wordt verondersteld de volledige topologie van het Internet te kennen. Maar de netwerken kunnen nu wel groter worden.

Er moet bovendien een vorm van security voorzien worden om hackers of ‘lollige studenten’ tegen te werken.

<sup>1</sup> OSPF = Open shortest Path First, RFC2328. Oorspronkelijk RFC 1131 en OSPF-2 RFC 1583.

Ook IS-IS (Intermediate System to Intermediate System) het OSI routeringsprotocol is een link state protocol.

<sup>2</sup> om duplicaten te vermijden wordt een ‘age’- en ‘volgnummer’ -veld voorzien in het LSP packet. Oudere LSP's worden niet meer geforward om de trafiek een einde toe te roepen.

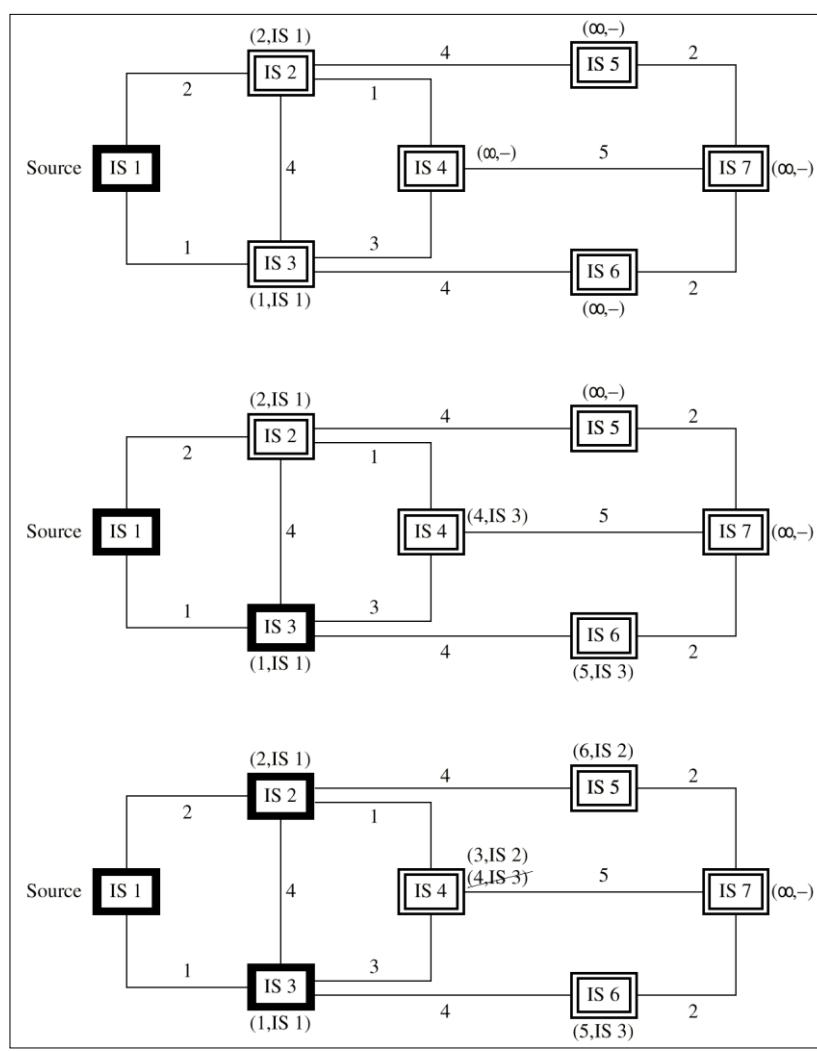
<sup>3</sup> Bvb. cisco's implementatie van OSPF kan 6 paden bijhouden naar één bestemming.

## WERKING

Het principe is best uit te leggen a.h.v. een figuur. Stel Intermediate System **IS1** is de 'source', t.t.z. we onderzoeken Dijkstra voor 1 bestemming, **IS1**, of de netwerken hieraan verbonden.

De *metrics* zijn aangegeven bij elk path als getallen naast of op de verbinding. We gaan ervan uit dat de cost dezelfde is per richting, t.t.z de cost van **IS1** → **IS2** = 2 = **IS2** → **IS1**.

**Elke IS** berekent zijn afstand tot **IS1**, en welke weg daarvoor moet gevuld worden. Vb. (2,IS1) bij **IS2** betekent afstand 2, te bereiken via **IS1**.



### Dijkstra stap voor stap (→ IS1) :

Na een LSP van **IS1** weten we de afstand en de te volgen weg van:

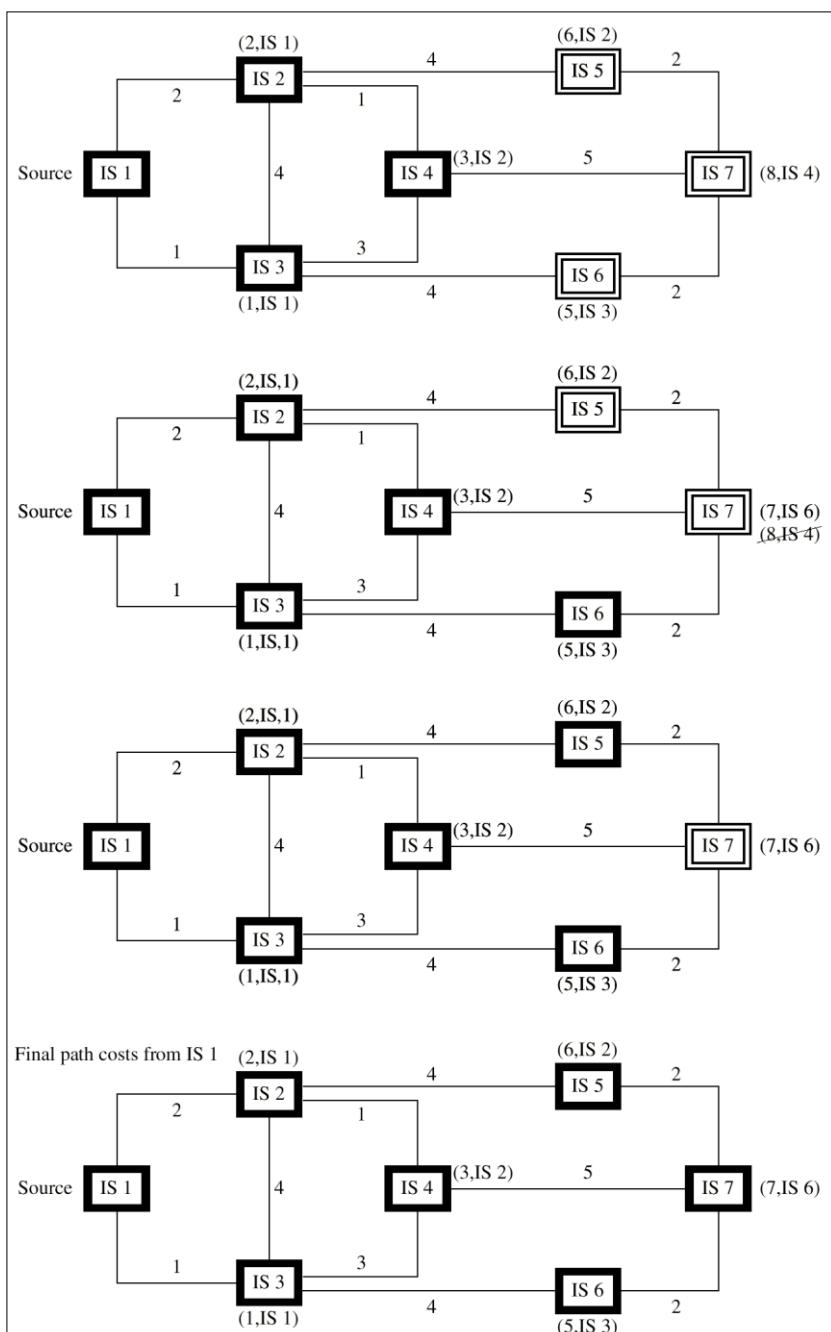
- **IS2** → **IS1** (2,IS1) en
- **IS3** → **IS1** : (1,IS1)

### Na een LSP van **IS3** :

- **IS4** : (3+1= 4,IS3) ...
- **IS6** : (4+1= 5,IS3)
- **IS2** : (4+1= 5,IS3)? > (2,IS1)  
→ niet vervangen

### Na een LSP van **IS2** :

- **IS4** : (1+2=3,IS2) ?< (4,IS3)  
→ vervangen !!!
- **IS5** : (4+2=6,IS2)
- **IS3** : (4+2=6,IS2)? > (1,IS1)



Figuur 308 Link state routing algoritme

Ook van de andere IS kunnen de kortste paden berekend worden en uiteindelijk de routing tabellen opgesteld, zie volgende blz.

**Samenvatting voor link state protocollen :**

*"tell the world about your neighbours"*

Na een LSP van **IS4** :

- IS7 :  $(5+3=8, \text{IS4})$
- IS2 :  $(1+3=4, \text{IS4}) > (2, \text{IS1})$
- IS3 :  $(3+3=6, \text{IS4}) > (1, \text{IS1})$

Na een LSP van **IS6** :

- IS7 :  $(2+5=7, \text{IS6}) < (8, \text{IS4}) !!!$
- IS3 :  $(4+5=9, \text{IS6}) > (1, \text{IS1})$

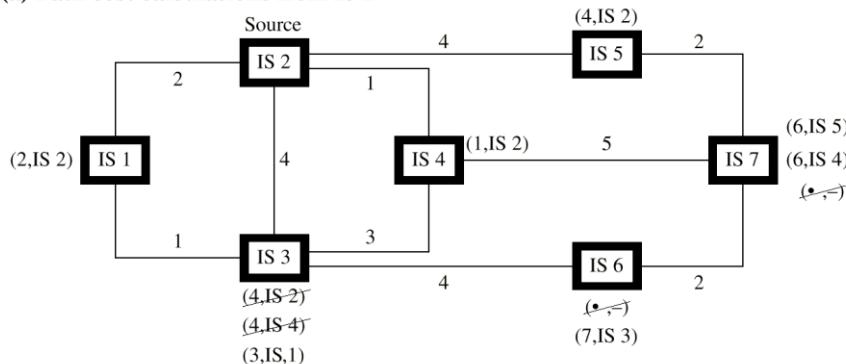
Na een LSP van **IS5** :

- IS7 :  $(2+6=8, \text{IS5}) > (7, \text{IS6})$

Het uiteindelijke shortest path van/naar IS1.

**Oefening.** Zoek zelf de oplossing voor de bereikbaarheid van IS2 ... oplossing hieronder.

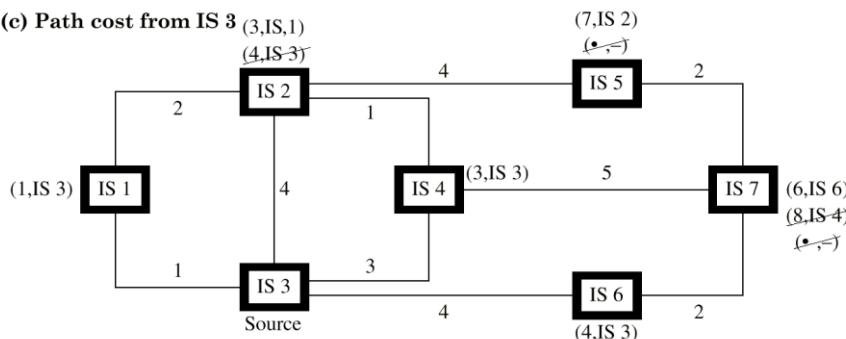
(a) Path cost calculations from IS 2



(b) Shortest path cost routes from IS 2

IS 2 → IS 1(Cost = 2)  
 IS 2 → Source (Cost = 0)  
 IS 2 → IS 1 → IS 3(Cost = 3)  
 IS 2 → IS 4(Cost = 1)  
 IS 2 → IS 5(Cost = 4)  
 IS 2 → IS 1 → IS 3 → IS 6(Cost = 7)  
 IS 2 → IS 4 → IS 7(Cost = 6)  
 IS 2 → IS 5 → IS 7(Cost = 6)

(c) Path cost from IS 3 (3,IS,1)



(d) Shortest path cost routes from IS 3

IS 3 → IS 1 (Cost = 1)  
 IS 3 → IS 1 → IS 2 (Cost = 3)  
 IS 3 → Source (Cost = 0)  
 IS 3 → IS 4 (Cost = 3)  
 IS 3 → IS 1 → IS 2 → IS 5 (Cost = 7)  
 IS 3 → IS 6 (Cost = 4)  
 IS 3 → IS 6 → IS 7 (Cost = 6)

(e) Routing table examples

IS 1:		IS 2:			IS 3:	
Destination	Path Cost	Dest	Path	Cost	Dest	Path Cost
IS 1	- -	IS 1	IS 1	2	IS 1	IS 1 1
IS 2	IS 2 2	IS 2	-	-	IS 2	IS 1 3
IS 3	IS 3 1	IS 3	IS 1	3	IS 3	- -
IS 4	IS 2 3	IS 4	IS 4	1	IS 4	IS 4 3
IS 5	IS 2 6	IS 5	IS 5	4	IS 5	IS 1 7
IS 6	IS 3 5	IS 6	IS 1	7	IS 6	IS 6 4
IS 7	IS 3 7	IS 7	IS 4/5	6	IS 7	IS 6 6

Figuur 309 Oefening op link state routing.

**Merk op :**

- De kortste pad routes zijn **omkeerbaar** omdat we ze in beide richtingen dezelfde cost hebben gegeven: IS2→IS1→IS3 en dus ook IS3→IS1→IS2.
- Als 2 IS dezelfde ‘path cost’ hebben naar een bestemming kan gekozen worden welke wordt weerhouden, of... er worden zelfs **meerdere routes** weerhouden  
➔ OSPF verdeelt de trafiek : ‘*load balancing*’.
- Als  $IS_x$  op het kortste pad ligt van  $IS_y$  naar  $IS_z$ , dan is dit ook het kortste pad van  $IS_x$  naar  $IS_z$ : bv. kortste pad van  $IS_1 \rightarrow IS_7 = IS_1 \rightarrow IS_3 \rightarrow IS_6 \rightarrow IS_7$ , dus ook  $IS_3 \rightarrow IS_7 = IS_3 \rightarrow IS_6 \rightarrow IS_7$ .  
Alle routers bezitten uiteindelijk dezelfde database met alle links van het netwerk (‘totaalbeeld’) en voeren hetzelfde dijkstra algoritme uit, dus komen dezelfde kortste paden uit.

**VERGELIJKING TUSSEN LINK STATE EN DISTANCE VECTOR**

- OSPF kan voor **verschillende metrics** router tabellen berekenen ➔ afh. v. de gevraagde QoS in de TOS bits van IP, zie pt3.2 op p. 225 Het IP datagram (low delay, high throughput,...) zal (kan) op basis van de gepaste tabel gerouteerd worden.
- Als gevolg hiervan is **OSPF veeleisender** voor de routers : het vraagt een grotere database en meer berekeningen.
- Beide protocollen hebben een vergelijkbaar bandbreedte-gebruik aangezien RIP **grote** routetabellen doorstuurt, maar enkel naar zijn **buren**, t.o.v. OSPF : **kleine** link state pakketten, maar naar **alle** routers.
- Het link state algoritme zal echter sneller convergeren (= stabiliseren) voor **grote netwerken** : RIP zijn grote probleem is de ‘count to infinity’ en zijn beperking tot 15 hops.
- OSPF kent de **volledige topology** van het netwerk, RIP enkel tot zijn buur.
- OSPF kan onderscheid maken in de richting (*direction*) van het pad, wat we in onze uiteenzetting niet hebben gedaan. RIP kan dit niet.
- OSPF gebruikt IP direct : het heeft zijn eigen protocolnummer (= 89) in de IP-header.  
RIP gebruikt UDP poort 520.
- OSPF gebruikt IP multicasting, RIP-1 broadcasting, RIP-2 ook multicasting.
- OSPF voorziet security via een passwoord protectie, RIP zendt ‘*plain text*’ messages.
- Beiden hebben uiteraard een begrensde database waardoor bvb. de kennis van het netwerk beperkt moet blijven ➔ ... tot 1 *autonomous system*.

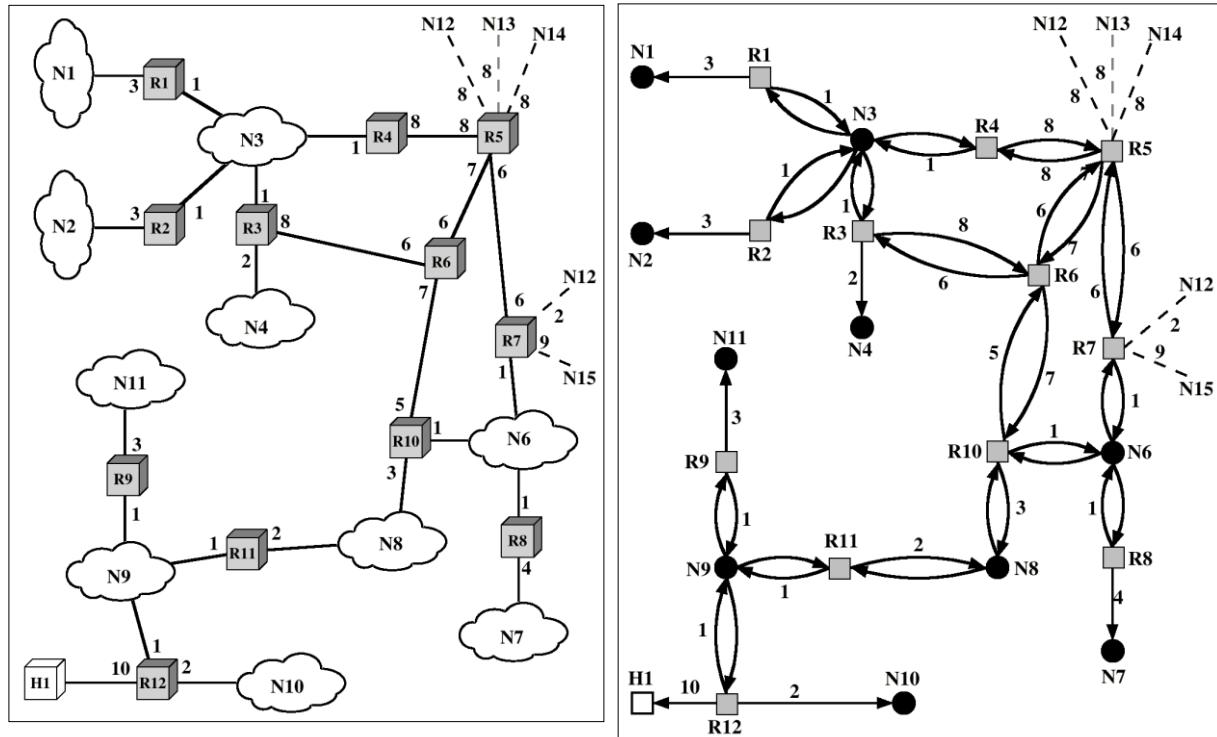
## INFORMATIEF: OSPF GRAPH VOORSTELLING

Een netwerk wordt voorgesteld door een graph waarbij aan elke verbinding een ‘cost’ wordt gegeven. Zowel routers als netwerken (en evt. hosts, H1) worden voorgesteld als een punt (**‘vertex’**).

De verbindingen worden voorgesteld door **pijlen** die gewichten dragen. Deze gewichten kunnen verschillen afh. v. de richting.

Per 'cost', bvb. *delay*, *throughput*, *reliability* (=TOS bits van IP), wordt een graph gemaakt.

Figuur 310 Een autonoom systeem en zijn graph voorstelling.



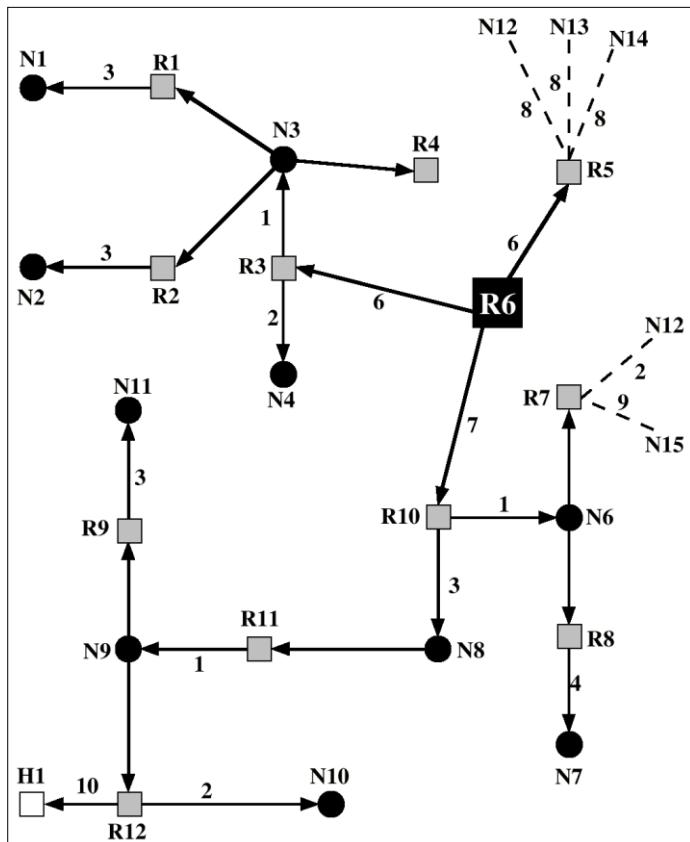
3 soorten verbindingen worden voorgesteld door pijlen:

- 2 routers verbonden met een **P2P** verbinding krijgen een pijl in elke richting, bvb. R4 en R5, mogelijk met een verschillend gewicht. Bvb. R6 en R10 of R3 en R6.
  - Als **meerdere** routers verbonden zijn aan een netwerk (bvb. X25 of LAN), dan worden ze **allen** bidirectioneel verbonden aan de netwerk vertex : bvb. R1,2,3 en 4 aan N3...
  - Als slechts één enkele router verbonden is aan een netwerk, dan is dit een 'stub connection' en is er slechts 1 pijl : N7, N10, ... (er kunnen geen routing loops ontstaan!)
  - Als een host rechtstreeks aan een router is aangesloten krijgt deze ook 1 pijl : H1
  - Routers die aan **andere autonome systemen** zijn aangesloten moeten hun link cost ver-krijgen van een EGP (Exterior Gateway Protocol) ➔ stippellijn naar bv. N12, N13 en N14.

Een link cost wordt geassocieerd met de **uitgang** van een router en is instelbaar door de system administrator. Omgekeerd, een '**ingang**' van een router die rechtstreeks verbonden is aan een netwerk heeft een cost = 0. Ze krijgen geen gewicht aan hun pijl.

Elke router houdt deze graph bij als een database, ze is samengesteld op basis van de ontvangen Link State Packets. Aan de hand hiervan berekent hij de **minimale cost** paden **naar** alle netwerken → de 'Shortest Path First (SPF) tree'.

Bvb. voor R6 wordt dit :

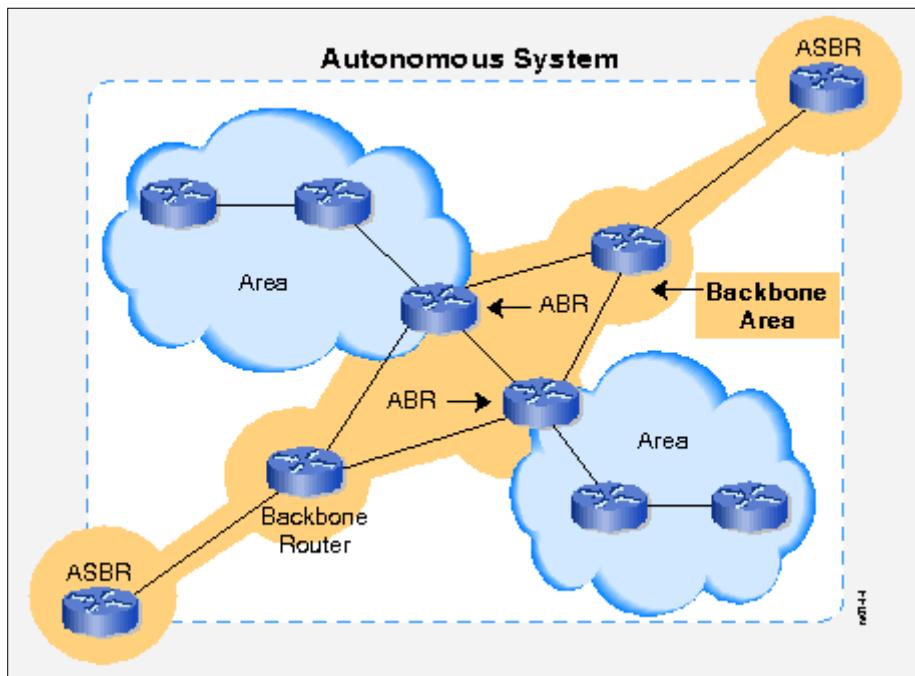


Figuur 311 De SPF tree voor router 6

De vraag is nu : hoe groot kan 1 *autonomous system* worden?

Als dit té groot wordt, waardoor de routerprotocollen teveel processortijd van de routers in beslag nemen (en bandbreedte van het netwerk), gaat men over tot een hiërarchische opsplitsing in 'area's.

### INFORMATIEF: HIERARCHIE IN OSPF NETWERKEN.



**Autonomous systems**, ook soms '**domains**' genoemd, worden opgesplitst in meerdere '**areas**'. Routers in zo'n area moeten enkel de topologie van hun eigen area kennen.

Er wordt een backbone voorzien die op zichzelf alle eigenschappen van een area heeft → '**backbone area**'.

Figuur 312 Hierarchie in OSPF netwerken.

Het doorzenden ('flooding') van de LSP's (of LSA's, Link State Packets/Advertisements) wordt nu beperkt tot 1 area → buiten de area is de topologie ervan niet gekend.

Ook de routers worden ingedeeld volgens deze hiërarchie :

- **Internal routers** : de routers binnenin **één area**. Ze kennen (→topologische database) enkel deze area en wisselen routeringsinfo uit met de andere internal routers in deze area. Ze hebben bijgevolg allen een **identieke** topologische database van deze **area**.
- **ABR** : *Area Border Router*. Ze nemen deel aan **meerdere OSPF areas**, en stellen bijgevolg voor elke gekoppelde area een aparte topologische database op. Ze routeren pakketten die 'area-overschrijdend' zijn.
- **ASBR**: *Autonomous System Border Router*. Dit zijn routers die verbonden zijn aan **meerdere AS**. Dit betekent dus netwerken met mogelijk een **ander IGP** (Interior Gateway Protocol), bvb. RIP. Hij leert over deze netwerken via een **EGP (Exterior Gateway Protocol)**, bvb. BGP zie p.291

Het pad naar elke ASBR is gekend door alle routers in het volledige AS.

Een speciaal geval is de backbone router : dit is een interne router van de backbone area. Ook de ABR's en ASBR's worden beschouwd als backbone routers.

Bij het opstellen van zijn route tabellen maakt OSPF onderscheid tussen 3 types van paden :

- *Intra-area route* : een pad waarvan de bestemming **in** dezelfde OSPF area ligt.
- *Summary route* : een pad met bestemming **buiten** de OSPF area → naar de **ABR**.
- *External routes* : een pad met bestemming **buiten** het OSPF AS → naar de **ASBR**.

Routes binnennin 1 area worden rechtstreeks gerouteerd door de interne routers.

Routes tussen 2 area's verlopen in 3 stappen : eerst naar de **ABR**, vervolgens over de backbone (area) naar de ABR van de bestemmings-area, en uiteindelijk naar de bestemming.

#### INFORMATIEF: ONDERHOUD VAN DE TOPOLOGISCHE DATABASE.

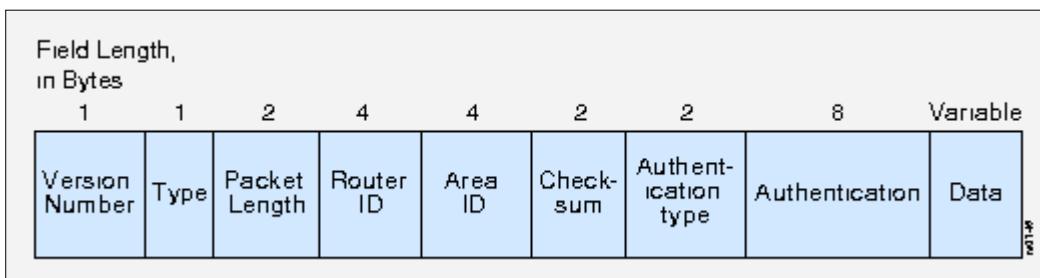
Wanneer een router op(/her)-start zendt hij een '**hello**' bericht over al zijn P2P lijnen en naar de IP-multicast groep op LAN's<sup>1</sup>. De buren antwoorden hierop, waardoor hij hun identiteit kent en bovendien, a.h.v. de verstreken tijd, de 'delay' cost kan berekenen (/2).

Bij grote netwerken is het nu niet efficiënt om link-informatie uit te wisselen met **al zijn buren** → enkel naar de **aangrenzende** of '**adjacent**' routers. → Er wordt een '**designated**' router gekozen die '**adjacent**' is aan **alle** routers van het netwerk. Er is bovendien een reserve of '**backup designated**' router voor het geval de eerste uitvalt.

Op gezette tijden, en wanneer de cost van een link wijzigt, stuurt een router nu een LSP, meer bepaald een '**link state update**', naar elk van zijn **adjacent** routers. (weliswaar met info over **AL** zijn buren!). Deze pakketten worden '**flooding**' doorgestuurd naar de andere **adjacent** routers, uiteraard binnennin één area. Elk bericht heeft een **volgnummer** zodat een ontvangende router kan zien of het bericht recentere is dan wat hij momenteel heeft. Voor de betrouwbaarheid worden deze berichten **bevestigd**.

Door een bericht van het type '**database description**' geeft een zender het volgnummer van al zijn link states, = alle elementen uit zijn database. Een ontvanger vergelijkt dan deze nummers met zijn database en kan bepalen wie de meest recente informatie heeft. Hij kan zonodig een '**link state request**' sturen om de nieuwste gegevens te verkrijgen.

Deze **types** LSP zijn terug te vinden in de OSPF Packet Header Format.



Figuur 313 OSPF Packet Header Format

**Version Number** : Identificeert de gebruikte OSPF versie.

**Type** :

Hello	Kennismaking met buren
Link-state update	Geeft de kosten weer van de router - interfaces
Link-state ack	Bevestigt een <i>Link-state update</i>

<sup>1</sup> Op WAN links heeft hij supplementaire configuratie-informatie nodig om te weten wie zijn buren zijn.

<i>Database description</i>	Maakt bekend welke update een zender heeft
<i>Link-state request</i>	Vraagt stukken van de database aan een buur

**Packet Length** -- geeft the packet lengte weer in bytes, inclusief de OSPF header.

**Router ID** : identificeert de zender van het packet.

**Area ID** : bepaalt de area waartoe het packet hoort.

**Checksum** : controleert het volledige packet.

**Authentication Type** : Alle OSPF pakketten worden beveiligd door een authenticatie type dat in te stellen is per area.

**Authentication** – Dit is de authenticatie informatie.

**Data** : de OSPF informatie.

OSPF voorziet ook de mogelijkheid tot ‘tunneling’, maar hiervoor verwijzen we naar een van de volgende hoofdstukken.

### 3.4.5 BGP, BORDER GATEWAY PROTOCOL

Dit routeringsprotocol is veel gebruikt in routeringen tussen ISP's. Om zijn specifieke toepassing geven we dit enkel mee ter informatie, het tijdsbestek laat een uitvoerige(re) besprekking niet toe.

BGP is een '**exterior gateway protocol**' voor communicatie tussen routers in verschillende autonome systemen die de TCP/IP suite gebruiken. Het vervangt het oudere EGP en versie 4 is gebruikt voor het Internet → RFC 1771 met ondersteuning van CIDR. Vorige versies zijn gedefinieerd in RFC 1267 (versie 3) en RFC 1268 (het gebruik ervan voor het internet).

Een BGP systeem wisselt de bereikbaarheid van netwerken uit, m.a.w. het volledige pad van AS (Autonome Systemen) dat moet afgelegd worden om het *dest. netid* te bereiken. Het is een **path vector** protocol, maar i.t.t. RIP laat BGP **policy based routing** toe : er worden paden gekozen op basis van beveiligings-, economische- of politieke grondslagen. (Bvb. niet langs Irak, ...)

Eerst gaan we IP datagrammen in een AS indelen in **local traffic** of **transit traffic**. Voor *local traffic* is **minstens 1** van de 2 adressen (source of dest.) een host **op** het AS. Elk ander IP datagram is *transit*. Een van de hoofddoelstellingen van een BGP is om *transit traffic* te minimaliseren.

Vervolgens worden de AS'en ingedeeld :

1. een **stub AS** heeft enkel **1** verbinding met een ander AS → alleen *local traffic*.
2. een **multihomed AS** heeft meerdere connecties naar andere AS maar **weigert transit traffic**.
3. een **transit AS** heeft meerdere connecties en is ontworpen om zowel *local- als transit traffic* te transporteren.

BGP kent 3 functionele toestanden/**procedures** en 4 soorten **messages**, over een TCP connectie:

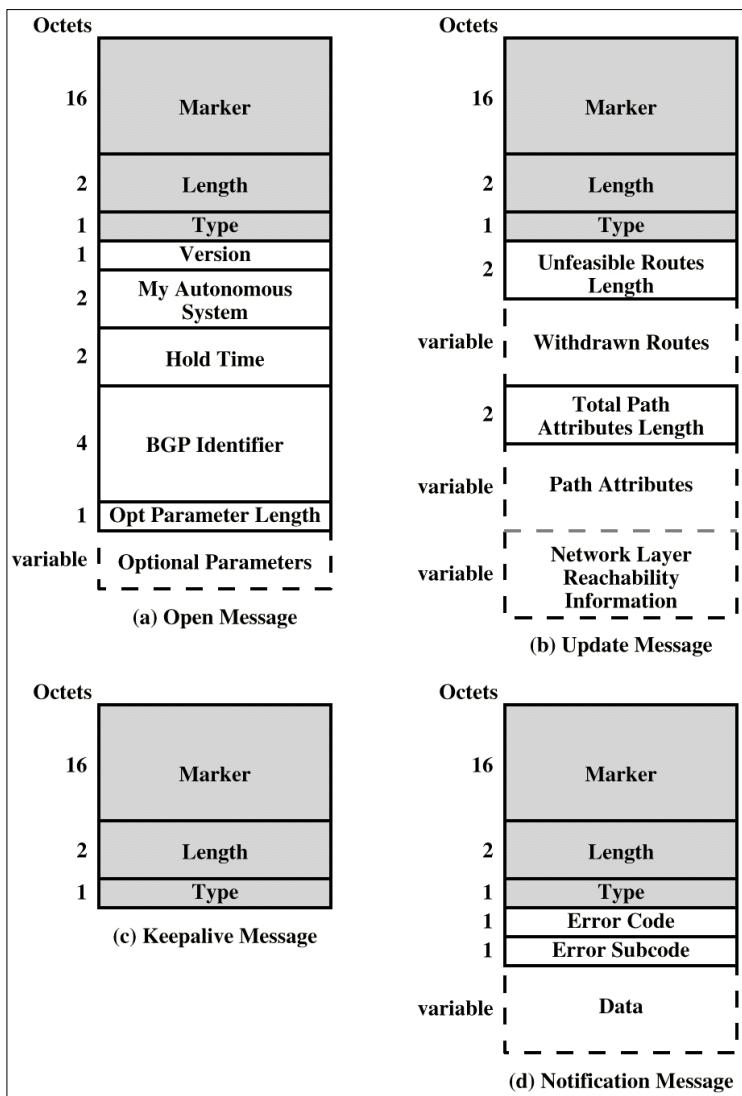
- |  |   |
|--|---|
| <ol style="list-style-type: none"> <li>1. Neighbor acquisition</li> <li>2. Neighbor reachability</li> <li>3. Network reachability</li> </ol> | <ol style="list-style-type: none"> <li>1. Open message</li> <li>2. Update message</li> <li>3. Keepalive message</li> <li>4. Notification message</li> </ol> |
|--|---|

Twee routers die in verbinding met elkaar staan worden beschouwd als 'neighbors'. Als ze routingsinfo wensen uit te wisselen moeten ze eerst overgaan tot de **neighbor acquisition** procedure. Hiervoor zal een van de routers een '**open message**' sturen naar zijn buur. Deze laatste kan zijn aanvraag weigeren als hij overbelast is of als hij niet geïnteresseerd is in *transit traffic*, ... . Aanvaardt hij de acquisition, dan stuurt hij een '**keepalive message**' terug.

Eens de **neighbor acquisition** relatie staat gaan ze over in een neighbor reachability procedure om hun relatie te onderhouden : elke partner moet er regelmatig van overtuigd worden dat de ander nog 'up' is, wat gebeurt door regelmatig '**keepalive messages**' naar elkaar te sturen.

De uiteindelijke procedure is network reachability. Elke router heeft een database van de netwerken die hij kan bereiken via een optimaal pad. (Distance vector!). Als er een wijziging plaats heeft in deze database stuurt hij een '**update message**' via een broadcast naar alle andere BGP-routers.

Elke BGP-message begint met een 19 byte header met 3 velden :



- **Marker** : gebruikt voor **authenticatie** : de zender schrijft hierin een waarde die de ontvanger kan ontsleutelen.

- **Length** : lengte van de message in bytes.
- **Type** : open, update, keepalive of notification.

Een **open** message, gebruikt voor neighbor **acquisition**, bevat :

- **Version**: de versie van BGP → 4
- **My AS**: AS nummer v/d zender.
- **Hold-Time**; het maximum aantal seconden tussen de keepalive- of update messages van de zender.
- **BGP Identifier** : een IP-adres, gedefinieerd bij het opstarten.
- **Optioneel** worden er supplementaire parameters meegegeven bvb. de authenticatie methode.

Figuur 314 BGP message formaten.

De **keepalive** message bevat enkel de header en dient om de hold timer te resetten (en in 1<sup>e</sup> instantie als ACK voor de open message).

De **update** message deelt enerzijds mee welke paden uit de database moeten **verwijderd** worden en anderzijds geeft het de totale informatie van **1 enkel** pad door het Internet dat moet **toegevoegd** worden aan de database. Deze informatie, de **AS\_Path attributes**, geven een volledige lijst van AS'en die moeten gevuld worden om deze route te volgen. Op basis hiervan zal BGP 'policy routing' toepassen. Voor meer informatie verwijzen we naar de cisco CD en Stallings p. 576.

Als laatste message is er de **notification** message die wordt gezonden als er **fouten** worden ontdekt. Zo zijn er message **header** errors voor syntax- of authentificatie-fouten, **open** message errors, **update** message errors, **hold timer expired**, ... .

BGP gebruikt **TCP** als transportprotocol [i.t.t. RIP (UDP) en OSPF (IP)]. Eerst moet een connectie opgezet, en vervolgens de volledige routing tabel uitgewisseld worden.

## 3.5 IP VERPAKKING

IP pakketten dienen verpakt te worden door een datalink, en dat brengt een aantal consequenties met zich mee.

### 3.5.1 ADDRESS RESOLUTION PROTOCOL ARP EN RARP

Wanneer hosts op een LAN een IP pakket moeten inpakken om bvb. naar een router te sturen moeten de hosts het **laag-2** (=MAC) -adres van deze routers kennen. (het IP adres is opgegeven door de gebruiker/applicatie). Adres resolutie voorziet in een ‘mapping’ tussen de 32 bit IP adressen en de 48 bit-MAC adressen van bv. ethernet of token ring netwerken(m.a.w. broadcast-netwerken met verschillende hosts, niet voor een P2P verbinding). Het kan worden beschouwd als een routerprotocol op het laagste niveau.

Een op te sturen datagram met het IP bestemmingsadres wordt door de IP laag doorgegeven aan de datalink, bv. ethernet. Deze moet het inpakken in een ethernet-frame met het resp. MAC-adres van de **bestemming** als deze zich op hetzelfde (sub)netwerk bevindt, of het MAC-adres van de (default) router anderzijds, ook op hetzelfde netwerk. (De keuze tussen deze 3 gebeurt door het IP-routeringsmechanisme : een vergelijking met het eigen IP-adres en –subnetmasker.)

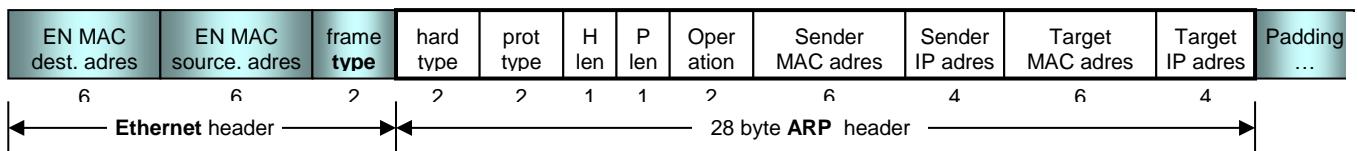
De werkwijze om deze MAC-adressen te bekomen is ARP, gedefinieerd in RFC 826.

In alle gevallen moet de datalink het frame verzenden naar een interface op hetzelfde (sub-)netwerk. ARP zendt een (laag-2) **broadcast** uit met de vraag :

- gegeven is het IP-adres van de bestemmingscomputer/router, geef mij uw MAC-adres ?
- Aangezien een broadcast door **elke** aangesloten interface wordt gelezen moet de bestemmingscomputer reageren op deze *ARP request*.
- Hij reageert met een *ARP reply*. (nu niet meer als broadcast, maar **direct** geadresseerd aangezien de ARP request zender zijn MAC-adres heeft meegegeven.) Na aankomst van de *reply* kan het datagram worden verzonden.

Om deze broadcasts te minimaliseren houdt elke host/router een **ARP cache** bij. Elke mapping die de laatste 20 minuten is gebruikt wordt bijgehouden.

Een ARP / RARP request of reply op ethernet ziet er uit als volgt :



Figuur 315 Ethernet verpakking van een ARP pakket.

- Voor ARP **request** zal het **dest.** ethernet adres  $\forall 1^L$  zijn, een **broadcast**. Het source adres is uiteraard de zender zijn MAC adres.
  - Het frame **type** ( $\neq$  length, zie ook Verschil tussen ethernet en IEEE 802.3 frames op p. 201) is voor ARP = **0x0806**. (RARP = 0x8035).

In het ARP pakket is :

- het eerste veld het **hardware type** (= 1 voor ethernet)
  - **Protocol type** is 0x0800 voor IP, niet toevallig dezelfde waarde als het ethernet type-veld voor een IP datagram.
  - De volgende 2 velden zijn de **lengte** van het **Hardware** (of MAC-) adres in bytes → 6 en het **Protocol** (of IP-) adres → 4 bytes.
  - Het **operation** veld specificeert het type ARP :
    - = 1 : ARP request
    - = 2 : ARP response
    - = 3 : RARP request
    - = 4 : RARP response
  - De volgende 4 velden verzorgen uiteindelijk de mapping.

**Figuur 316 ARP en RARP pakket.**

Merk op dat er duplicatie van informatie optreedt : de zender zijn MAC-adres komt 2 keer in het totale ethernetframe voor : 1 keer in de ethernet header, 1 keer in het ARP pakket.

Voor een ARP **request** worden alle velden ingevuld, behalve het **target** MAC adres.

**ARP reply** : wie het (IP) schoentje past ... zendt de reply terug mét zijn MAC adres ingevuld in de ethernet header **source**-adres, vervangt het broadcast destination adres in de ethernet header met het opgestuurde source MAC adres, wisselt in het ARP pakket de 2 sender en target-adressen en zet het *op*-veld =2.

De andere machines op het netwerk kunnen uit de 'request' de 'mapping' van de source computer (IP/MAC) ook opslaan in hun ARP cache aangezien dit een broadcast is.

### RARP.

Een opstartend systeem leest normaal zijn IP adres van een configuratie-bestand van schijf. Voor X-terminals of schijfloze systemen is dat een probleem. Elk systeem aangesloten aan een netwerk heeft op de netwerkkaart (in ROM) weliswaar een uniek MAC-adres, ingebakken door de fabrikant. De werking van RARP is nu dat het dit MAC adres leest en vraagt : gegeven is dit MAC-adres, wil iemand mij een IP adres geven → ‘**Reverse ARP**’. (*EN\_type = 0x8035*). (RFC 903).

Net zoals bij ARP is de **request** een broadcast, de **reply** een unicast.

Problemen :

- Er wordt enkel een IP adres opgestuurd, geen subnetmasker, default gateway, ...
- RARP is een laag 2 broadcast → geblokkeerd door een router → er is een RARP server nodig pér broadcast domein.

**BOOTP** (en de nieuwste versie daarvan : **DHCP**) zorgen wél voor een vollediger opstart-configuratie : de bootp server levert een *IP-adres*, maar daarnaast ook de ‘*default gateway*’, ‘*subnetmask*’, ‘*DNS server*’, en bovendien is het routeerbaar, dus niet beperkt tot 1 broadcast domein. Zie ook pt. 3.5.2 op p.297.

Zie ook RFC’s 951, 1048 en 1084.

### Gratuitous ARP.

Bij het opstarten kan elke machine zijn ‘mapping’ broadcasten aan het netwerk door een ARP-request. Hij vraagt hierbij het MAC adres van zijn **eigen** IP adres. Er mag nu geen antwoord komen, maar de andere hosts op het netwerk nemen de ‘mapping’ van de ‘source’ op in hun ARP cache.

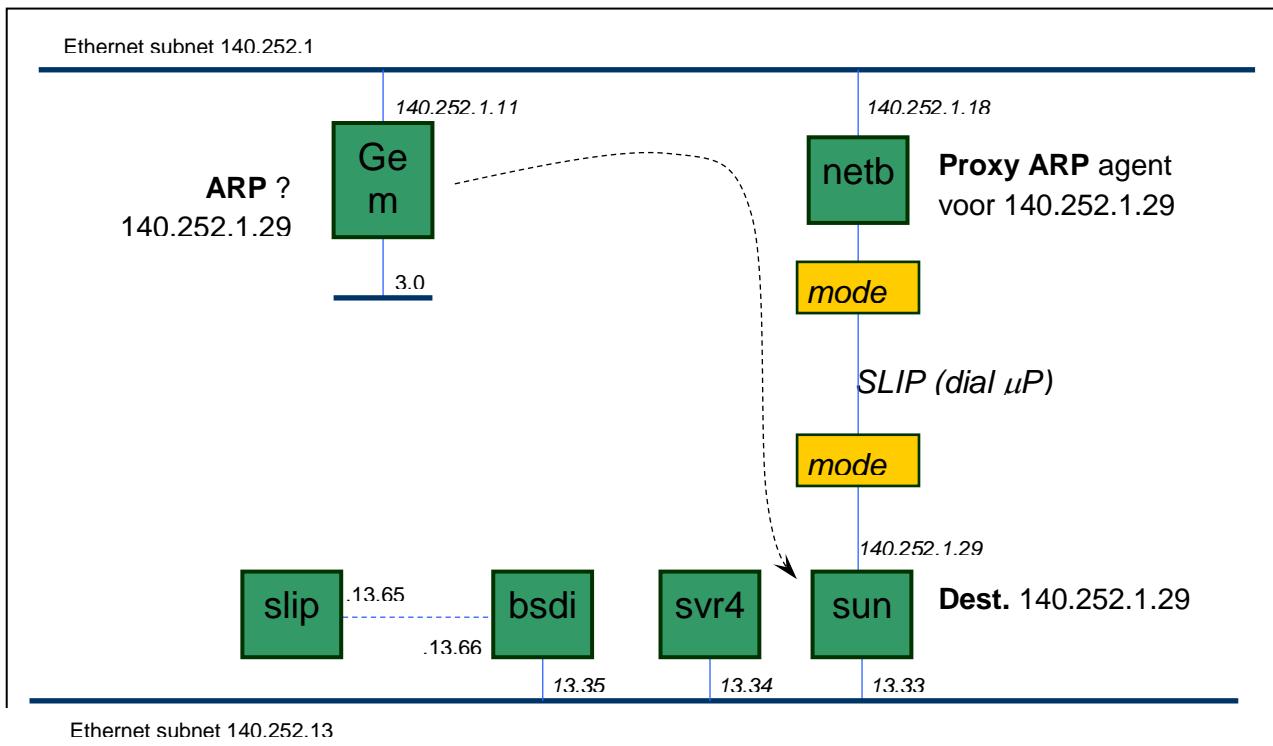
Komt er wel een antwoord, dan betekent dit dat een andere host reeds dit IP adres is toegekend. De nieuwe host moet de systeem manager op de hoogte brengen van de dubbele toekenning en dit uiteraard niet gebruiken.

### Proxy ARP.

We weten dat broadcasts beperkt zijn tot het lokale (sub)netwerk. Proxy ARP laat een **router** antwoorden op een ARP request op één van zijn aangesloten netwerken, bestemd voor een host op een *ander* netwerk. Voor de *sender* lijkt het dan dat de bestemming de router is, terwijl die gewoon als '*proxy agent*' fungeert voor een host 'aan de andere kant'.

De werking is het eenvoudigst te verklaren a.h.v. Figuur 317. Stel dat gemini een IP datagram heeft voor 140.252.1.29. Hijzelf zit op hetzelfde netwerk 140.252 en hetzelfde **subnet 1**. Hij stuurt nu een ARP broadcast uit om het MAC adres te kennen van 140.252.1.29. De router netb ontvangt dit en is geconfigureerd als proxy agent voor dit adres. Hij antwoordt met een ARP reply naar gemini : ik ben het, dit is mijn MAC adres (0x 00:80:ad:03:6a:80).

Het IP datagram vertrekt naar netb (via zijn MAC adres), daar aangekomen wordt het gerouteerd naar de SLIP lijn waارlangs het de SUN bereikt.

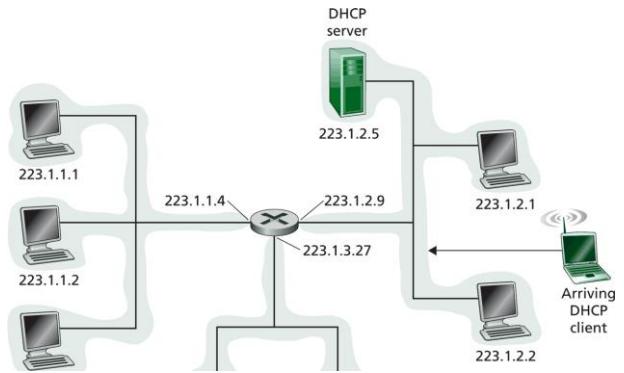


Figuur 317 Een voorbeeld van een proxy ARP

### 3.5.2 DHCP - DYNAMIC HOST CONFIGURATION PROTOCOL

In de figuur wordt een DHCP cliënt voorgesteld als een mobiele node, maar dit kan eveneens een (opstartend) vast werkstation zijn.

In alle geval : de node wenst een geldig **IP-adres**, **subnetmask** en **def. Gateway** te bekomen om te functioneren op het (sub)net 223.1.2.0. (evt. ook nog het adres van een DNS server, zie later)



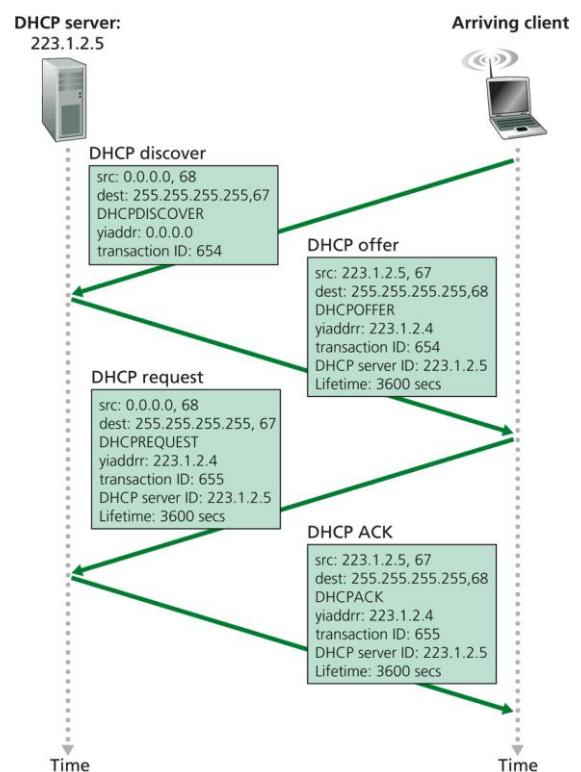
Figuur 318. DHCP toekennen van IP adressen.

Op ditzelfde netwerk is een DHCP server<sup>1</sup> aanwezig. Als eerste stap zal de arriverende cliënt deze DHCP server opsporen → ‘DHCP discover’.

Hij heeft echter zelf nog geen IP adres, en kent het adres van de DHCP server ook niet. Daarom verzendt hij het als ‘**lokale broadcast**’ dest. IP = 255.255.255.255 → iedereen op het subnet! (broadcast worden niet ge’forward’ door een router).

Het source adres = ‘this host’ = 0.0.0.0. Het bericht bevat ook nog een transaction-ID om de berichten die bij deze uitwisseling horen te bundelen.

Iedereen op het subnet ontvangt dit bericht, maar enkel de DHCP server zal een antwoord sturen → **DHCP offer**<sup>2</sup>. Hierin vermeldt hij reeds zijn eigen IP (source-) adres : 223.1.2.5 én het voorgestelde te gebruiken IP adres, ‘yiaddr’ ('YourInternetADDRess), 223.1.2.4



Figuur 319. DHCP uitwisseling.

Het dest. IP is nog steeds iedereen. Iedereen luistert, maar slechts 1 host heeft trans-id 654 gekozen. Hij krijgt dus het voorstel om ‘yiaddr’ 223.1.2.4 te gebruiken, voor 1 uur.

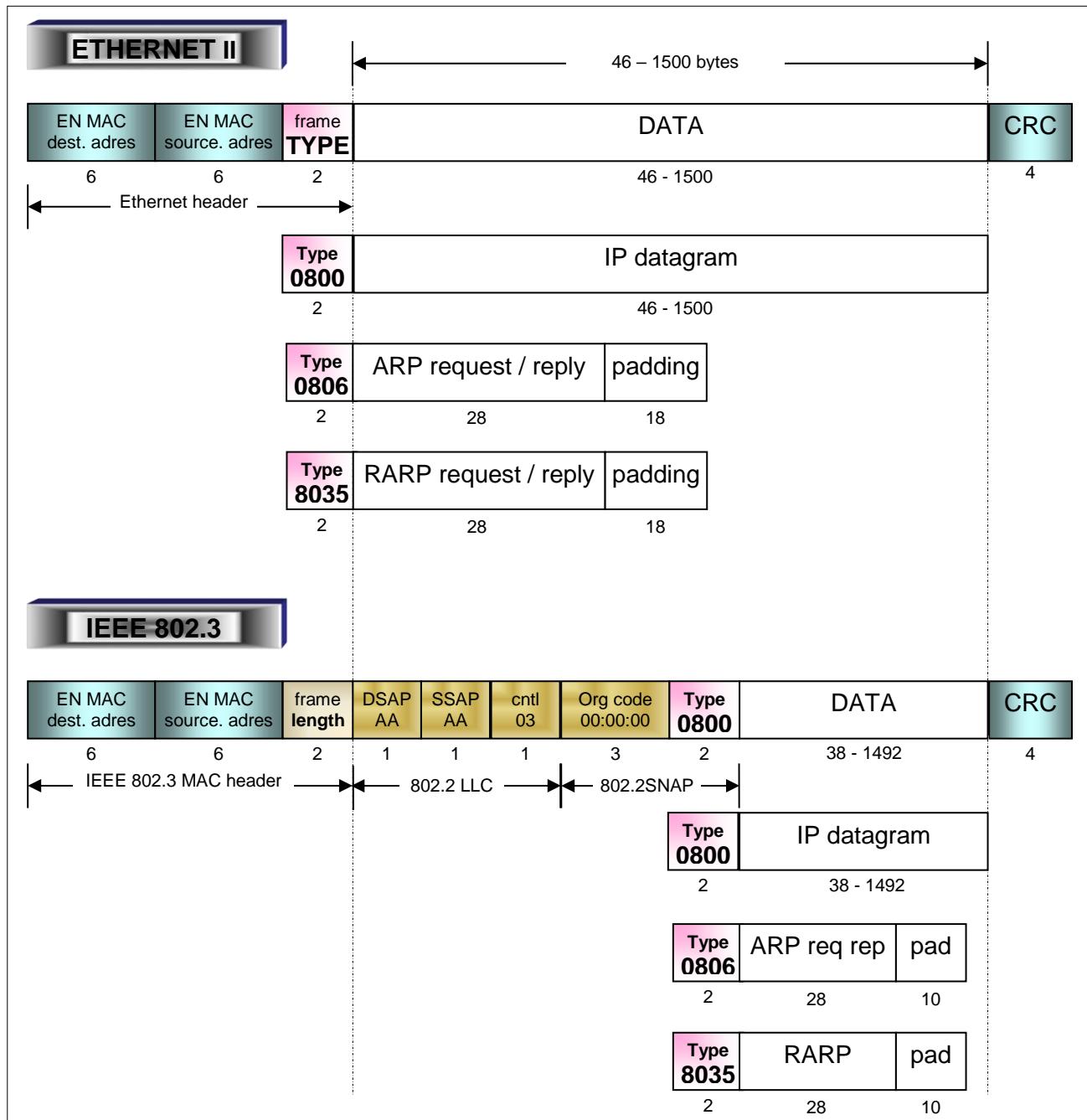
De DHCP cliënt gaat akkoord met het voorstel en antwoordt terug met een ‘**DHCP-request**’ bericht. Van zodra de DHCP server dit terug bevestigt, ‘**DHCP-ACK**’ kan het definitief in gebruik genomen worden, en gemarkerd als ‘in use’ bij de server.

<sup>1</sup> Bv. een PC met **WIN2003srv** waarop de DHCP services ‘enabled’ zijn. Dezelfde functie kan ook overgenomen worden door de **router** zelf! Er wordt een ‘**pool**’ van adressen aangeduid waaruit de DHCP server kan kiezen bij het uitreiken van de individuele IP adressen. Uiteraard kiest hij geen 2 maal hetzelfde IP adres uit de pool, zolang dit nog ‘leeft’. Op de DHCP server wordt ook het **subnetmask** én de **def. Gateway** geconfigureerd die moet worden uitgedeeld.

<sup>2</sup> DHCP berichten maken gebruik van UDP poorten 67 (srv) en 68(clt). Enkel de DHCP server heeft een ‘antwoordservice’ draaien op deze poort. Zie later bij TCP/UDP en poorten.

### 3.5.3 ETHERNET VERPAKKING VAN IP PAKKETTEN.

Om te beginnen herhalen we het verschil tussen ETHERNET II en IEEE 802.3 – 802.2 frames met SNAP header, de 2 meest voorkomende datalink verpakkingen.



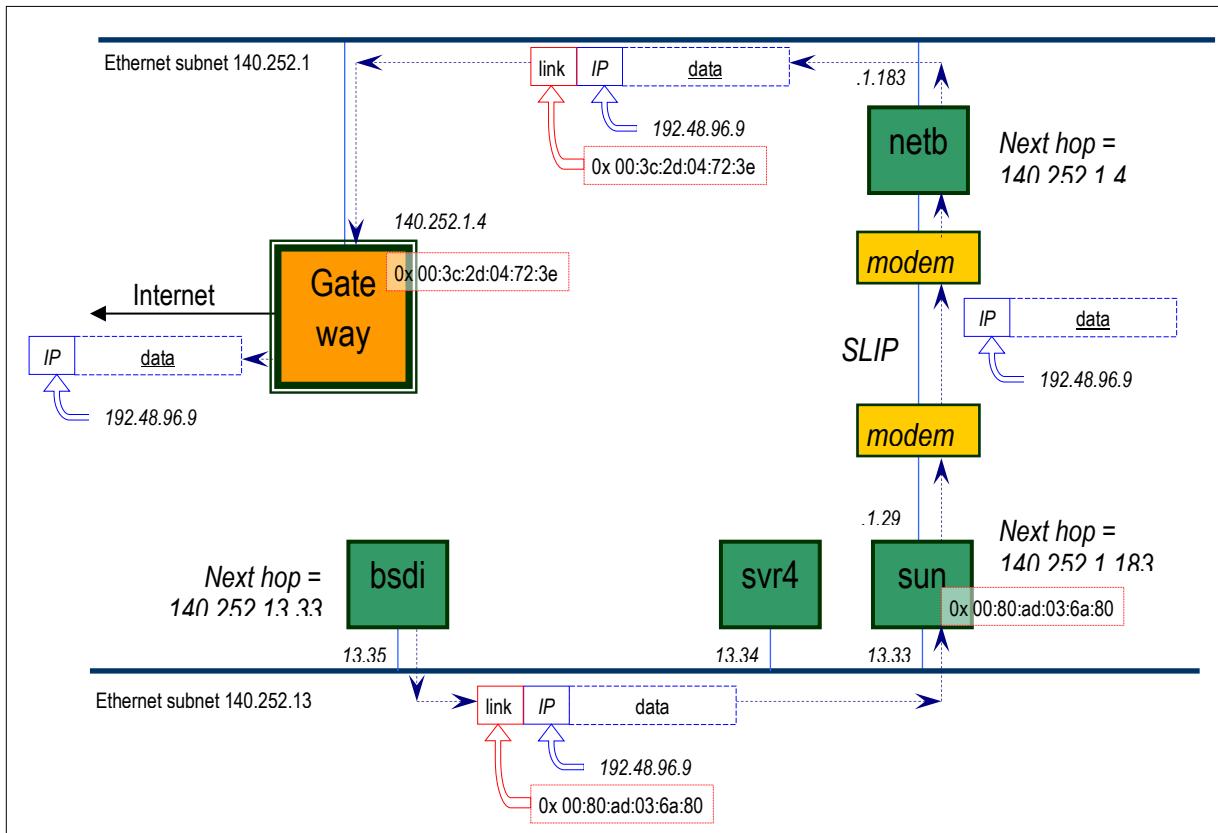
Figuur 320 Datalink frame formaat : ethernet II en IEEE 802.3 - 802.2

Een IP datagram op een ethernet LAN kan dus max. 1500 of 1492 bytes lang zijn.

### IP ADRESSEN ↔ ETHERNET-ADRESSEN

Stel dat *BSDI* een datagram wil verzenden naar een host [ftp.uu.net](ftp://ftp.uu.net), wiens IP adres is opgezocht via DNS = 192.48.96.9 . *BSDI* loopt zijn routertabellen af, maar vindt geen ‘match’. (noch host, noch netwerk). Hij stuurt het naar *SUN* (default) waarvan hij het MAC adres heeft in zijn ARP cache of opvraagt via ARP. Het ethernet frame op subnet 13 heeft als bestemmings-MAC-adres dat van *SUN* maar nog altijd als bestemmings - IP adres de uiteindelijke geadresseerde : 192.48.96.9 .

Figuur 321 Vergelijking tussen ethernet en IP adressen



Bij ontvangst door *SUN* merkt deze dat het niet gaat over een van zijn eigen IP adressen. Hij is geconfigureerd als router en zoekt bijgevolg in zijn router tabellen. Daar vindt hij geen match en stuurt het door naar de *default router NETB* over een SLIP link die geen datalink adres heeft aangezien dit een P2P verbinding is.

*NETB* ontvangt op zijn beurt, en merkt eveneens dat het niet gaat over een van zijn eigen IP adressen. In zijn routing tabellen is de internet-gateway op subnet 1 de default router. Hij verpakt het datagram met het MAC-adres (ARP) van deze gateway en stuurt het op... . (zie ook cisco-CD : ‘path switching’)

#### MERK OP dat :

- Het IP adres in al deze pakketten niet wijzigt.
- Per link worden andere link-layer headers gebruikt met bestemmingsadres de ‘next-hop’.

### 3.5.4 FRAGMENTATIE

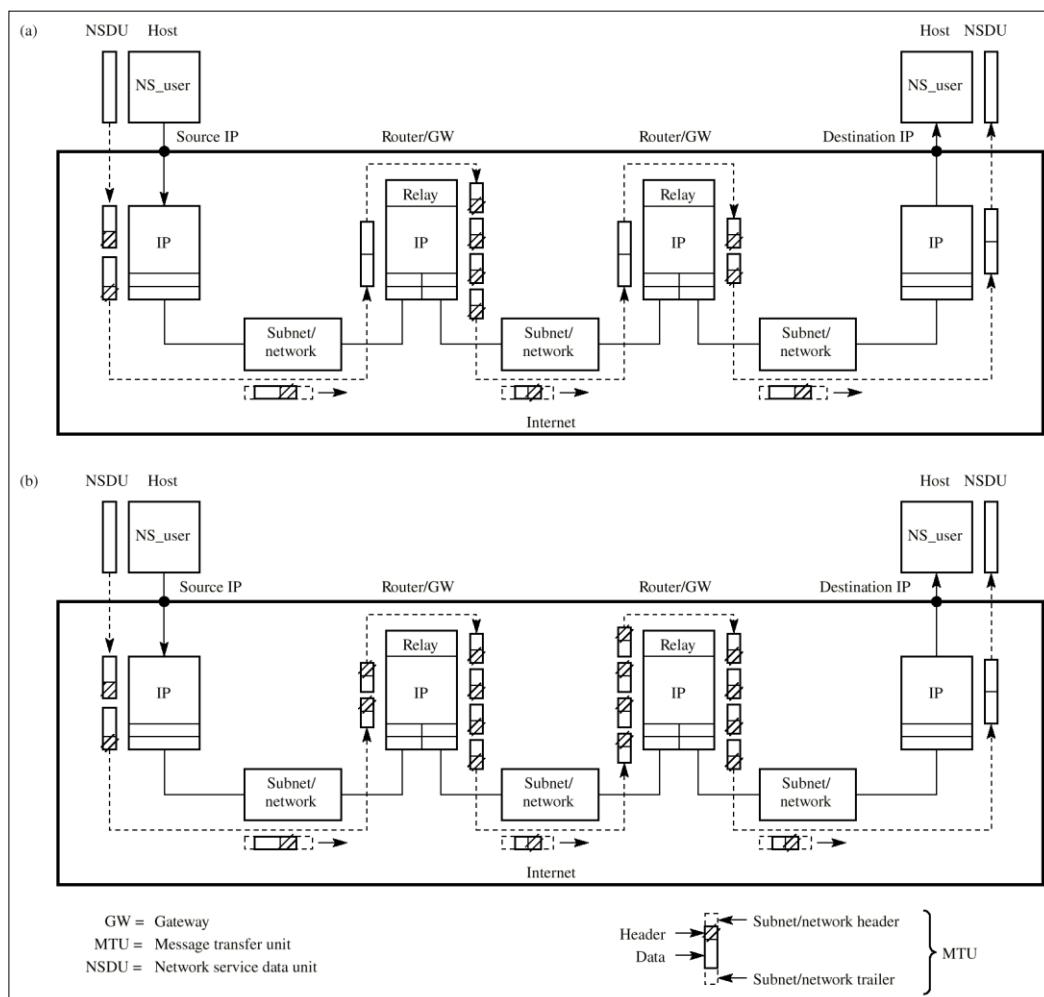
In pt. 3.4 IP routing. op p.268 hebben we een opsomming gemaakt van de **functies** die een IP laag moet uitvoeren. Een van de belangrijkere is ook **fragmentatie**. In theorie kan een IP datagram 65535 bytes lang zijn, maar praktisch hangt dit vooral af van de MTU (Maximum Transfer Unit) van het onderliggende netwerk : van 48 bytes voor ATM tot 8000 bytes voor sommige LAN's (lagere BER → grotere pakketten).

(De IP laag vraagt bij het starten de MTU op van de aangekoppelde verbindingen).

Twee benaderingen kunnen theoretisch toegepast worden bij het splitsen van datagrammen :

- **Intranet** fragmentatie : het opsplitsen en samenvoegen gebeurt per netwerk
- **Internet** fragmentatie : het (opsplitsen en) samenvoegen gebeurt in de **eindstations**.

Algemeen kent een host (/ router) enkel de MTU van het netwerk(/en) waaraan hij zelf is gekoppeld. Hij gaat dus fragmenteren op de grenzen die hem opgegeven zijn door het netwerk waaraan hij is gekoppeld → **intranet fragmentatie**.



Figuur 322 Fragmentatie a) intranet b) internet

- Een pakket dat vertrekt in de source IP moet over netwerk 1 gesplitst worden in verschillende datagrammen (bv.2) . Aangekomen in de router 1 wordt het eerst terug gereassembleerd door de IP laag ter plaatse en vervolgens gefragmenteerd in nog kleinere datagrammen (bv. 4) om over netwerk 2

gestuurd te worden. Router 2 reassembleert en fragmenteert op zijn beurt, IP op de destination reassembleert.

- b) Bij **internet**-fragmentatie zal de IP laag in de zender op dezelfde manier fragmentatie uitvoeren en versturen naar router 1 , maar deze zal **niet reassembleren**, enkele routeren. Mogelijk zal de router wel verder fragmenteren als hij beslist om het pakket een datalink op te sturen met een kleinere MTU. Meestal is dit intensieve werk echter niet gewenst voor de router en zal hij dit weigeren. In IPv6 wordt er per definitie NIET gefragmenteerd door de router. De datagrammen worden in alle geval enkel **gereassembleerd in de ontvanger**.

Conclusie : enkel **intranet** fragmentatie zal de maximum pakket size van de netwerken gebruiken, internet niet. Maar ... het reassembleren zou téveel tijd vragen van de internet-routers. Ook het verlies van een fragment zou leiden tot wachttijden (afh. v. TTL in IP-header) voor de reassemblering van het datagram in de routers en uiteindelijk leiden tot het verwerpen van het totale datagram. IP heeft geen ‘time out’ en retransmissie functie, TCP wel (UDP niet) en TCP zal eisen dat zijn totale segment (dat meestal overeenkomt met het totale IP datagram) wordt herzonden. Er is geen manier om een fragment te herzenden, daarom probeert men fragmentatie bij internet te **vermijden** door bvb. PATH-MTU discovery !!

#### Problemen :

- 2 ethernetnetwerken, gekoppeld door een router, waarbij op EN1 het **ethernetII** frame wordt toegepast en op EN2 het **802.3-802.2-SNAP** frame. Als een host1 op EN1 een IP-pakket stuurt naar een host2 op EN2, dan is het mogelijk dat de router moet fragmenteren als host1 de MTU van zijn datalink heeft toegepast : **1500** bytes IP pakket. EN2 kan slechts **1492** bytes transporteren in 1 frame → fragmenteren.
- IEEE 802.5 en FDDI netwerken hebben MTU's van **4500** tot **>8000** bytes. → Aanbevolen wordt dan om IP pakketten niet groter dan **576** bytes te maken, de **internet MTU**.

#### Werking :

Het fragmenteren en reassembleren gebeurt op basis van het '**identification**' veld in de IP header dat een unieke waarde bevat voor elk oorspronkelijk opgesteld IP datagram. Het origineel bevat bovendien een '**total length**' veld dat het **volledige pakket** omvat, **offset** = 0 en **More flag** = 0.

Een router die dit pakket moet fragmenteren zal :

1. Een aantal nieuwe datagrammen genereren met de oorspronkelijke header. (**¶'identification'** van alle fragmenten is **identiek!**)
2. Het dataveld wordt zo gelijkmatig mogelijk verdeeld op een 8 byte grens.
3. In het 1<sup>e</sup> nieuwe fragment wordt het '**total length**' veld aangepast en de '**More flag**' = 1.
4. In het 2<sup>e</sup> (... of volgende) wordt ook '**total length**' veld aangepast en '**More flag**' = 1, ... + **offset** = de **lengte van fragment1/8**. (... of de totale lengte van de vorige fragmenten/8)
5. In het laatste fragment : ¶ pt. 4 maar de '**More flag**' = 0.

M.a.w. Het **identification**' veld wordt **gekopieerd** in elk fragment en de vlaggen ('more fragments') worden aangepast := 1<sup>l</sup> voor alle fragmenten behalve het laatste. Het 13 bit "**fragment offset**" veld geeft

de **plaats** (in 8-byte eenheden!) van elk fragment in het oorspronkelijke datagram (maximum 65535 bytes). Het ‘total length’ veld van een fragment wordt uiteraard ook aangepast. Hierna moet de header checksum worden herrekend.

Elk fragment wordt afzonderlijk gerouteerd en kan dus ‘out of order’ aankomen in de bestemming. Er is voldoende informatie in de header om de reassemblering door IP toe te laten : fragmenten die arriveren met dezelfde ID worden op de juiste plaats in een buffer geplaatst tot het volledige datagram is samengesteld.

Een probleem is dat IP een CLNS is en dat sommige fragmenten mogelijk niet aankomen. Er zou dan  $\infty$  lang moeten gewacht worden... → er moet een **timeout** mechanisme in werking worden gesteld dat een maximum tijd instelt voor het aankomen van alle fragmenten. Wordt dit overschreden dan worden alle fragmenten verworpen. Deze timeout werkt op basis van de TTL van de aangekomen fragmenten : deze TTL wordt voortdurend gedecrementeerd door de reassembly functie. Wordt TTL=0 → timeout.

We merken dat dit fragmentering/reassemblering - proces veel tijd en bronnen in de routers in beslag zal nemen waardoor het, indien mogelijk, moet worden vermeden. Bovendien zou dit betekenen dat alle fragmenten **dezelfde weg** moeten afleggen → geen dynamische routering! Het Internet zal enkel reassembleren op de **bestemming**.

**D-vlag** = Don’t fragment vlag. Als in de IP header de D-vlag = 1 → dan mag het datagram niet worden gefragmenteerd. Dit kan bvb. gebruikt worden als de bestemming niet kan reassembleren. Routers moeten het datagram verzenden langs een interface die het volledig kan inpakken, zoniet wordt het verworpen.

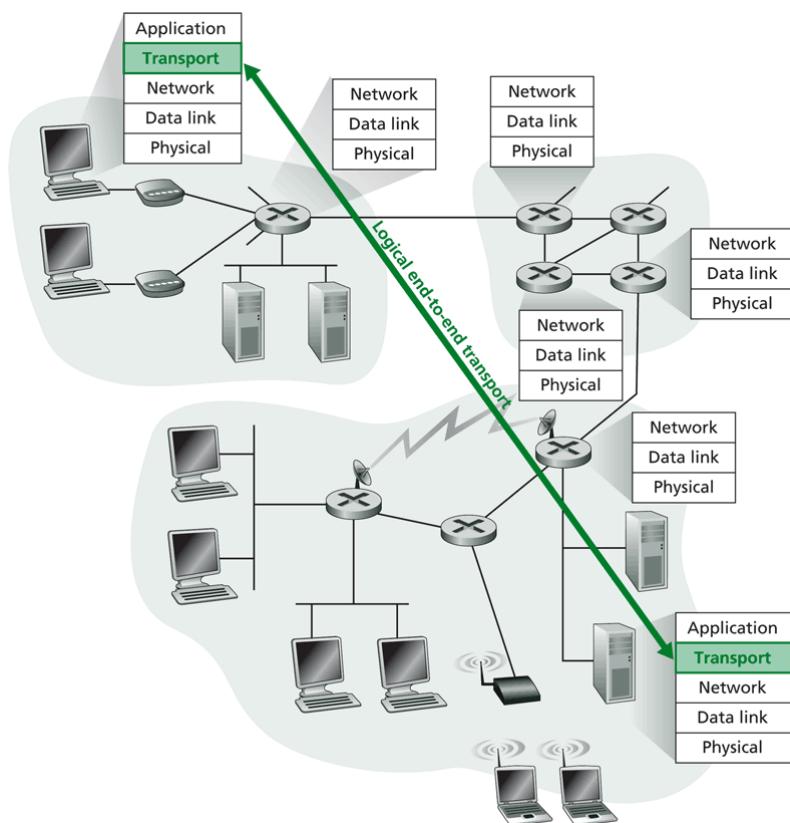
Dit principe wordt gebruikt voor MTU path discovery door TCP. Als transportlaag kan TCP eerst de MTU opzoeken van het volledig af te leggen pad door steeds grotere IP-pakketten op te sturen met de ‘Don’t fragment’ vlag =1. Op een bepaald moment zullen de pakketten worden verworpen en ge’timeout’ door TCP, of TCP ontvangt een ‘error message’ van ICMP. Een keer de PMTU gekend is zal TCP zijn segmenten zo opdelen dat IP niet meer moet fragmenteren.

# 4 DE TRANSPORTLAAG.

De transportlaag, laag 4, bevindt zich uitsluitend op de **end-systemen**. Ideologisch communiceren beide transportlagen rechtstreeks.

Alle tussenliggende netwerk apparaten hebben slechts functionaliteit tot < L4 :

- L1 (hubs/repeaters),
- L2 (bridges/switches)
- L3 (routers/gateways).

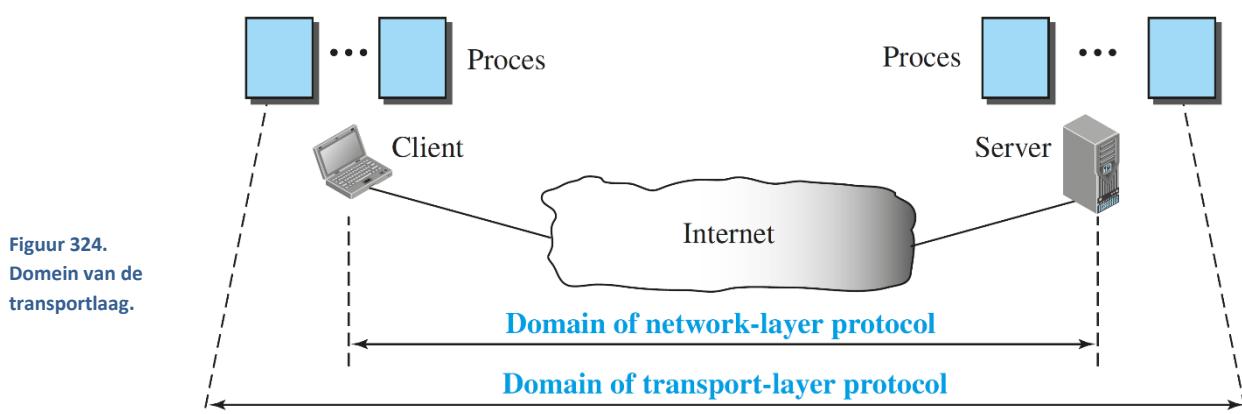


Figuur 323 De werkbare lagen in een netwerk

Het werk domein van de transportlagen bevindt zich tussen de verschillende applicatieprocessen **op de computers**, en enkel daarop. Niet op de tussenliggende netwerkapparaten.

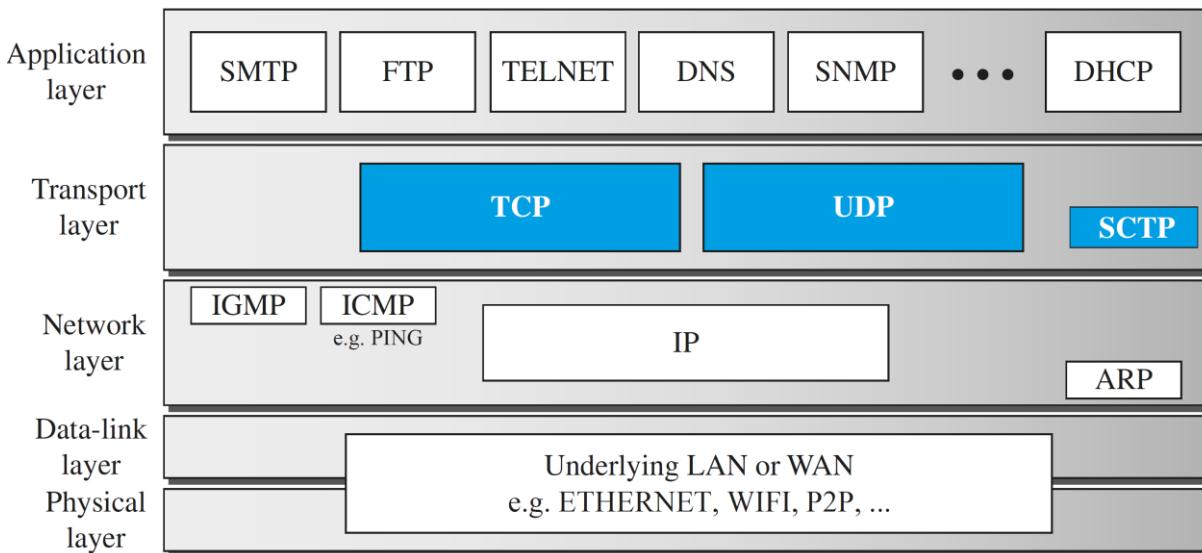
Het realiseert “**process-to-process communication**”: intern in de computers dient de transportlaag dus om de data af te leveren bij het goede proces, een soort ‘**interne routering**’.

De **netwerklaag**, die zorgt voor de ‘**externe routering**’ start vanaf de interfaces.



TCP en UDP<sup>1</sup> zijn de 2 belangrijkste transportlaag protocols die de applicaties rechtstreeks ondersteunen.

<sup>1</sup> Recent, met de opkomst van streaming services, is er een 3<sup>e</sup> protocol SCTP bij gekomen, maar dit valt buiten het bestek van deze cursus.



Figuur 325 De TCP/IP protocol suite

Beiden gebruiken ze **IP** als **netwerklaag**, onafhankelijk of de onderliggende laag een LAN, WAN of een internettwerk is.

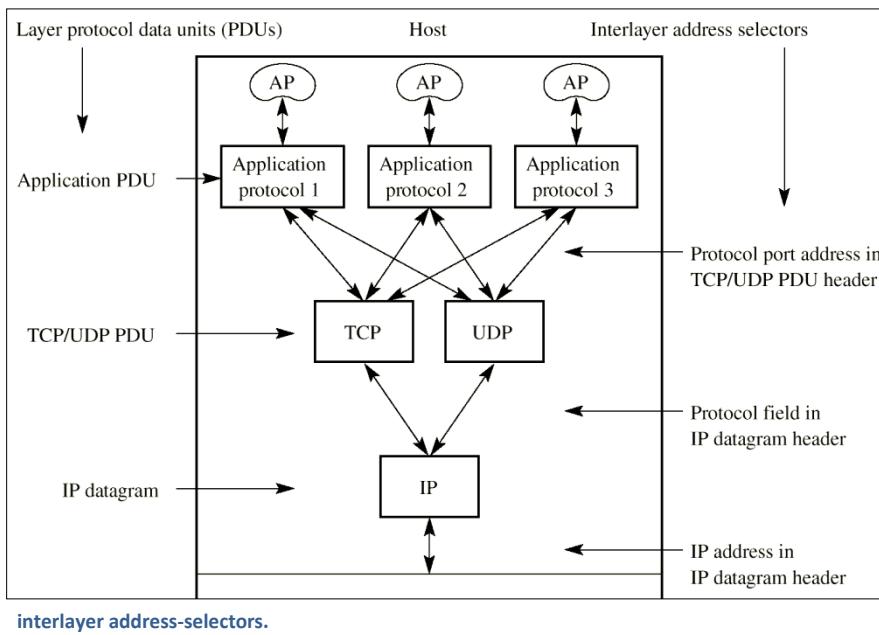
## WAAROM 2 TRANSPORT-PROTOCOLS?

- **TCP** is een zeer **uitgebreid** protocol dat **foutvrij** transport garandeert via een **CO** service. (**Connection-Oriented**, zie later). TCP = ‘**reliable**’.
- **UDP** daarentegen is een zeer **minimaal** protocol. Het doet zijn best, maar garandeert niets. **Snelheid** is voor UDP veel belangrijker dan betrouwbaarheid. UDP = ‘**unreliable**’, ‘best try’ verkeer. UDP = **CL** service. (**Connection-Less**).

Niet alle applicaties eisen betrouwbare verbindingen : bij video- of audio- informatie is het niet zo erg dat er al eens een fout optreedt, de **snelheid** daarentegen is wel belangrijk → men noemt dit wel eens '**isochroon**' verkeer. Daarom mag UDP enkel gebruikt worden voor applicaties die af en toe een fout toelaten : voice, video, DNS, ... In alle andere gevallen wordt **TCP** (= CO) gebruikt, bv. FTP : File transfert, maar ook HTTP : webverkeer, mail : SMTP (simple mail transfert protocol), TELNET (TELeType NETwork).

TCP staat dan ook voor een schijnbaar onmogelijke taak : het moet gebruik maken van IP, een **onbetrouwbare** netwerkdienst, maar biedt de toepassingsprogramma’s een betrouwbare data-aflevering. Het moet **verlies**, **duplicatie** of **vertraging** van data in het internet compenseren zonder de onderliggende netwerken al te zwaar te beladen. Hij biedt bovendien deze data aan in **dezelfde volgorde** als ze is verzonden.

De transportlaag biedt een end to end dienst aan door het verzorgen van een ‘directe’ verbinding van de toepassing op de ene computer met de toepassing op een ‘remote’ computer. We bespreken eerst de eenvoudigste, CL verbinding : UDP, en daarna, veel uitgebreider CO : TCP.



Applicaties hebben de keuze tussen TCP en UDP als transport-protocol.

In werkelijkheid liggen die keuzes redelijk vast. Ze hebben typisch een voorkeur voor 1 van de 2.

Ze hebben bv. een voorkeur voor foutloos transport (TCP) zoals FTP, SMTP, ... of een snel transport zoals DNS, DHCP, ...  
→ UDP .

Figuur 326 TCP en UDP stack met de interlayer address-selectors.

#### ANDERE NOEMENSWAARDIGE PROTOCOLLEN :

- **ICMP** (Internet Control Message Protocol) is een toevoegsel aan IP. Het wordt gebruikt door IP om **foutberichten** en andere vitale informatie door te geven. Vb. netwerken die niet kunnen bereikt worden. De IP laag zal **proberen** (best try) de zender op de hoogte te brengen.  
Ping en traceroute, 2 populaire diagnose-tools, gebruiken ICMP : vanuit een dosvenster...

C:\...> ping [www.uantwerpen.be](http://www.uantwerpen.be)

C:\...> tracert [www.uantwerpen.be](http://www.uantwerpen.be)

(als de tussenliggende firewalls dit toelaten. Kan ook rechtstreeks op IP adressen gebeuren)

- **IGMP** (Internet Group Management Protocol) wordt gebruikt bij multicasting om een UDP datagram naar verschillende gebruikers te zenden. Bv. digitale TV zenders.
- **ARP** (Address Resolution Protocol) is een speciaal protocol dat gebruikt wordt bij sommige netwerkinterfaces (zoals ethernet en token ring) voor de conversie van IP adressen naar MAC adressen (en omgekeerd).

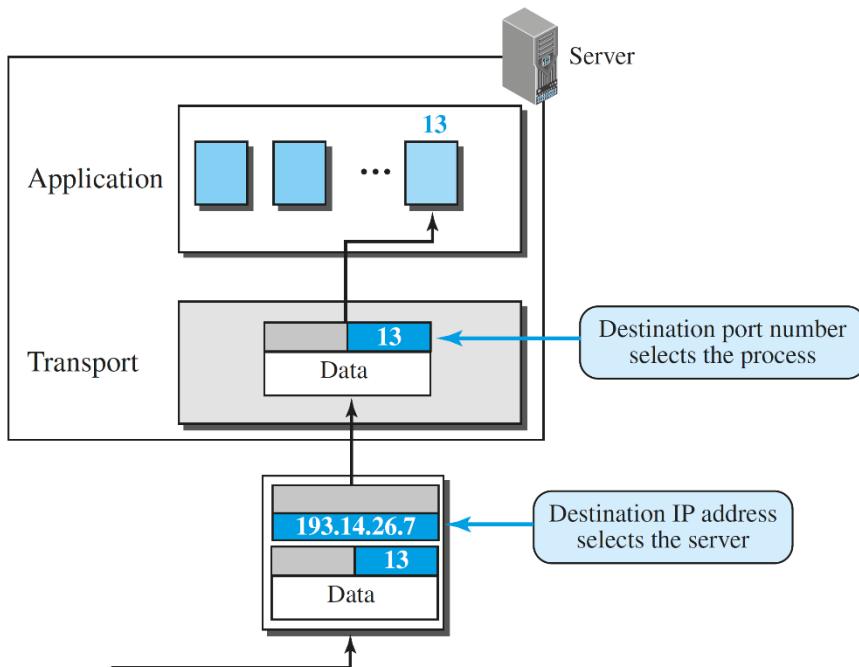
#### DE APPLICATIES UIT DE FIGUUR ZIJN :

- **SMTP** : Simple Mail Transfer Protocol
- **FTP** : File Transfer Protocol, laat gebruikers bestanden uitwisselen
- **Telnet** : TELetyp NETwork, is een protocol dat het mogelijk maakt op afstand in te loggen op een machine, bv. een router, en die via een opdrachtregel 'command line' te besturen
- **SNMP**: dient om management- en log - informatie van netwerktoestellen te verzamelen.
- **Ping** : onderzoekt of een host bereikbaar is of niet door een ICMP pakket te zenden en er een terug te verwachten.
- **DHCP** en **DNS**: zijn reeds besproken.

## 4.1 L4 ADRESSEERING : PORTS

Zoals ook te zien is in Figuur 324 op p. 303 moet een transportlaag binnenin 1 computer het geschikte proces vinden/aanduiden. Het proces dat de communicatie realiseert met een tegenpartij en hierbij gebruik maakt van de transport services van laag 4. Maar 1 computer kan verschillende processen of applicaties tegelijk draaien, allen met afzonderlijke datastromen naar verschillende bestemmingen. De **poortnummers** van laag 4 moeten de **interne routering** verzorgen van de binnengkomende IP pakketten.

M.a.w. de netwerklaag (en alle onderliggende lagen) zorgen er dus voor dat een IP pakket aankomt op 1 fysieke computer (via een interface), en binnenin die computer zal laag 4 de data die in het IP pakket zit afleveren aan het exacte proces, aangeduid via de poortnummer.



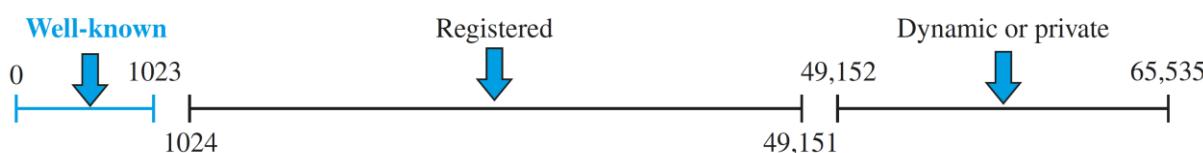
Figuur 327. Transportlaag interne routering via poortnummers.

Er zijn een aantal manieren om een process-to-process communicatie op te zetten, maar meestal is dit een '**cliënt -server**' opzet. Het initiatief voor de verbinding start dan **vanuit de cliënt**, ('active open') gericht naar een server die continu 'open staat' voor aanvragen (passive open) en die zal reageren op een vraag voor verbinding.

Maar hoe weet een cliënt naar welke poort een applicatie luistert op een (ver) afgelegen **server** die hij wenst aan te spreken? Dat zijn nu '**well known ports**', of systeempoorten die universeel vastgelegd zijn door IANA.

En welke poort gebruikt de **cliënt** zelf om zijn proces aan te duiden? Dit zijn wat men noemt de '**dynamische poorten**'. Dynamisch omdat ze een korte levensduur hebben: namelijk net zo lang als de communicatie 'staat'.

Poortnummers zijn 16-bit getallen (zie later) die dus een integer aanduiden tussen 0 en 65535. Dit bereik wordt nu 'georganiseerd' door IANA in volgende delen:



Figuur 328. IANA poort nummer indeling

- **'Well-known'** poorten. Gelegen tussen 0 en 1023. Het zijn serverpoorten vastgelegd door IANA. Bv. 80 voor HTTP.
- **Registered** poorten. Van 1024 tot en met 49151, ook wel "Gebruikerspoorten" genoemd. Ze worden (min of meer) vast gebruikt en zijn in het beste geval eveneens geregistreerd bij IANA. Bv. 3389 voor RDP.
- **Dynamische** poorten. De poorten vanaf 49,152 tot 65,535 zijn vrij te gebruiken door de cliënt s.

Een uitgebreidere lijst van deze poorten kan je ook vinden op <https://nl.wikipedia.org/wiki/TCP- en UDP-poorten>.

### SOCKET ADDRESS.

De samenstelling van **IP-adres** en **UDP** of TCP poort noemt men een '**socket**', en identificeert uniek elk **applicatieproces** in een computer, wereldwijd. Zowel bij de cliënt als bij de server.

200.23.56.5, UDP 53  
computer 5 op netwerk 200.23.56.0 (kl C),  
→ protocol = UDP, poort 53  
DNS service.



Figuur 329. een socket address

Om te zien welke sockets er bij uw computer open staan :

`c:>netstat -a`      of `c:>netstat -n`      (of probeer andere opties → `c:>netstat /? : -b, -o`)

**Servers** hebben over dus een overeengekomen, '*well known*' poortnummer onder de **1023** met een voorkeur voor een transportprotocol. vb. :

- |   |                                   |
|---|-----------------------------------|
| ▪ UDP poort 53 : DNS                                    | ▪ TCP poort 20 : FTP servers data |
| ▪ UDP poort 67 en 68 : DHCP resp. server en cliënt .    | ▪ TCP poort 21 : FTP control      |
| ▪ UDP poort 69 : TFTP (Trivial File Transfert Protocol) | ▪ TCP poort 23 : Telnet servers   |
| ▪ UDP poort 520 : RIP en RIP 2                          | ▪ TCP poort 25 : SMTP             |

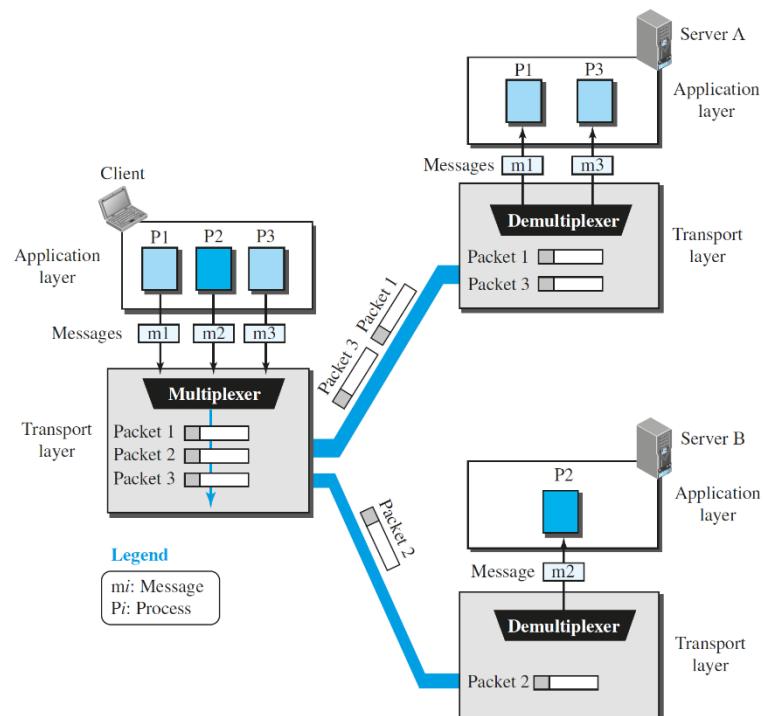
## MULTIPLEXEN

Een transportlaag zal dus d.m.v. poortnummers de gepaste applicatie bereiken voor de continue stroom van IP pakketten die binnenkomen.

Stel een cliënt die 3 actieve processen heeft:

- met server A: P1 en P3,
- met server B: P2.

De servers worden onderscheiden door hun IP adressen, de processen door hun poorten m<sub>i</sub>.



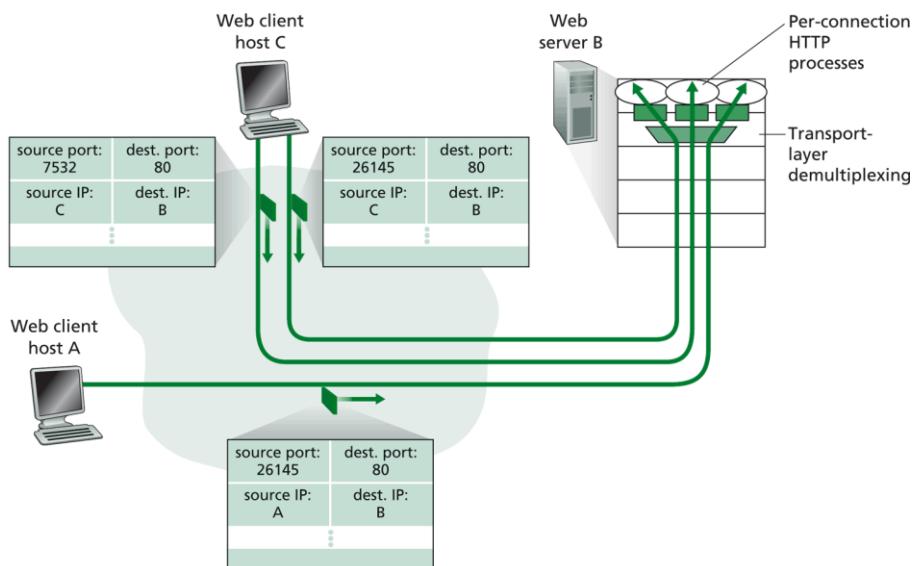
Figuur 330. Applicatie multiplexing in de transportlaag.

Let op, het is de **volledige** (dubbele) socket die uniek bepaalt wat de communicatie eindpunten juist zijn.

Een cliënt kan bv. tegelijk 2 connecties opzetten met 1 HTTP: (web) server B. (HTTP = poort 80).

Een 'cliënt' is niet geïnteresseerd in het exacte nummer van zijn 'source port', zolang het maar uniek is gedurende gebruik.

Ze worden **dynamisch** toegekend, normaal nu in het bereik tussen 49152 tot en met 65535 .



Figuur 331 TCP en UDP sockets bij source en destination.

In bovenstaande figuur zal host A een webpagina opvragen bij webserver B. A stuurt een IP pakket naar  $IP_B$  met daarin een TCP segment dat een dest. poort = **80** opgeeft → de afgesproken poort voor ‘webservices’<sup>1</sup>. A kiest zelf dynamisch een antwoordpoort, bv. 26145.

Dit pakket komt aan bij B en wordt ge’demultiplext’ naar het **http-server** proces (bv. ‘apache-web server’). Dit proces zal antwoorden naar A’s poort 26145. M.a.w. in de antwoordpakketten zijn source- en destination- **IP-adres** én – **TCP-poorten gewisseld!**

Één gebruiker/machine kan ook **meerdere** webpagina’s opvragen bij dezelfde server, zoals host C. Hij moet er dan wel voor zorgen dat hij **andere antwoordpoorten** voorziet. Een volledige socket bestaat dus eigenlijk uit de 4 velden uit de figuur. Van zodra 1 veld verschilt spreekt men een andere socket aan.

We hebben dit TCP voorbeeld, http, genomen omdat dit waarschijnlijk het meest tot de verbeelding spreekt. De werking voor UDP sockets is uiteraard volledig identiek.

---

<sup>1</sup> Men moet server poorten ‘afspreken’ aangezien servers niet het initiatief nemen om een connectie op te zetten, dat doet de cliënt. Een server staat te wachten op aanvragen. Een cliënt kan niet ‘gokken’ naar de te gebruiken poort, vandaar de afgesproken poortnummers die gebruikt worden bij servers, meestal < 1024.

## 4.2 UDP USER DATAGRAM PROTOCOL

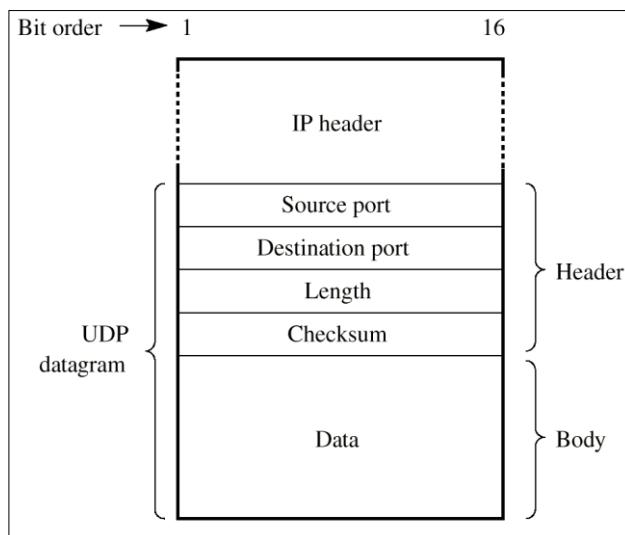
UDP is een eenvoudig, CL **transport** protocol, altijd gelegen bovenop IP. **Eén** boodschap van een bovenliggend proces creëert **één** UDP **user datagram** wat verpakt wordt in **één** IP datagram. Er wordt geen verbinding opgezet wat de overhead minimaliseert. Er zijn geen nummers die de pakketvolgorde aanduiden: die is er niet.

UDP doet **één** goede poging, ‘best try’, CL → voor simpele vragen zoals bvb. DNS. Als er iets mis gaat, moet de **applicatie** de vraag opnieuw stellen. Bij TCP doet de **transportlaag**, TCP zelf, dat.

In de UDP-header zullen 16 bit **poortnummers** de applicatie identificeren die zendt of ontvangt. Dit is voor TCP/UDP de **NLPID** = ‘*Next Layer Protocol Identifier*’.

### HET UDP DATAGRAM.

Het UDP **lengte** veld definieert de lengte van het volledige UDP datagram : **header én data**.



De UDP **checksum** is optioneel en, indien niet gebruikt, = 0. Ze wordt berekend over **header én data**. Indien de checksum niet klopt wordt het datagram simpelweg verworpen zonder verder verhaal.

(*Indien een applicatie toch error-controle wenst zal het ofwel TCP moeten gebruiken, ofwel het zelf voorzien, bv. TFTP*).

Dit is buiten de mux/demux werking van de poorten het enige fundamentele dat UDP (evt.) bijdraagt: : een evt. integrale **foutencontrole** op de **data**.

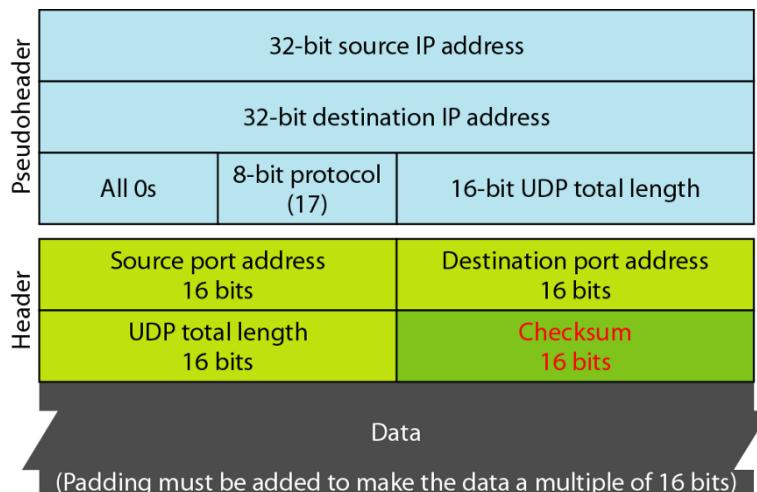
Figuur 332 UDP datagram formaat

Daarbij hoort nog 1 aanvulling:

Als er toch foutencontrole wordt uitgevoerd door UDP zal hij ook een deel velden van de IP header meenemen in de checksum: de **pseudoheader**.

Dit vooral om te voorkomen dat de **socket** fouten zou bevatten. Dus moet hij de IP adressen meenemen en de IP - NLPID, hier de protocol byte die aangeeft UDP vs TCP, ... .

Voor de berekening van deze checksum, zie bespreking van de TCP header op p.315.



Figuur 333 pseudoheader bij de berekening van een UDP (of TCP) checksum.

## 4.3 TCP TRANSMISSION CONTROL PROTOCOL

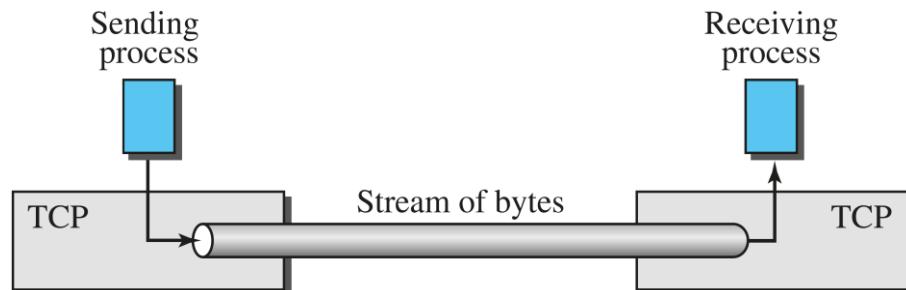
Daar waar UDP niet veel meer doet dan een ‘multiplex-functie’ van de applicaties is **TCP VEEL uitgebreider**. TCP voorziet op dezelfde manier deze applicatiemux, nu door TCP-poorten, zoals beschreven in 4.2 UDP User Datagram Protocol, op p. 310 maar daarnaast ook veel meer.

UDP (**CL** = connection less) verschilt duidelijk van TCP (**CO**= connection oriented) :

- **TCP** is een ‘**stream oriented**’ protocol: de **bytestroom** die vertrekt in de zender wordt gegarandeerd identiek aangeleverd in de ontvanger, zie Figuur 334 hieronder.
- de boodschap moet **opgedeeld** worden in segmenten: het kan hier gaan om grote bestanden (bv. FTP) die niet zondermeer in 1 IP pakket passen. (i.t.t. UDP). Er moet in stukken gedeeld worden, en elk stuk wordt apart verzonden.
- Dit verzenden gebeurt door **IP**, maar dat is een **CL**, ‘**best try**’ service.  
Dat moet dus gecontroleerd worden door TCP!:
  - o **Error-controle**: de checksum zal nu **verplicht** zijn  
+ ontvangst van een pakket moet (indien ok) bevestigd worden (**ACK**).
  - o IP kan pakketten **verliezen**. TCP moet dit merken en herzenden.
  - o Bij elk verzonden pakket start er een timer. De ACK ervan moet binnen zijn vóór de timer afloopt.
  - o Om te weten welk pakket wordt ge’ACK’ed moeten er **sequencenummers** worden toegepast!
  - o IP = CL → kan ‘out of sequence’ pakketten afleveren → TCP moet de bytestroom terug in volgorde krijgen → **sequencen**, ook m.b.v. deze nummers.
  - o Als je met nummers werkt moet je eerst afspreken vanaf welke nummer je begint... →
- Allereerst moet een **verbinding opgezet** worden: **CO!**. Bij deze opzet worden een reeks te gebruiken parameters afgesproken. Bv. de start van de sequencenummers.
- Ook wordt daarbij aangegeven **hoeveel buffers** er voor de communicatie worden voorzien.
- Deze buffers moeten op het einde terug vrij gemaakt worden voor het OS
- Een verbinding moet dus ook worden **afgesloten** om buffers en gebruikte poorten terug vrij te krijgen.

Je kan een TCP connectie voorstellen als een pijp tussen zender en ontvanger waarin bytes worden resp. geduwd en ontvangen (abstractie makend van het onderliggende netwerk → IP etc. ).

Er worden **geen markeringen** of grenzen van de individuele pakketten weerhouden → een ‘byte stroom’.



Figuur 334. TCP, een stroom service.

Omdat dit zend- en ontvangst proces niet noodzakelijk tegen dezelfde snelheid moet lopen (bv. snelle server, trage cliënt) moeten zend- en ontvangst-buffers voorzien worden door het OS / TCP.  
Dit in beide richtingen → 4 buffers!.

**TCP** is hét transportprotocol voor **betrouwbare** verbindingen en biedt de applicaties :

- **CO, Verbindingsgerichtheid** : elke connectie moet eerst opgezet worden tussen de 2 eindpunten en, na data, terug wordt afgebroken.
- **Point to Point communicatie** : Elke TCP-connectie heeft precies 2 eindpunten.  
Broadcasting of multicasting is **niet** van toepassing bij TCP.
- **Full duplex communicatie** : Eens een connectie is opgezet kan de data in **2 richtingen** vloeien
- **Volledige betrouwbaarheid** : TCP garandeert dat de data die via een verbinding wordt verstuurd, precies zo wordt aangeleverd als ze verzonden is, **zonder fouten en in volgorde**.
- **Elegante verbreking van de verbinding** : TCP garandeert dat alle data in betrouwbare toestand is aangeleverd voordat het de verbinding afbreekt.

**WERKING VAN TCP :**

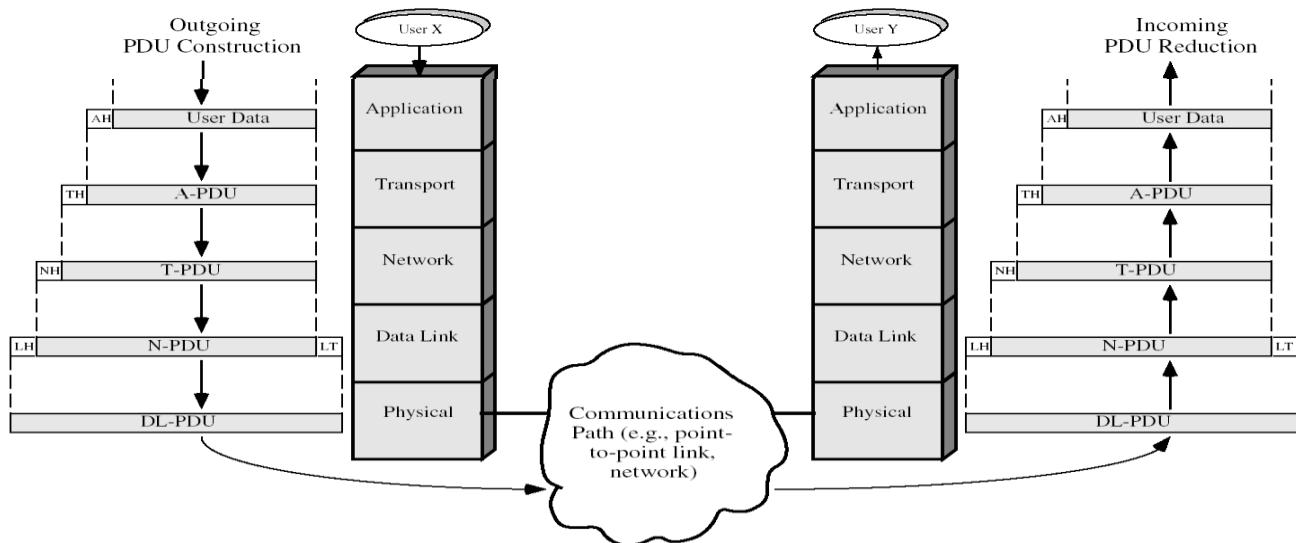
1. De data van de bovenliggende applicatie wordt in een buffer opgeslagen en **gebroken** in **segmenten** met een, volgens TCP, optimale grootte (= MSS, Maximum Segment Size,<sup>1)</sup>). Dit is in tegenstelling tot UDP waar de applicatie exact 1 UDP datagram genereert.
2. Elk verzonden segment creëert een **timer** en moet ge'ACK'nowledged worden door de ontvanger ( $\rightarrow$  duplex verbinding). Komt die ACK niet binnen de timer-tijd, dan wordt het segment herzonden.
3. TCP controleert in de ontvanger het segment op **transmissiefouten** door een **checksum** op **header én data**. Is de checksum foutief dan zal een TCP -ontvanger geen ACK terugzenden, hij verwacht een timeout en retransmissie van de TCP-zender.
4. TCP segmenten zitten verpakt in een IP datagram dat 'uit volgorde' kan aankomen  $\rightarrow$  TCP moet de volgorde herstellen om aan de applicatie aan te bieden : '**sequencing**'.
5. TCP segmenten kunnen zich vermenigvuldigen : bvb. een segment is vertraagd, ge'time-out' en herzonden, de ontvanger krijgt ze beiden  $\rightarrow$  TCP moet **dubbels** verwerpen.
6. TCP verzorgt data-flow : beide zijden hebben een beperkte buffercapaciteit. TCP verhindert dat een snelle host teveel data stuurt  $\rightarrow$  '**sliding windows**' .

De timeout wordt ingesteld op de RTT, Round Trip Time, van de connectie : segment v. Z  $\Rightarrow$  O, en terug ACK v. Z  $\Leftarrow$  O. Deze RTT wordt continue gevuld aangezien hij sterk kan variëren afh. v. de fluctuaties op het (inter)net. De timeout wordt gebaseerd op een gewogen gemiddelde van de laatste RTT's.

Op te merken valt dat er berichten kunnen geforceerd worden om **onmiddellijk** te worden afgeleverd ( $\rightarrow$  PSH-vlag) zonder te wachten tot de buffer bij de ontvanger vol is, bvb. een korte vraag voor data van een website.

---

<sup>1</sup> MSS wordt overeengekomen tussen de eindpunten bij het **opzetten** van de connectie.

INTERACTIE VAN TCP MET DE VERSCHILLENDEN SW LAGEN<sup>1</sup>.**Zender**

- L5 De **applicatie** geeft de data aan de transport layer (pointer van mem. buffer) met de vraag om dit te verzenden naar een bestemming: IP destination adres. (evt. eerst DNS).
- L4 **TCP** (bv.) verdeelt de data in stukken, TCP-segmenten, voegt een header toe met een **TCP-volnummer** en een 'port' i.f.v. de applicatie ( source & destination → IP= socket). TCP speelt dit door aan IP voor verzending en start een timeout-timer voor dit segment.
- L3 IP creëert een datagram met als **data** het TCP segment en voegt zelf een IP-header toe met source en dest. IP-adres.  
IP bepaalt vervolgens het fysische (MAC) adres van de 'next hop' (via ARP) en geeft dit alles door aan de datalink-laag.
- L2 **Datalink** verpakt het IP pakket in het datagedeelte van een frame en verzendt het naar de 'next hop' = de eerstvolgende router of de uiteindelijke bestemming (via MAC-adres).

**Ontvanger**

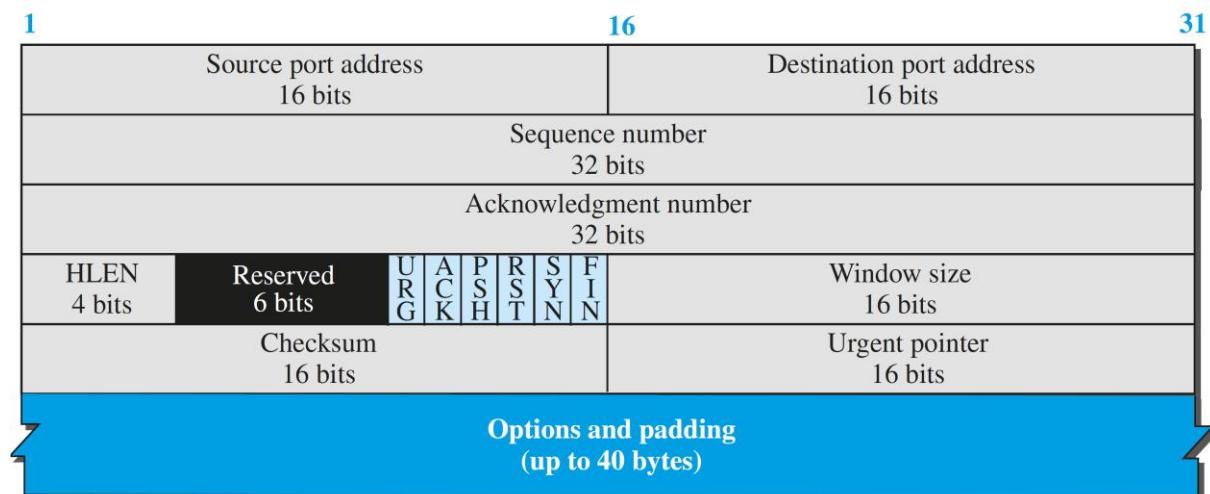
- L2 De **datalink** controleert zijn CRC. Indien OK checked hij de **NLPID?**<sup>2</sup> : in de L2 header staat het typeveld → = IP ? Ja → datagedeelte naar IP-laag.
- L3 IP laag controleert IP-datagram-header en checksum. ∀ OK → **NLPID?** : protocol = TCP → datagedeelte naar TCP-laag.
- L4 **TCP** berekent de checksum van het TCP segment data en header.  
Indien OK → zendt een ACKnowledge terug naar de zender (**CO!**).  
TCP verzamelt alle segmenten op basis van de volgnummers, error- en flowcontrole, en speelt de exacte data in de goede volgorde door via de 'port/socket' aan de applicatie.
- L5 De **applicatie** ontvangt de data van de transport layer in **exact dezelfde stroom** als de zendende applicatie ze heeft verzonden.

<sup>1</sup> We bespreken het veel uitgebreider in 5.2 overzicht TCP/IP over EtherNet op p. 372

<sup>2</sup> NLPID = 'Next Layer Protocol Identifier'. In de header van elke laag staat een aanduiding welke 'payload' de laag transporteerd, m.a.w. van welke bovenliggende laag de data afkomstig was, ... en dus teruggestuurd moet.

### 4.3.1 TCP HEADER

Om al deze functies feilloos te kunnen uitvoeren heeft TCP een veel uitgebreidere header dan UDP. Direct na de (normaal) 20 bytes IP header volgt de (normaal) **20 bytes** TCP header.



Figuur 335 TCP segment header en codebits

- De **poortnummers**: 16bit source- en destination. Ze dienen net zoals bij UDP, om de zendende en ontvangende **applicatie** te identificeren via het **socket address**: de poorten, tezamen met het zender- en ontvanger- IP adres definiëren **uniek** de 2 eindpunten van een wereldwijde internet-connectie.
- De '**sequence number**': heeft de functie van het '*send sequence*' nummer. Maar aangezien TCP het geheel samenstelt tot een **datastream** waar geen berichtgrenzen meer zichtbaar zijn, definieert de 'sequence number' niet een 'segment nummer' maar een '**byte nummer**' in de bytestroom.  
Meer bepaald: de plaats van de eerste byte uit dit segment in de oorspronkelijke 'byte stream'.

Aangezien het kan gaan over ellenlange 'streams', bv. bestanden van mega/gigabytes, is een **32 bit** pointer als sequence nummer niet overdreven.

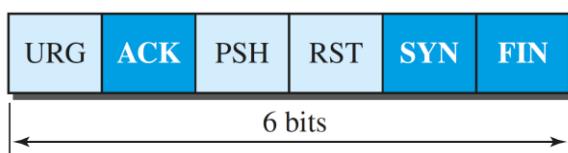
Het wordt gebruikt als een 32 bit ongetekend nummer als byte teller: → tellen tot  $2^{32}$  (of  $\pm 4 \cdot 10^9$ ) dat terug begint op 0 na  $2^{32}-1$ .

- De '**acknowledgment number**' : Net zoals de 'sequence number' de functie van *zendnummer* uitoefent, heeft dit de rol van **ontvangstnummer** : Meer bepaald:  
hij duidt de volgende byte in de stroom aan die de ontvanger van dit segment wenst te ontvangen.  
Hij doet dit in de datastream in de andere richting ('piggy back').  
De ACK-nr. is enkel geldig als de ACK-vlag = 1.

Eens de connectie is opgezet verzorgt TCP een full duplex verbinding : data kan in 2 richtingen vloeien → elk eindpunt moet een zend- én ontvangstnummer bijhouden.

TCP werkt meestal op een **Go back N** principe zonder Negatieve ACK, géén selectieve ACK, voor meer info zie pt. 4.3.7, TCP implementatie policy's. op p. 342.

- De 4-bit 'data offset' of 'header length': geeft de lengte van de TCP header in 32 bit woorden. Normaal = 5 (→ 20 byte) kan hij maximaal 15 zijn (→ 60 byte) als er opties zijn. (vooral bij setup, zie verder.)
- De gereserveerde bits zijn ... gereserveerd voor toekomstig gebruik.
- Er zijn **6 vlaggen**: die onafhankelijk van elkaar zijn. Sommige geven aan of de pointers die volgen in de header geldig zijn. Als een vlag = 1, staat ze. Er kan meer dan 1 vlag tegelijk staan.



URG: Urgent pointer is valid  
 ACK: Acknowledgment is valid  
 PSH : Request for push  
 RST : Reset the connection  
 SYN: Synchronize sequence numbers  
 FIN : Terminate the connection

- URG : de urgent pointer is geldig : er is dringende data tussen (naast) de normale stream gevoegd die onmiddellijk moet beantwoord worden, bvb. een request voor data.
- ACK : de ACK nummer is geldig → bevestiging van verkeer in de andere richting.
- PSH : de ontvanger moet dit segment (ASAP) direct doorgeven aan de applicatie. Hij moet niet wachten op een volle ontvangstbuffer. Bvb. bij interactief verkeer zal PSH  $\forall=1$ , voor file transfer zal alleen op het laatste segment PSH=1.
- RST : reset de connectie
- SYN/SEQ : SYNchroniseer de zend- en ontvangstnummers om een connectie **op te starten**, sporadisch ook SEQ vlag genoemd naar de SEQuence nummers die daarvoor gebruikt worden.
- FIN : de zender stopt met data te zenden.
- In vgl. met de standaard TCP header van Figuur 335 op vorige pagina zijn er recent in RFC 3168 nog 2 vlaggen gedefinieerd die in overleg, bij de opstart, kunnen worden afgesproken. Dit komt in werking bij **ECN**, Explicit Congestion Notification. Ze worden enkel gebruikt indien beide partijen het erover eens zijn.



Figuur 336. Explicit Congestion Notificatin vlaggen in TCP.

Indien een **ontvanger** merkt (via IP vlaggen, zie volgende cursus 5 infrastructuur bij diffserv werking) dat er onderweg **congestie** is opgetreden zal hij dit signaleren aan de **zender** in de TCP-vlag **ECE**: 'Explicit Congestion Echo'. (bij de ACK v.e. segment). De **zender** kan dan berichten dat hij dit idd. heeft ontvangen en dat hij zijn snelheid van zenden heeft **aangepast**: **CWR** = 'Congestion Window Reduced'. Zie ook verder in deze cursus: 4.3.4 TCP congestion controle op p.332.

- Het 'window' veld: geeft de grootte van het venster ('sliding window') principe weer. Opgeteld bij het ACK nummer geeft het weer tot welke bytenummer in de stroom de ontvanger kan ontvangen. Of

m.a.w. het geeft weer hoeveel bytes er nog vrij zijn in de ontvangstbuffer. Het veld is 16 bit → max 65535 bytes.

Het wordt typisch gebruikt bij flow- (en congestion-) controle om te vermijden dat een snelle zender (server) een tragere ontvanger zou overstelpen met data.

- De ‘checksum’: is hier **verplicht** en idem als UDP. Ze wordt berekend over header én data.

Berekening bij de zender:

1. De boodschap (pseudoheader, header én data) wordt opgeliist in **16 bit** woorden.
2. Al deze 16-bit getallen worden opgeteld in 1CC (one's complement, geen carry's)
3. De som wordt gecomplementeerd en ingevuld in dit veld, 16 bit.

Berekening bij de ontvanger is **identiek** als bij de zender, maar nu **mét** de opgezonden checksum inbegrepen. De uitkomst moet dan  $\forall 1's$  zijn indien er geen fouten plaatsvonden. Bekijk evt. Figuur 271. Een 16-bit checksum berekening, bv. van de IP header. op p. 227 hoe dit in zijn werk gaat.

Net zoals bij UDP worden voor de berekening velden uit de IP-header meegerekend (S.A, D.A., prot & length) → ‘pseudoheader’ om te controleren dat het datagram wel bij de exacte recipiënt is aangekomen : TCP beschermt zich tegen de ‘unreliable’ service van IP.

Zie ook Figuur 333 op p. 310 voor de velden van de pseudoheader.

- De ‘urgent pointer’ : is enkel geldig als de URG vlag staat. geeft de plaats aan t.o.v. de sequence nummer waar de laatste byte van de dringende data staat. M.a.w. hij geeft aan hoeveel *urgent data* er in het segment zit. Deze data moet **direct** door TCP afgeleverd worden aan de applicatie van zodra het is ontvangen.
- De meest gebruikte ‘option’ is de MSS : Maximum Segment Size. Beide connecties specificeren dit in het eerste uitgewisselde segment (het segment met de SYN vlag = 1), het opzetten van de connectie. Het geeft aan wat de max. segment-grootte is dat de zender wenst te ontvangen. Het zou ideaal moeten overeenkomen met de kleinste MTU van de tussenliggende netwerken zodat IP de segmenten niet moet fragmenteren.

Als er WAN verbindingen in de route aanwezig zijn wordt dit veel op **536** databytes gehouden ( $\rightarrow 20 + 20 = 576$  bytes IP datagram) omwille van de relatief hoge BER.

Voor LAN's, bv. ethernet, ligt dit hoger: **1460** byte (= 1500 – 20 – 20) , IEEE 802.3 : 1452 byte..

Het dataveld is optioneel, bvb. bij het opzetten- en afbreken van een connectie, of een ACK-segment zonder data is er geen verkeer in die richting.

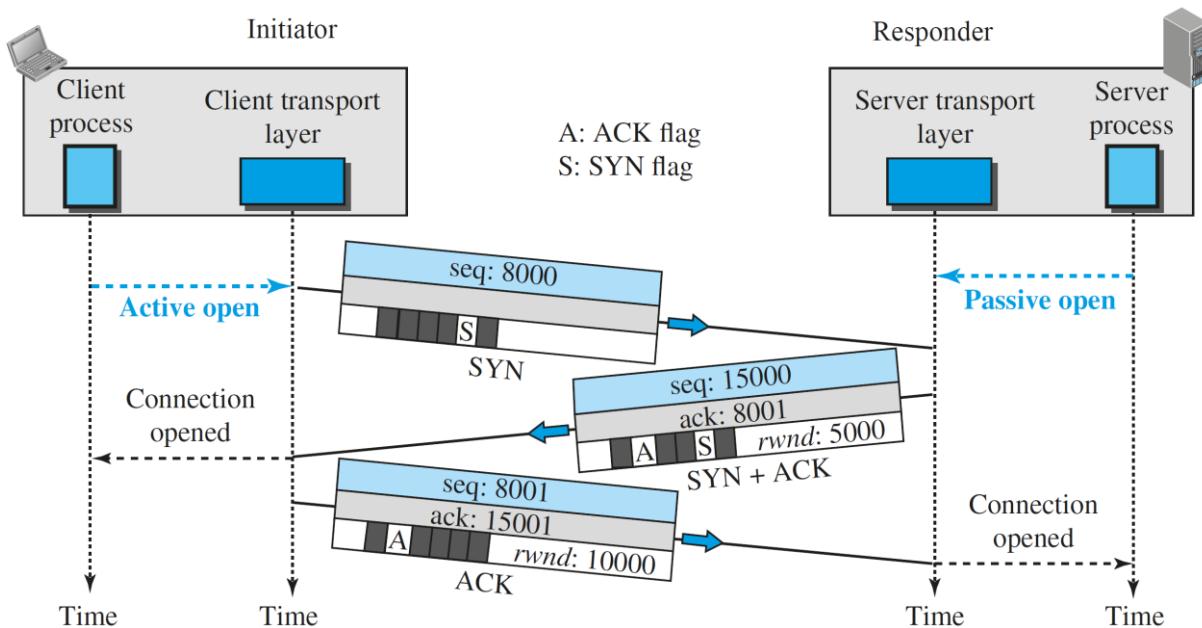
### 4.3.2 TCP PROTOCOL WERKING.

#### HET OPZETTEN VAN EEN CONNECTIE.

Zoals we reeds uitvoerig besproken hebben is **TCP Connection Oriented = CO**. Iemand moet het initiatief nemen om de connectie op te zetten, dit noemen we de cliënt. Hij neemt contact op met een server die van bij zijn opstart de ‘well known’ poorten van zijn service (bv. webservices voor een webserver) bewaakt en elke aanvraag beantwoordt. Hij staat continu ‘passive\_open’.

Er moeten bij de connectie-setup een aantal afspraken gemaakt worden om dit complexe protocol te ondersteunen (i.t.t. UDP → CL = ‘connection Less’).

Er vindt daarom een ‘**three way handshake**’ procedure plaats : **SYN** - **SYN/ACK** - **ACK**.



Figuur 337 Het opzetten van een TCP-connectie.

1. De start van de connectie gebeurt door het zenden van een **SYN**-segment van de cliënt → “hello server  $IP_d$ , ik zou met u informatie willen uitwisselen”.
2. De server meldt dat hij de aanvraag goed heeft binnengekregen → **ACK** + gaat akkoord met de aanvraag en stuurt een **SYN** terug → **SYN-ACK**.
3. De cliënt meldt de goede ontvangst van segment 2 → **ACK** en de connectie is opgezet. Hij meldt ook aan de applicatie dat het opzetten van de connectie gelukt is: ‘Connection opened’.

Gedurende dit proces worden ook een aantal parameters geïnitialiseerd zoals **sequencenumbers** in beide richtingen ( $8000 \Rightarrow$  en  $\Leftarrow 15000$ ) om de volgorde van de (volgende) segmenten te kunnen bepalen. Hierbij wordt gebruikt gemaakt van de seq.nr. in de TCP-header : als  $SYN=1 \Rightarrow$  seq.nr. = ISN.

De eerste effectieve bytes zullen deze **ISN ‘Initial Sequence Number’ +1** zijn: ( $8001 \Rightarrow$  en  $\Leftarrow 15001$ ). Dit is om de ACK’s toe te laten een onderscheid te maken tussen een SYN segment en het eerste datasegment.

De client zendt een segment met de SYN vlag gezet en een voorstel voor een sequence number : 8000, =ISN.

De server noteert dit nummer als ontvangstteller voor het binnenkomend verkeer en zendt een segment terug met opnieuw een sequence number : 15000 voor het verkeer in de andere richting. Het segment bevat zowel de SYN vlag als de ACK vlag wat een geldig acknowledgement nummer betekent, = 8001 → een akkoord voor het voorstel van de client. Merk op dat de server een segment verwacht beginnend met byte 8001.

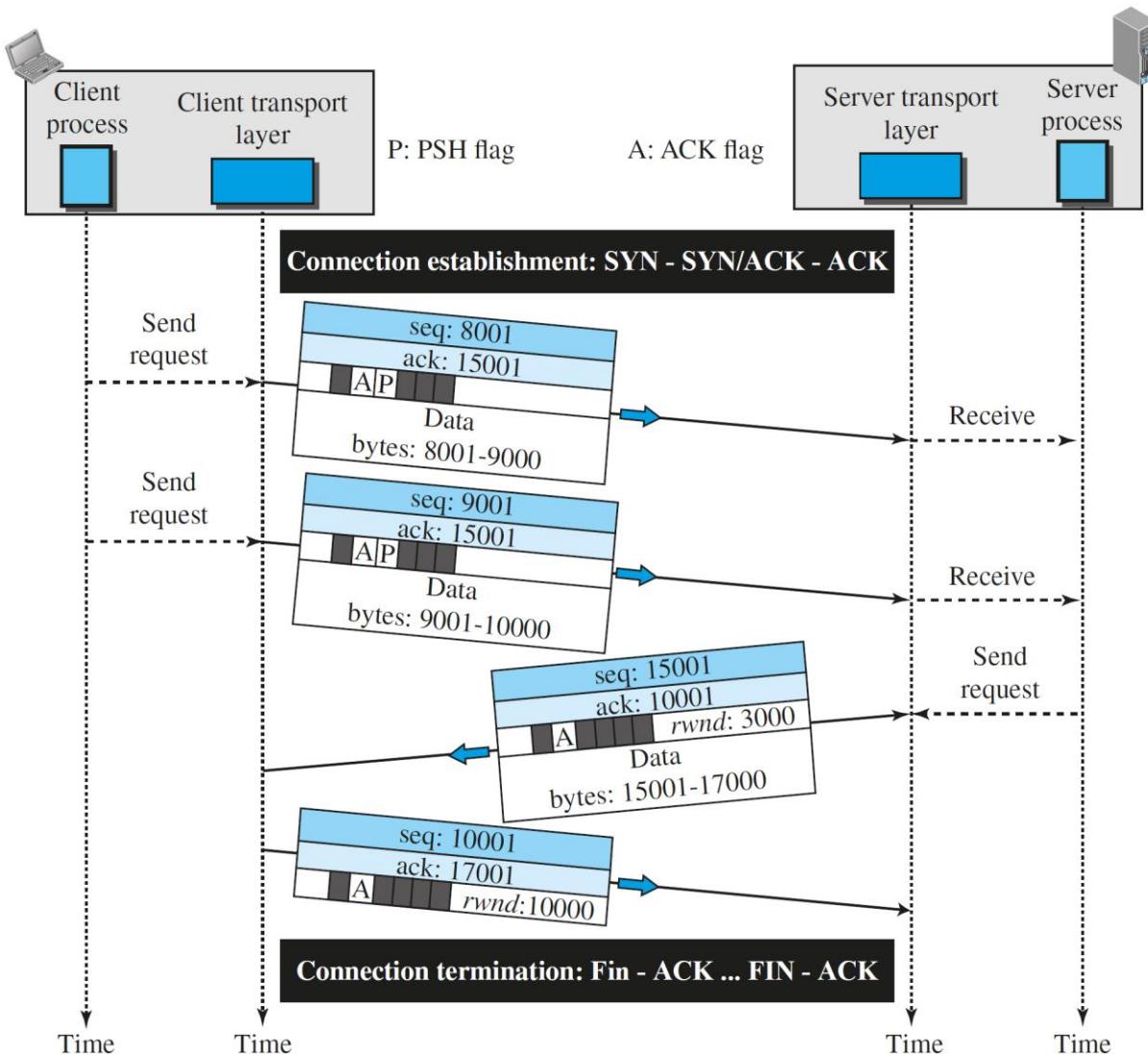
De client die dit ontvangt noteert 15000 als ontvangstteller voor zijn binnenkomend verkeer en stuurt een segment terug met enkel de ACK vlag gezet, en ACK nr. = 15001. Hij meldt ook aan de applicatie dat het opzetten van de connectie gelukt is: ‘Connection opened’ . (net zoals de server bij bericht 3).

Bij het uitwisselen van de SYN-segmenten kunnen parameters genegotieerd worden, bvb. een MSS (Maximum Segment Size, de maximale grootte van de te sturen segmenten) wordt opgegeven die door de andere zijde kan bevestigd (of verlaagd) worden.

Ook QoS (Quality of Service) en de window size worden overeengekomen. → Deze setup reserveert dan ook zend- en ontvangstbuffers en een ‘entry’ in de poort/socket-connectietabel.

### TCP - DATATRANSFER.

Van zodra de connectie 'staat' is ze steeds 'full duplex', d.w.z. er kan data stromen in **beide richtingen!**



Figuur 338. TCP data transfer

Eerst worden er hier bv. **2** segmenten van **1000** bytes gestuurd vanuit de cliënt: **seq.** of SN = **8001** en **9001**. SN = de plaats in de bytestroom van de **1<sup>e</sup> byte** uit het segment. Deze bytestroom is gestart op nr 8000 = ISN, de **1<sup>e</sup> byte** is 8001. (Het SYN segment uit de setup heeft ook 1 byte-nr. gekregen om het te kunnen ACKen.)

Deze eerste 2 berichten worden door de server tezamen ge'ACKnowledged' bij het verkeer in de andere richting: **ack of AN = 10 001**. AN = de plaats in de bytestroom van de **eerstvolgende byte** die verwacht wordt. We hebben reeds ontvangen t/m byte nr 10 000 → AN = de volgende byte... .

De server zelf zendt in hetzelfde segment ook **data**, startend van ISN+1 = **15 001**. Er zitten nu bv. 2000 byte in dit segment. (Dit is zeldzaam, max. ong. 1450 byte = het maximum dat bv. ethernet kan transporteren).

Dit segment van de server wordt op zijn beurt vervolgens ge'ACK' ed door het 4<sup>e</sup> segment: **ack = AN = 17001**.

OPM. De eerste 2 segmenten van de cliënt hebben ook de **P-vlag (PUSH)** gezet. Meestal zijn dit segmenten waar een interactieve vraag in zit: de cliënt klikt bv. een link aan van een bestand dat hij wenst te ontvangen. Die vraag moet zonder dralen bij de server aankomen → 'PUSH'. (**niet** eerst opslaan in een ontvangstbuffer, en wat later eens doorspelen aan de server applicatie). De server kan dan (snel) intern op zoek gaan naar de gewenste data en ... op zijn eigen tempo dit doorsturen → geen PUSH, dus wel de ontvangstbuffers passeren. Typisch zal bv. ook het laatste segment met de laatste data (van een bestand) worden ge'PUSH' ed.

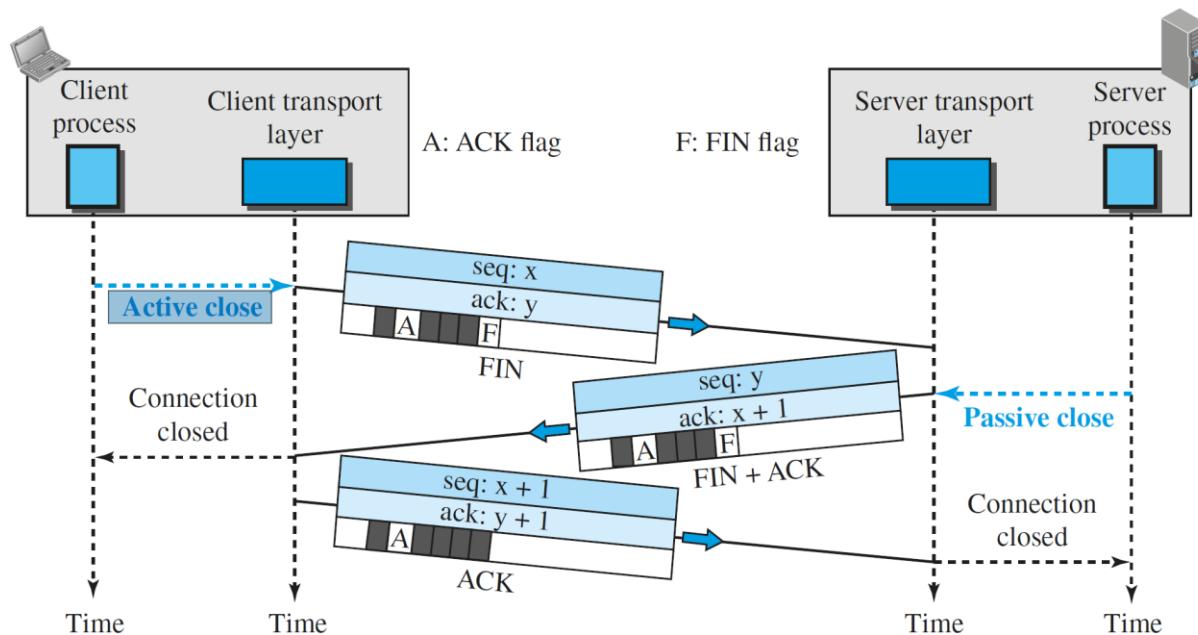
De **rwnd** parameter is belangrijk bij flow-control. De impact hiervan komt later.

Merk op dat **ALLE** segmenten op fouten worden gecontroleerd én geACKed.

### HET AFBREKEN VAN EEN CONNECTIE.

#### THREE-WAY HANDSHAKE

Als 1 van de 2 TCP stacks al zijn data heeft verzonden wil hij de **verbinding afbreken**. Hij verstuurt een segment met de **FIN vlag** gezet. De verbinding kan nu afgesloten worden als de server ook geen data meer te verzenden heeft → **FIN-ACK**. Dit moet alsnog bevestigd worden door de tegenpartij : **ACK**.



Figuur 339. TCP 3-way close.

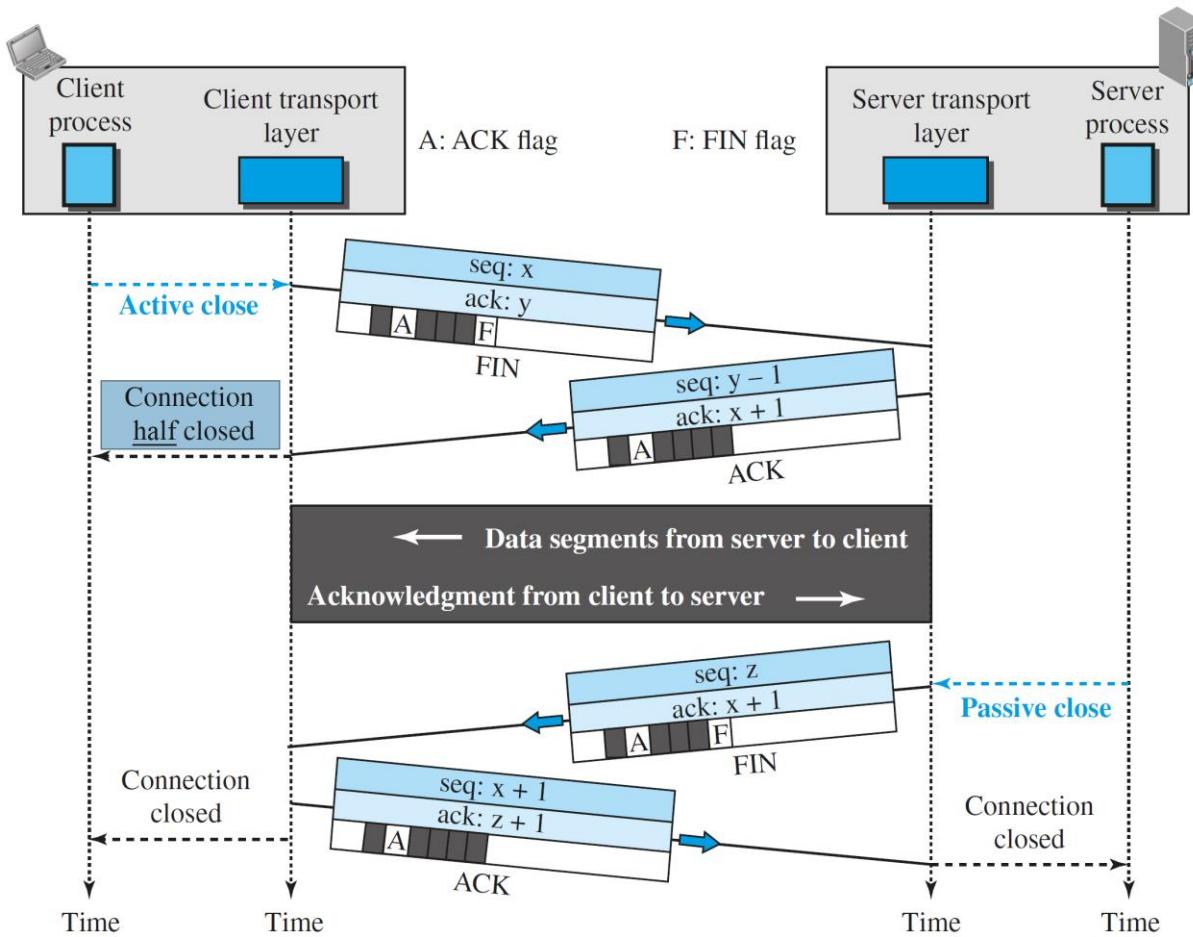
In dit voorbeeld bevatten de segmenten geen data meer, maar ze zullen wel nog 1 byte in de seq. of SN voorstellen omdat ze moeten ge'ACK' ed worden : seq = X ⇒ , ... ack = X+1 ⇌ en Y/Y+1.

Het is ook mogelijk dat het laatste datasegment in de stroom de FIN vlag zet. De werking is verder identiek.

## FOUR-WAY HANDSHAKE

Het is ook mogelijk dat 1 kant, bv. de cliënt, geen data meer te zenden heeft: hij heeft bv. aangeduid welk bestand hij wenst te ontvangen, en daar hoort verder geen uitleg meer bij. De server moet dit bestand ophalen van zijn extern geheugen, in stukken opdelen en 1 voor 1 doorsturen. Dat duurt nog wel even.

We krijgen dan (mogelijk) een ‘half open’ of ‘**half close**’ connectie.



Figuur 340. TCP, een half-open connectie.

De cliënt zal aangeven dat hij niets meer wil toevoegen aan zijn boodschap => **FIN**.

De server accepteert deze vraag → **ACK**.

Maar dat wil niet zeggen dat de TCP-stack van de cliënt deze connectie kan sluiten: hij moet nog steeds open staan om de data van de server te blijven ontvangen én dit bovendien nog ACK'en.

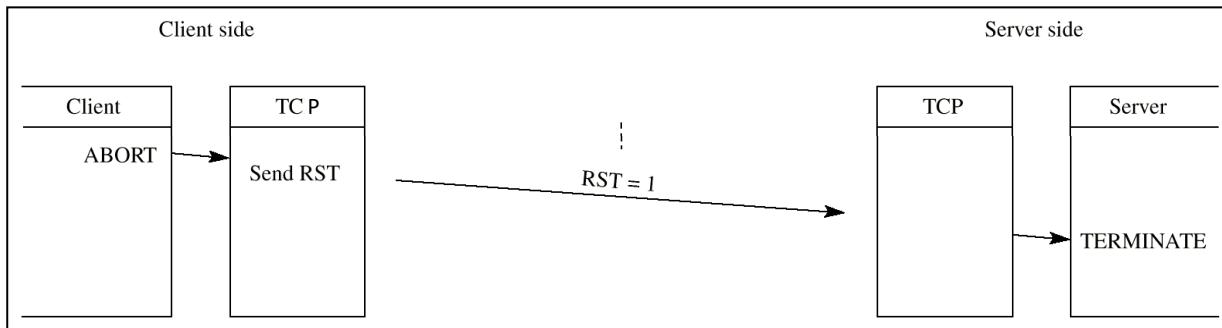
De cliënt \_TCP zal bij ontvangst van het laatste (FIN-)segment een definitieve ‘connection closed’ doorgeven aan zijn gebruiker, en een ACK op het FIN-segment terugsturen. Deze ACK, aangekomen in de server, genereert op zijn beurt een ‘connection closed’ en de verbinding is verbroken.

Beide voorgaande technieken, 3- en 4-way close, zijn ‘**gracefull ends**’ van een TCP connectie. Alle data is netjes doorgestuurd én ge ‘ACK’ed door de resp. ontvanger. Maar het kan ook mis gaan...

Als bv. seq.nrs. de mist ingaan kan ook een 'abort' sequence worden uitgevoerd. De cliënt zal **onmiddellijk** eenzijdig de connectie **afbreken** door een segment te sturen met de **RST**-vlag =1. Hij sluit alle poorten en buffers en zal geen verdere communicatie meer voeren op deze socket.

(Dit kan ook gebeuren door FWs FireWalls die een host beschermen, een half open connectie waarbij de andere kant is ge 'crashed', buffer overflows, ...).

De RST-vlag was bedoeld om alle resources terug vrij geven als er abnormale condities voorkomen, en zodoende de TCP verbinding te verbreken. (... maar komt de laatste jaren steeds meer voor als 'normale' beëindiging van de socket, zie ook lab.)



Figuur 341 Een abrupt TCP einde.

### 4.3.3 TCP PROTOCOL WERKING : UITGEBREID.

#### (HERHALING) Werking van TCP :

1. De data van de bovenliggende applicatie wordt in een buffer opgeslagen en **gebroken** in **segmenten** met een, volgens TCP, optimale grootte (... = MSS, Maximum Segment Size, wordt overeengekomen tussen de eindpunten bij het **opzetten** van de connectie). Dit is in tegenstelling tot UDP waar de applicatie exact 1 UDP datagram genereert.
2. Elk verzonden segment creëert een **timer** en moet ge'ACK'nowledged worden (→ duplex verbinding). Komt die ACK niet binnen de tijd, dan wordt het segment herzonden.
3. Als TCP zelf een segment ontvangt, moet hij een ACK terug zenden.
4. TCP controleert het segment door een **checksum** op **header én data**. Is deze foutief dan zal TCP geen ACK terugzenden, hij verwacht een timeout en retransmissie van de zender → bvb. 'go back N' retransmissie schema, zie HDLC.
5. TCP segmenten zitten verpakt in een IP datagram dat 'uit volgorde' kan aankomen → TCP moet de volgorde herstellen om aan de applicatie aan te bieden : '**sequencing**'.
6. TCP segmenten kunnen zich vermenigvuldigen : bvb. een segment is vertraagd, ge'time-out' en herzonden, de ontvanger krijgt ze beiden → TCP moet **dubbels** verwerpen.
7. TCP verzorgt **data-flow** : beide zijden hebben een beperkte buffercapaciteit. TCP verhindert dat een snelle host teveel data stuurt → '**sliding windows**' .

De timeout wordt ingesteld op de RTT, Round Trip Time, van de connectie : segment v. Z  $\Rightarrow$  O, ACK v. Z  $\Leftarrow$  O. Deze RTT wordt continue gevuld aangezien hij sterk kan variëren afh. v. de fluctuaties op het net. De timeout wordt gebaseerd op een gewogen gemiddelde van de RTT's.

#### Services van TCP

TCP biedt aan zijn bovenliggende applicaties een set van services aan zodat die een logische connectie met een TCP laag in een 'remote host' kan opzetten, data uitwisselen in full duplex mode, en de connectie kan afbreken. TCP garandeert dan dat dit foutvrij, in volgorde, en zonder dubbels gebeurt.

TCP verzendt zijn data in segmenten en beslist zelf wanneer er een wordt opgestuurd. Aan de ontvangende zijde worden de ontvangen segmenten in een buffer verzameld en afgeleverd aan de applicatie als de buffer vol is. Een segment kan dus bestaan uit **meerdere** korte 'user messages' of uit **een deel van één** groot bericht, bvb. een 'file transfert' → TCP biedt een '**byte stream**' service.

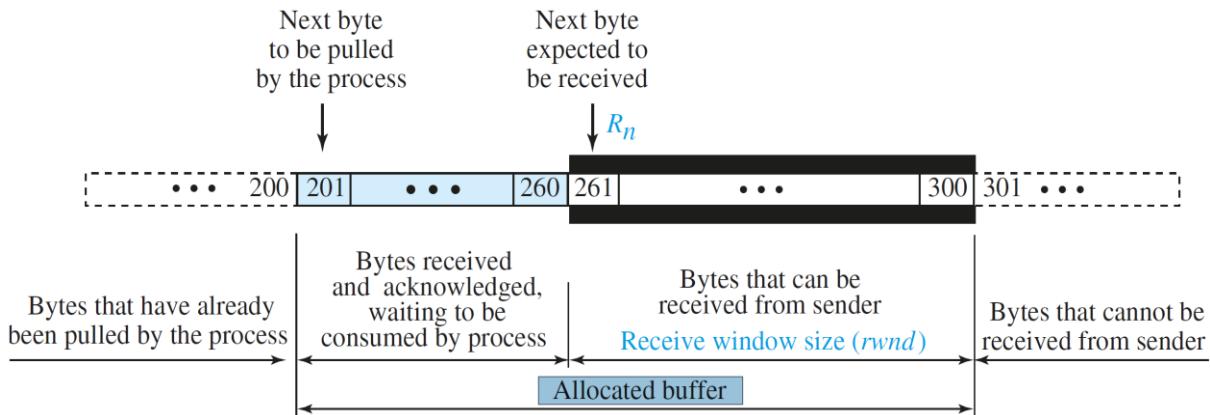
Op te merken valt dat er berichten kunnen geforceerd worden om **onmiddellijk** te worden afgeleverd (→ PSH-vlag) zonder te wachten tot de buffer vol is, bvb. een korte vraag voor data. TCP beslist zelf wanneer er voldoende data in zijn TX buffer zit om een segment te vormen. De TS\_user (appl.proces) kan nu TCP dwingen om alle wachtende data, incl. de data die ge'pushed' moet worden direct te verzenden. In de ontvangende zijde zal dit segment ook direct worden afgeleverd aan de TS\_user, met de aanduiding waar de dringende data zich bevindt.

Anderzijds kunnen berichten naast het gebruikelijke 'flow control' mechanisme voor normale data worden verstuurd als deze 'urgent' zijn → URG-vlag. De ontvanger is vrij hoe hij hierop zal reageren.

### DATATRANSFER IN DETAIL.

Het datatransport is gebaseerd op een sliding windows principe. TCP gebruikt hiervoor 2 windows (zend- en ontvangst-window) aan elke kant, dus voor een bidirectionele verbinding in totaal 4 windows. Ze worden kruiselings gesynchroniseerd met elkaar (op de segmenten in transit na).

Het is de ontvanger's ontvangst-window dat dicteert hoeveel er nog kan ontvangen worden /resp. gezonden worden door de tegenpartij → zend- window. We hebben deze parameter, *rwnd*, reeds opgemerkt in pt. TCP - Datatransfer. op p.320 e.v. .

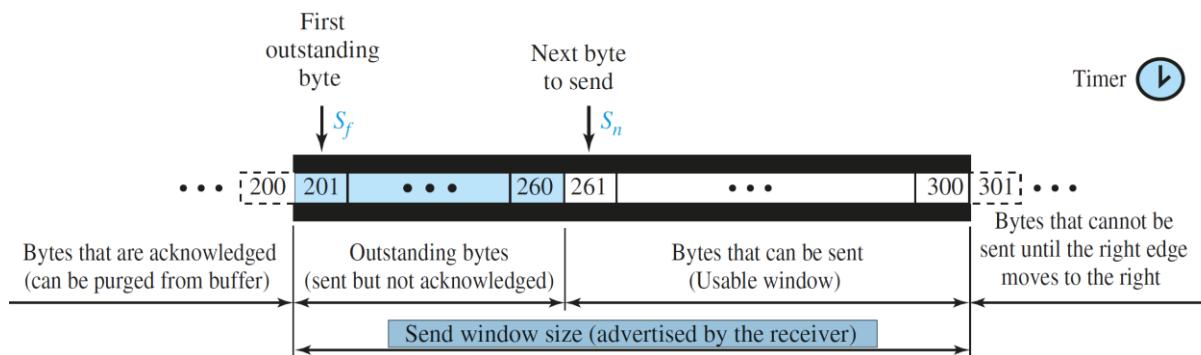


Figuur 342. TCP receive buffer en window.

Bij het opzetten van de connectie reserveert TCP een ontvangstbuffer, 'allocated buffer', in het geheugen. Daarin plaatst hij de ontvangen bytes ('bytestroom') terwijl die wachten om doorgestuurd te worden naar het applicatieproces. (*bytes to be pulled*).

De *rwnd* parameter geeft weer hoeveel **vrije ruimte** nog aanwezig is in de ontvangstbuffer. Dit wordt meegedeeld aan de tegenpartij door het '**window size**' veld in de TCP header, zie p. 315, tijdens het verkeer in de andere richting (rec → send) : ACK nr. = 261 (= volgende byte...) , *rwnd* = 40 (bytes) → er kan ontvangen worden van byte 261 tot 261 + 40 = byte 301!.

Op die manier kan de zender zich aanpassen aan de (snelheid van de)ontvanger: '**flow control**'.



Figuur 343. TCP send buffer en window.

De zender krijgt dus maar 'krediet' t/m byte 300 om te zenden. De vensters lijken dezelfde, maar **verschillen** wel degelijk! De bytes t/m 260 zijn reeds verzonden, maar nog niet ge 'ACK' ed door de ontvanger (vanuit het standpunt van de zender).

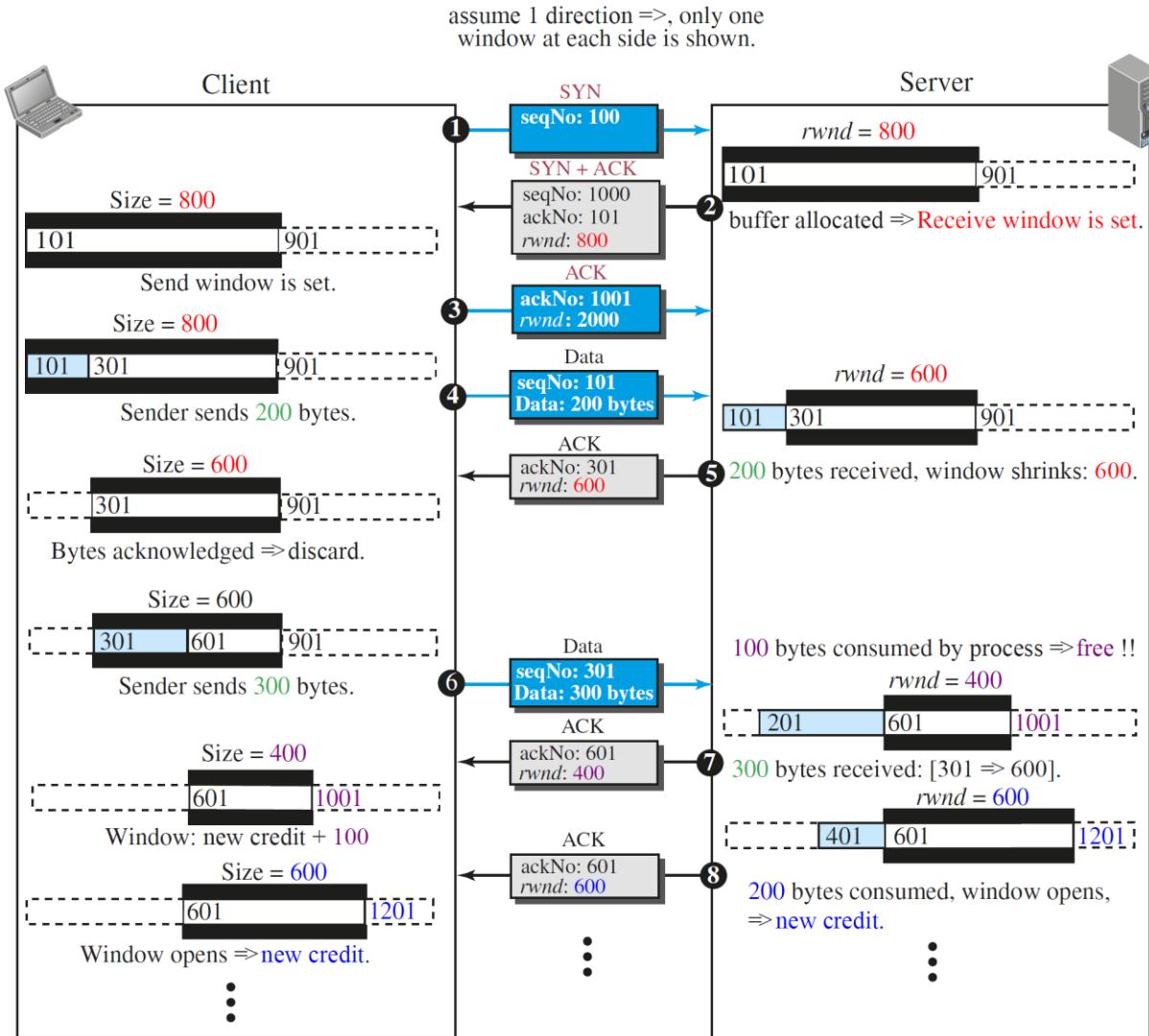
Dat betekent, vergeleken met de ontvanger, dat deze ACK's nog **onderweg** zijn!

De te verzonden bytes zijn wél gesynchroniseerd : t/m byte **260**: verzonden + ontvangen.

We zien ook de timer die ingesteld wordt bij de sender bij elk verzonden segment. Een ACK dient ontvangen te worden door de zender binnen deze tijd of het segment wordt herzonden (error control).

### TCP WINDOW WERKING

Laten we het verkeer voor de eenvoud in 1 richting bekijken, van cliënt → server.



Figuur 344. TCP window werking.

1. De verbinding wordt opgezet op initiatief van de cliënt : SYN. ISN = **100**. (1<sup>e</sup> bye = 101)
2. De server ontvangt deze aanvraag en reserveert een ontvangstbuffer van **800** byte. Hij deelt dit mee aan de cliënt via de **rwnd** parameter. + ...
  - a. Hij ACKt ook het 1<sup>e</sup> SYN segment. ACKnr = 101.
  - b. SYN: hij stelt een ISN voor, 1000, voor het verkeer in de andere richting, in deze figuur van geen belang.
3. ACK: de cliënt ACKt het vorige segment. Verder van geen belang.
4. De cliënt zendt een 1<sup>e</sup> segment met 200 bytes erin. Dit wordt ontvangen door de server waardoor die zijn window krimpt van 800 tot 600 bytes.
5. De server deelt dit mee aan de cliënt via de **rwnd**, in de ACK die erop volgt. +

De cliënt ziet dat de eerste 200 bytes goed zijn ontvangen (ACKnr = 301) en kan deze uit de retransmissiebuffer verwijderen.

6. De cliënt zendt nog eens 300 bytes. Die worden ontvangen door de server in zijn buffer: [301 – 600].

Er werden tussen 5 – 6: 100 bytes doorgestuurd naar het proces waardoor de ontvangstbuffer terug ruimte krijgt.

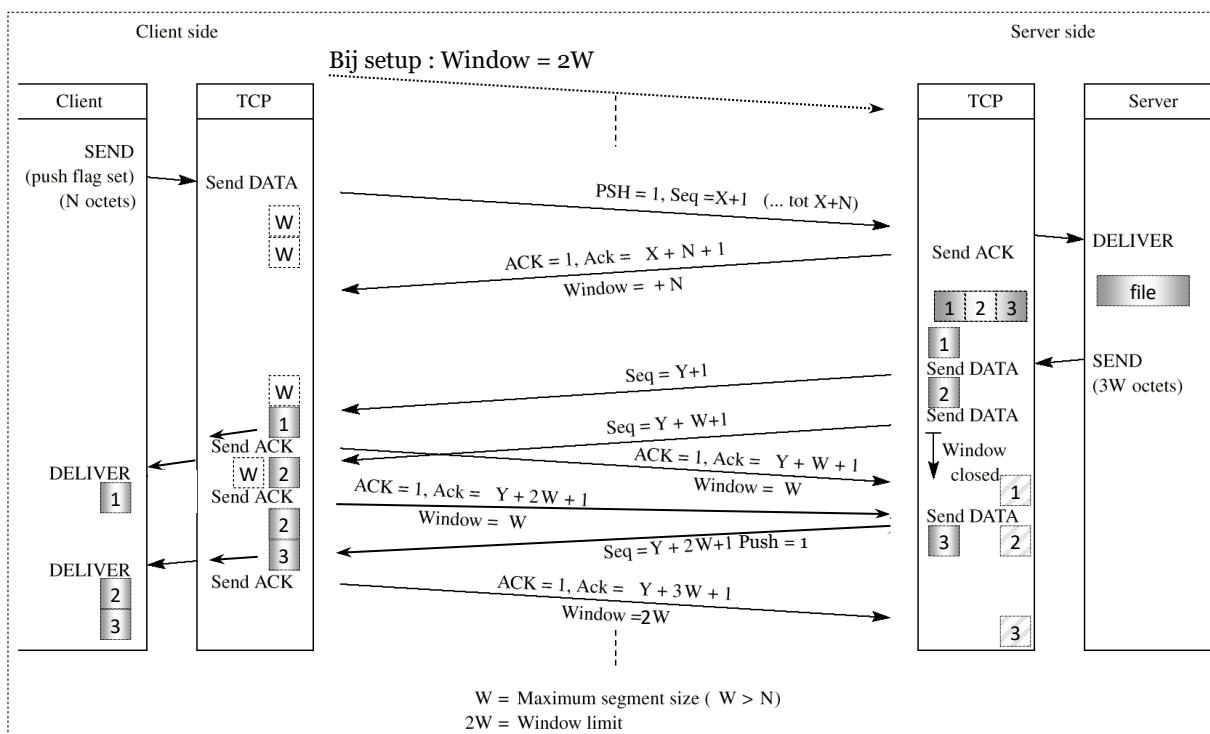
7. Bij de volgende ACK van de server
  - a. Bevestigt hij ontvangst tot byte 601 + ... meldt hij dat er bufferruimte is vrijgekomen : nieuw krediet tot byte 1001. Hij doet dit door de *rwnd* parameter = 400! →  $601 + 400 = 1001$ !
  - b. Dit komt aan in de cliënt die zijn venster uitbreidt van 901 tot 1001.
8. Wanneer nog eens 200 byte worden doorgestuurd naar het proces kan de bovengrens van beide buffers opnieuw worden vergroot...

### TCP WINDOW WERKING 2 : CLOSED WINDOW

Stel we volgen het transport van een FTP aanvraag van de cliënt . Deze aanvraag wordt ge'push'ed omwille van zijn interactief karakter, en is N bytes lang → SEND primitief aan de cliënt zijde. TCP verzendt een segment (van N bytes) met PSH=1 en seq.nr.=X+1, X is overeengekomen bij 'connection setup'.

Bij ontvangst merkt de (FTP-)server-TCP de PSH vlag → hij levert de aanvraag direct af, maar geeft eveneens een ACK terug met ACK.nr. = X+N+1 : de volgende byte die hij wenst te ontvangen. Aangezien de gereserveerde Rx-buffer niet is gebruikt, kan het venster N bytes verder worden geplaatst. Dit is in dit betoog niet zo belangrijk aangezien we focussen op het verkeer in de andere richting.

Stel dat de server een file aflevert die 3x de max. segment size W is. Zijn TCP segmenteert de data van het SEND primitief in 3 delen. Stel nu dat het cliënt -venster 2W groot is. Hij kan dan max. 2 segmenten afleveren maar moet vervolgens wachten op een ACK voor verder 'credit'.



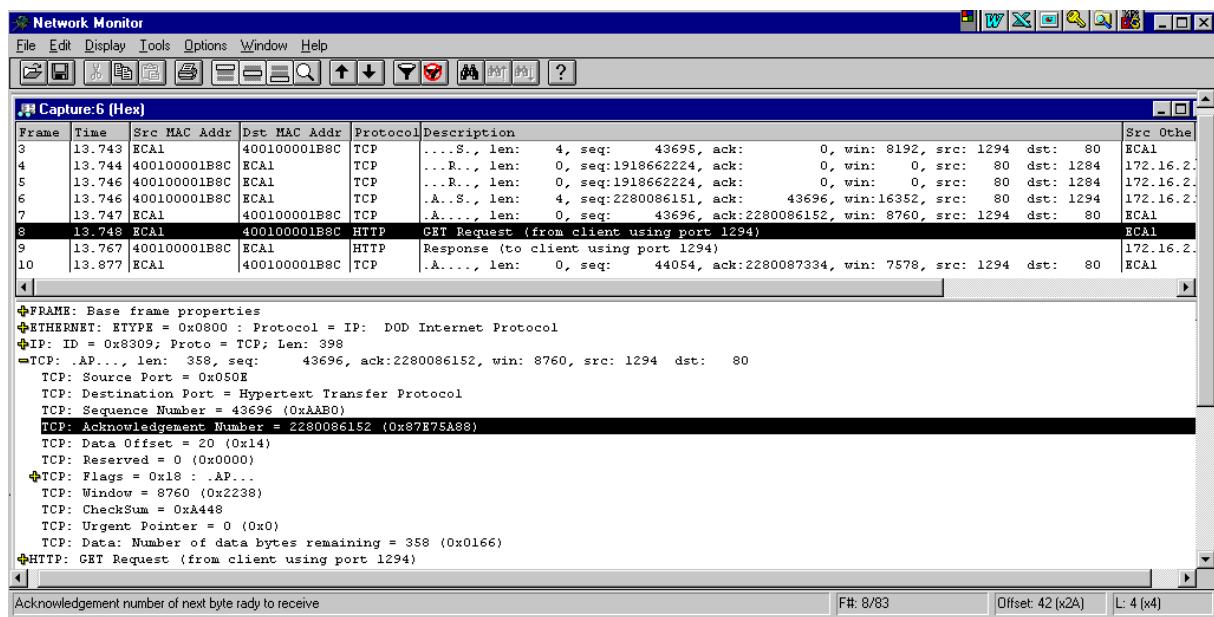
Figuur 345 TCP data transfer.

De cliënt -TCP zendt een ACK terug na elk segment (zou ook kunnen cumuleren, zie verder) :  $ACK=Y+W+1$  (= de volgende te verwachten byte) en  $ACK=Y+2W+1$ . Bij ontvangst van het 1<sup>e</sup> segment heeft hij nog de mogelijkheid om 1 segment te ontvangen → window = W. Hij levert vervolgens segment 1 af aan de applicatie.

Bij ontvangst van het 2<sup>e</sup> segment kan hij dan opnieuw krediet verlenen → window = W. De zender was ondertussen gestopt met zenden wegens geen krediet meer. Pas bij de ontvangst van deze 2<sup>e</sup> ACK ( $Y+2W+1$ , &W) kan hij het 3<sup>e</sup> segment leveren. Bij de cliënt is de buffer vol en hij levert de 2 ontvangen segmenten af aan de applicatie.

De server zal bovendien de segmenten waarvoor hij een 'acknowledge' heeft ontvangen uit de retransmissiebuffers verwijderen.

Onderstaande figuur laat, m.b.v. een sniffer, de praktische opeenvolging van seq.nrs zien. ECA1 is een machine vanuit het lab die een webpagina opvraagt aan een webserver ( $\rightarrow$  mac 400100001B8C).



Figuur 346 Een praktische TCP connectie opgezet.

1. frame 3 =**SYN** : ECA1 zet een connectie op met de http-server,  $\rightarrow$  dest. poort 80 en zelf gekozen (dynamische) antwoord-poort 1294. Hij stelt ISN (Initial Sequence Number) = 43695 voor.
  - frame 4 en 5 : deze server ziet nog een connectie met dezelfde cliënt op poort 1284 die hij reset.
2. frame 6 =**SYN/ACK**: de server 'ACK'ed de aanvraag van ECA1, op poort 1294, ACK-nr = 43696, en **SYN**  $\rightarrow$  Een TCP verbinding is full duplex  $\rightarrow$  verkeer in de andere richting : ISN 2280086151.
3. frame 7 =**ACK**: ECA1 ACK'ed deze SYN van frame 6 zonder data, de ACK-nr. = 2280086152.
4. frame 8 = 1<sup>e</sup> **DATA** frame: ECA1 stuurt een HTTP commando, GET, dat wordt verder bekijken :
  - SN = 43696, ACK nr. = 2280086152
  - lengte dan het segment = 358 bytes
5. frame 9 = 1<sup>e</sup> **DATA** frame in andere richting: de server antwoordt op deze GET ...(lengte = 1182 ??)
6. frame 10 = 2<sup>e</sup> **DATA** frame: ECA1 geeft een volgend data-segment, de SN = 43696 + 358 = 44054 en ACK'ed het antwoord van de server (frame 9): ACK nr. = 2280087334. ( $\rightarrow$  lengte ↗ ).

## FLOW CONTROL

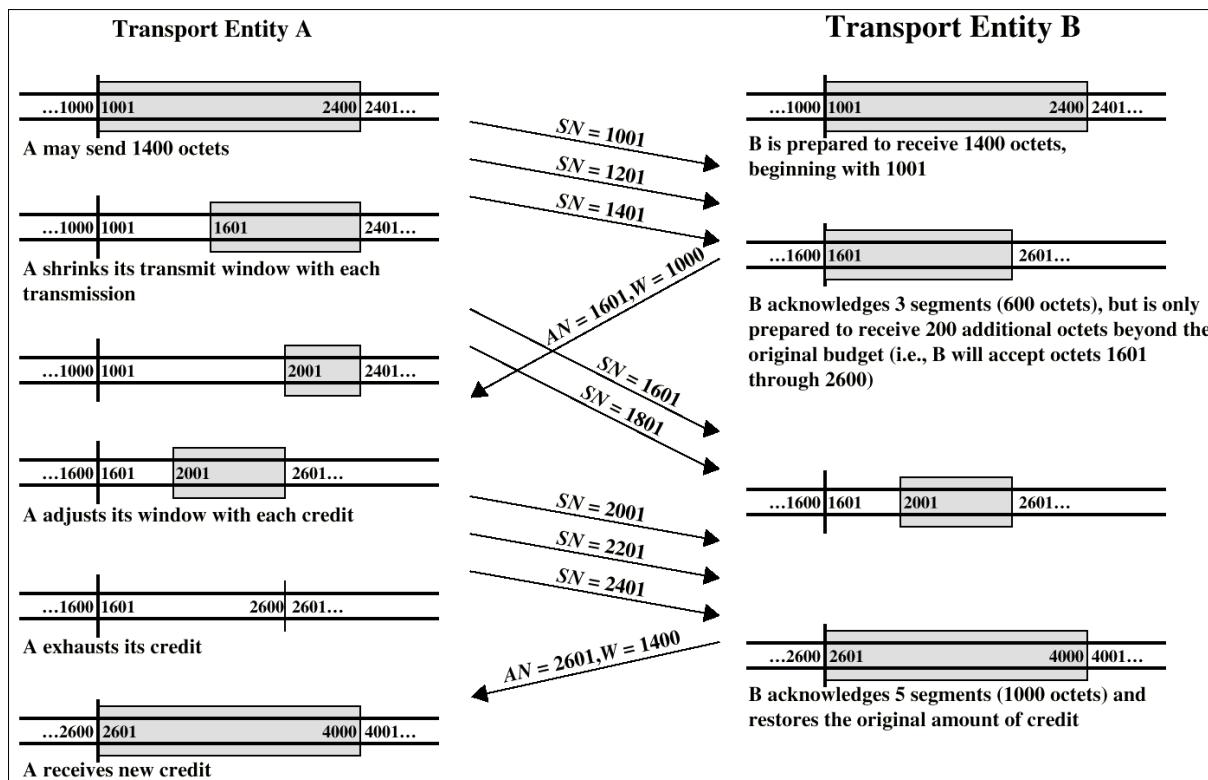
Het flow control mechanisme is iets complexer dan een L2 systeem, bv. HDLC, omwille van 2 redenen :

1. De ‘transmission delay’ is veel **langer** waardoor ‘flow-communicatie’ stroef verloopt.
2. Deze delay is bovendien sterk **variabel!**, wat ‘timeout’ tellers moeilijk instelbaar maakt.

Daarom wordt, i.t.t. een ‘*fixed sliding window*’ techniek zoals bij HDLC een ‘*credit*’ schema gebruikt. Hierbij worden **error-** (ACK) en **flow**-controle (window) van elkaar los gekoppeld om meer controle over de dataflow te krijgen en bvb. congestie te vermijden. Zo kan in een ‘*credit*’ schema een segment worden ge’ack’ed zonder nieuw krediet te verlenen en omgekeerd.

Hiervoor worden de 3 velden uit de TCP-header, seq.nr – ack.nr – window, gebruikt. Een segment dat correct aankomt geeft met seq.nr. SN de plaats aan van dit segment in de ‘*byte stream*’. Het wordt ge’ack’ed met (ack.nr. AN=  $i$  en window W=  $j$ ). Dit betekent :

- Alle bytes tot SN =  $i$ -1 zijn goed ontvangen, de volgende te verwachten byte heeft SN=  $i$ .
- Er wordt toegestaan om, ten opzichte hiervan ( $i$ ), nog  $j$  bytes te zenden → tot byte met SN=  $i + j - 1$ .



Figuur 347 Het credit-based flow controle schema van TCP.

Stel er is slechts verkeer in 1 richting en MSS = 200 bytes. De zend- en ontvangstnummers zijn gesynchroniseerd op 1000 bij de ‘setup’. A kreeg een venster van 1400 bytes, startend vanaf SN=1001.

Na 3 segmenten (=600 bytes) van A is het venster geslonken tot 800 bytes (van 1601 tot 2400). B besluit deze 600 byte te ‘ack’en en verleent een venster van 1000 byte. Dit betekent dat A nu kan zenden (van 1601) tot en met SN=2600, 200 byte verder dan voordien. Figuur 347 Het credit-based flow controle schema van TCP.

Maar tegen dat deze ACK van B aankomt in A heeft deze, onder de vorige voorwaarden, reeds 2 segmenten verzonden, m.a.w. SN=1601 tot 2000. M.a.w. A kon nog 400 bytes zenden, maar ziet zijn krediet nu stijgen tot 600 bytes (tot en met SN=2600).

Merk op dat dit mechanisme in beide richtingen kan werken en dat de ontvanger niet verplicht is om onmiddellijk een binnenkomend segment te 'ACK'en maar met een '**cumulatieve ACK**' een reeks segmenten kan bevestigen.

De ontvanger volgt een 'policy' in het verlenen van krediet. Bij een **conservatieve** benadering verleent hij krediet tot en met de grootte van zijn Rx buffer. Bij grote vertragingen kan dit de '*throughput*' ernstig verstoren. Hij kan dan een beetje bluffen en krediet verlenen voor ruimte die hij (nog) niet heeft als hij denkt dit te kunnen vrijmaken binnen de RTT (round trip time) → '*optimistic flow control*'.

#### 4.3.4 TCP CONGESTION CONTROLE

Congestie van een netwerk heeft 2 belangrijke gevolgen :

- eerst **stijgen** de **vertragingstijden** sterk omwille van het vollopen van de queue's (=uitgangsbuffers) van de tussenliggende routers.
- als de drukte niet afneemt zullen pakketten worden **weggeworpen** in de netwerknodes, omwille van het **overlopen** van de queue's van de tussenliggende routers.

Het flow control mechanisme van TCP/IP dient in eerste instantie om overstelping van de eindstations te voorkomen. Maar dit wordt nu ook ingeschakeld om congestie te vermijden : als de verhoogde **vertragingstijden** en het **verlies** van segmenten kan worden vastgesteld.

IP is een connectionless protocol, weinig service en control ... voorziet niet echt in congestion controle ... ook al is het eigenlijk zijn verantwoordelijkheid → verschoven naar TCP. (UDP trouwens ook niet.)

De aanpassingen die TCP uitvoert bij congestie kunnen we indelen in 2 categorieën : *retransmission timer management* en *window management*.

**Retransmission timer management** : het is voor de hand liggend dat een statische timer ofwel te lang of te kort is bij wijzigende netwerkcondities: congestie = stijgen van de RTT. Daarom zullen alle TCP implementaties de **momentele RTT** (Round Trip Time) zo nauwkeurig mogelijk berekenen en de timer iets groter in stellen. Deze berekening is te vinden in Stallings p. 636 waaruit blijkt dat vooral zal rekening gehouden worden met de laatste gemeten RTT's, dit om 'kort op de bal te spelen'. Een gelijkaardige berekening is te vinden in Forouzan p. 786.

**Window management** : de instelling van het 'send window' door TCP kunnen in belangrijke mate bijdragen om congestie te vermijden. We zien hiervan wel enkele technieken zoals *slow start*, *dynamic windows sizing*, *fast retransmit*....

## SLOW START

Hoe groter het zendvenster is dat TCP gebruikt, hoe meer segmenten er kunnen verstuurd worden alvorens er gewacht wordt op een ACK. Dit kan problemen opleveren als TCP te hevig van start gaat en segmenten injecteert tot het '*credit window*', aangekondigd door de receiver. Alhoewel dit OK is als beide hosts op dezelfde LAN communiceren, voor WAN verbindingen met verschillende intermediate routers, mogelijk over trage PPP links is dit een probleem → pakketten worden ge'dropped'.

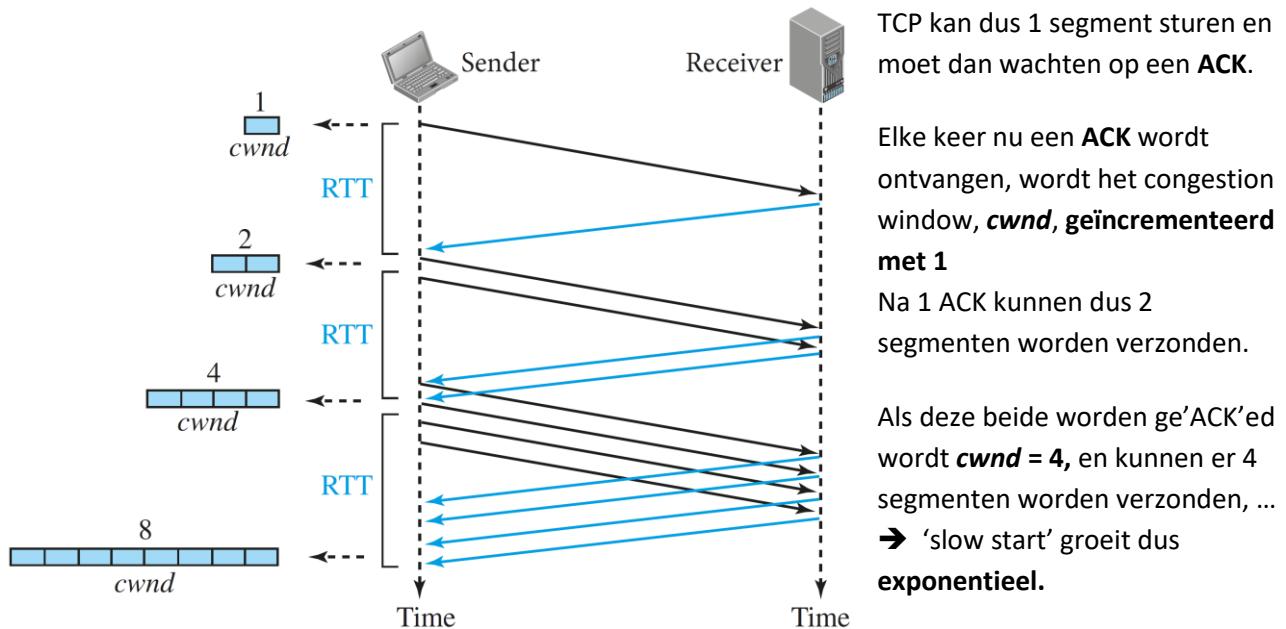
Jacobson stelt een procedure van '*slow start*' voor die ervan uitgaat dat de snelheid waarmee segmenten in het netwerk worden **geïnjecteerd** gelijk moet zijn aan de snelheid waarmee ze worden **ge'acked'**. Het maakt gebruik van een '*congestion window*', **cwnd**, uitgedrukt in **segmenten i.t.t. bytes** voor het '*credit window*'. Op elk ogenblik is het toegelaten venster, **awnd**, ('*allowed window*') :

$$\text{awnd} = \text{MIN} [\text{credit}, \text{cwnd}]$$

uitgedrukt in # segmenten

(Aangezien de ontvanger het '*credit window*' adverteert in **bytes** moet voor deze berekening de *credit*-waarde worden gedeeld door de *segment\_size*!)

Als nu een **nieuwe** connectie wordt geopend met een host op een ander netwerk, wordt : **cwnd = 1** gesteld.



Figuur 348 Slow start van TCP

(Het is misschien niet exact exponentieel omdat de ontvanger zijn ACK's mag uitstellen, typisch bvb. door 1 ACK te sturen per 2 ontvangen segmenten.)

Gaat dit door tot  $\infty$  ?? Uiteraard niet, zie bovenstaande formule → tot het '*credit window*' van de ontvanger is bereikt. (Of ... tot er een timeout optreedt! Zie verder.)

Daar waar '*credit window*' een flow control mechanisme is, opgelegd door de **ontvanger** om zijn buffers niet te laten overvloeden, is **cwnd** eigenlijk flow control opgelegd door de **zender** om congestie te vermijden.

## DYNAMIC WINDOWS SIZING ON CONGESTION → 'CONGESTION AVOIDANCE'

'Slow start' werkt goed om TCP snel een redelijk zendvenster te bezorgen voor een **nieuwe** connectie.

STEL. Op een bepaald moment, voor of na dat TCP het *credit window* van de receiver heeft bereikt, wordt een **segment verloren**. Hier zijn 2 indicaties voor :

1. er treedt een **timeout** op van het segment bij de zender,
2. er wordt een '**duplicate ACK**' ontvangen.<sup>1</sup>

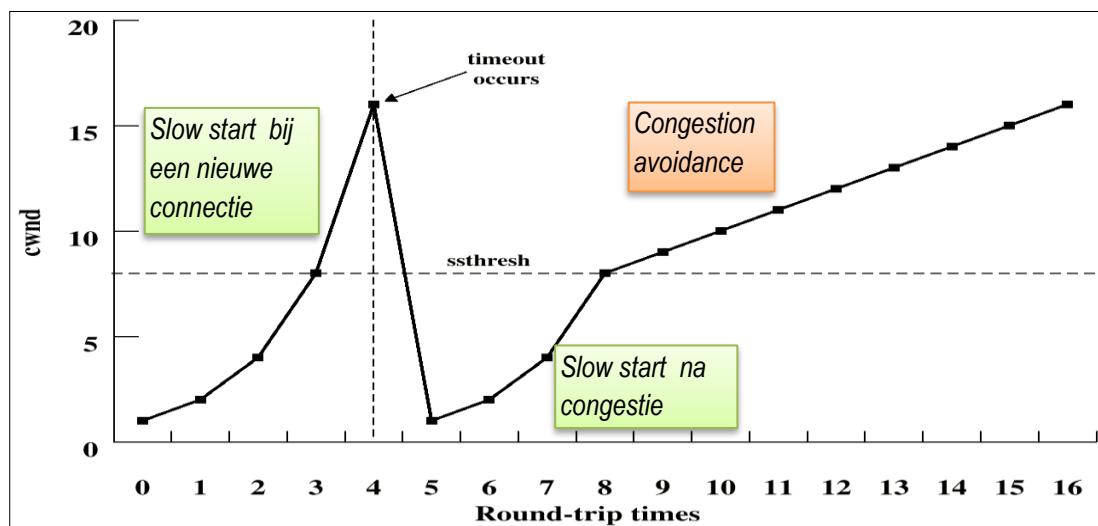
Er wordt verondersteld dat het verlies van segmenten door fouten in de checksum (→transmissie-fouten) zeer klein is (<1%). Dus : een verloren segment duidt op **congestie** → TCP moet zijn transmissie indijken.

Een **voorzichtige** benadering is dat *cwnd* terug =1 wordt en de **slow start** helemaal opnieuw begint. Alleen ... dit is **NIET VOORZICHTIG GENOEG** ! Jacobson poneert :

*" Het is gemakkelijk een netwerk in congestie te drijven, maar veel moeilijker om de congestie teniet te doen."*

→ de exponentiële groei van *cwnd* in **slow start** is **te agressief** : *cwnd* moet vanaf een bepaald moment **lineair** groeien! → een nieuwe variabele, *ssthresh*, = 'slow start threshold size' wordt ingevoerd. Het algoritme na congestie wordt dan :

1. zet *ssthresh* = *cwnd*/2, de helft van het *cwnd* op het moment dat congestie optreedt.
2. zet *cwnd* =1, doe terug een slow start, maar nu tot *cwnd* = *ssthresh*. (*cwnd* → +1 voor elke **ontvangen ACK**).
3. als *cwnd*  $\geq$  *ssthresh* → incrementeer *cwnd* (+1) voor **elke RTT, round trip time**.

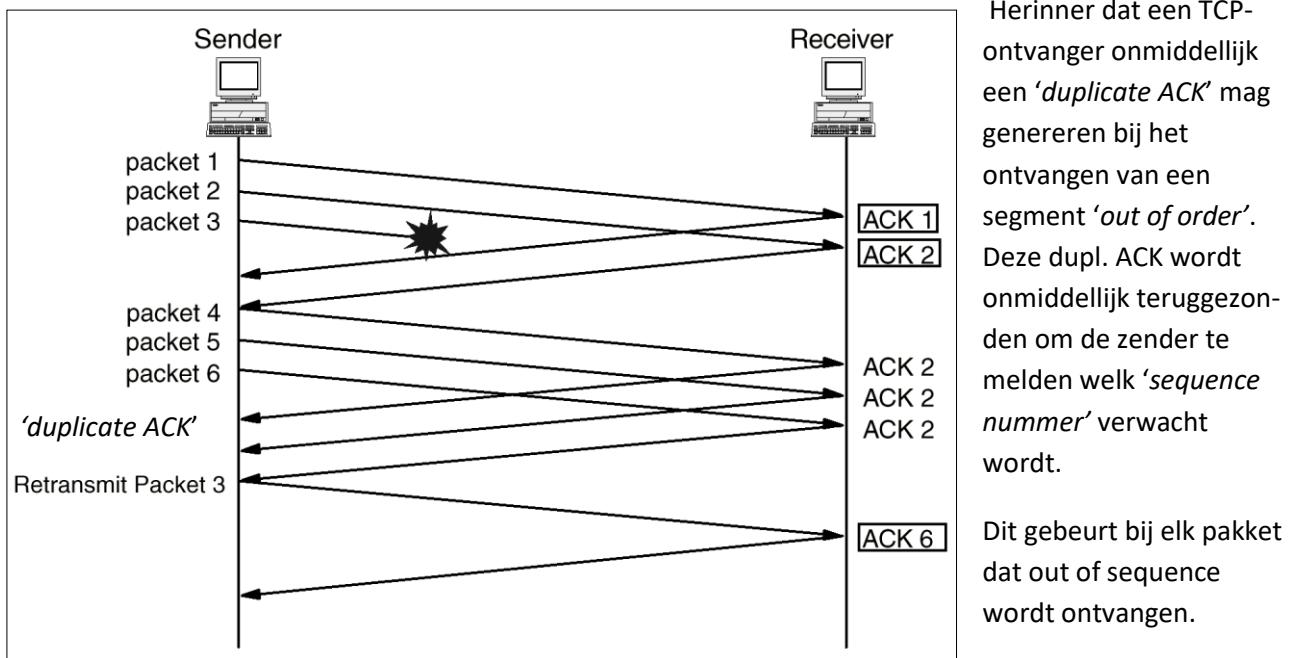


Figuur 349 TCP slow start en congestion avoidance.

<sup>1</sup> Een ontvanger krijgt een segment binnen dat niet in volgorde is → er is een segment weg of te laat → hij stuurt een ACK tot en met het laatste segment dat in volgorde was ontvangen, zie ook Fast retransmit op p.278.

## FAST RETRANSMIT & FAST RECOVERY

Fast retransmit verhindert dat TCP moet wachten op een timeout om een segment te herzenden.



Figuur 350 TCP fast retransmit.

Maar TCP<sub>s</sub> (sender) weet niet of het 'out of order' segment veroorzaakt is door een **verlies** van (packet) segment 3 dan wel door een **vertraging** ervan, bvb. omdat het een andere weg volgde. TCP<sub>s</sub> wacht daarom op een klein aantal 'duplicate ACK' om segment 3 te herzenden.

Typisch, bij een vertraging zullen enkel één of twee 'duplicate ACK's worden verstuurd alvorens segment 3 toch aankomt bij de ontvanger wat dan lijdt tot een cumulatieve ACK tot en met alle ontvangen segmenten.

Drie of meer 'duplicate ACK's na elkaar duidt erop dat segment 3 verloren is. TCP zal het nu **direct** herzenden **zonder te wachten** tot de retransmissietimer van het segment afloopt → **fast retransmit**.

## FAST RECOVERY

Na *fast retransmit* zal TCP terugschakelen naar 'congestion avoidance' i.p.v. 'slow start' omdat het optreden van meerdere 'duplicate ACK's erop wijst dat de ontvanger segmenten (4, 5 en 6) blijft ontvangen (die opgeslagen worden in de ontvangst buffer) en dat er dus nog altijd 'data flow' is. TCP wil dan niet de dataflow abrupt afbreken door een 'slow start'.

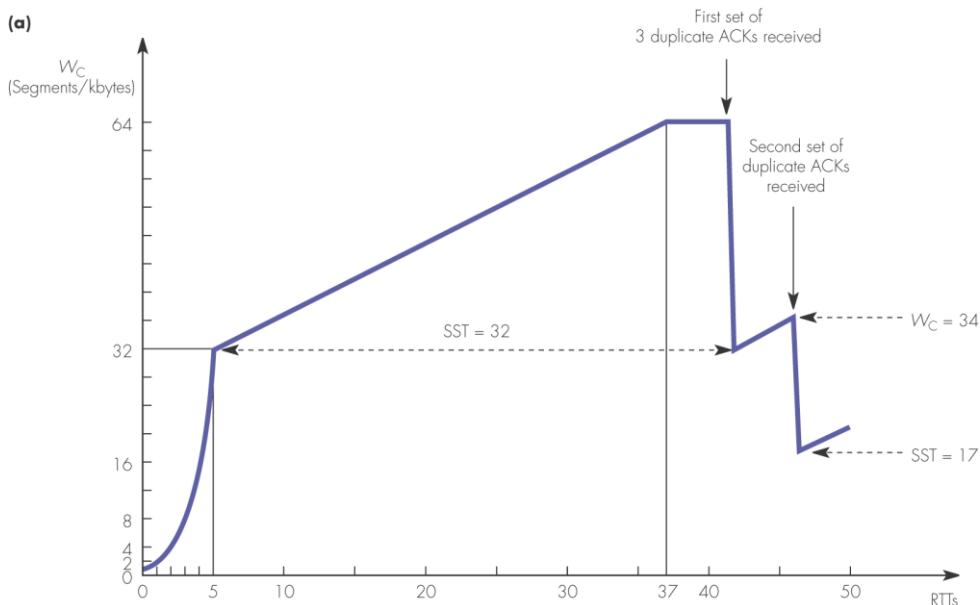
In volgende figuur is het cwnd (=W<sub>c</sub>) gestegen tot bv. 64 segmenten. Als de MSS bv. = 1024 bytes was afgesproken betekent dit ook 64Kbytes. Hij heeft daarmee bv. de max '**credit window**' bereikt van de verbinding (→ awnd = MIN [credit, cwnd]), m.a.w. ge 'gewone' **flow control**.

Bij ontvangst van 3 'duplicate ACK's schakelt het cwnd over naar SStresh = 32 en **congestion avoidance**.

Evt. treedt een beetje later dit fenomeen **opnieuw** op waardoor cwnd (=W<sub>c</sub>) opnieuw zal halveren.

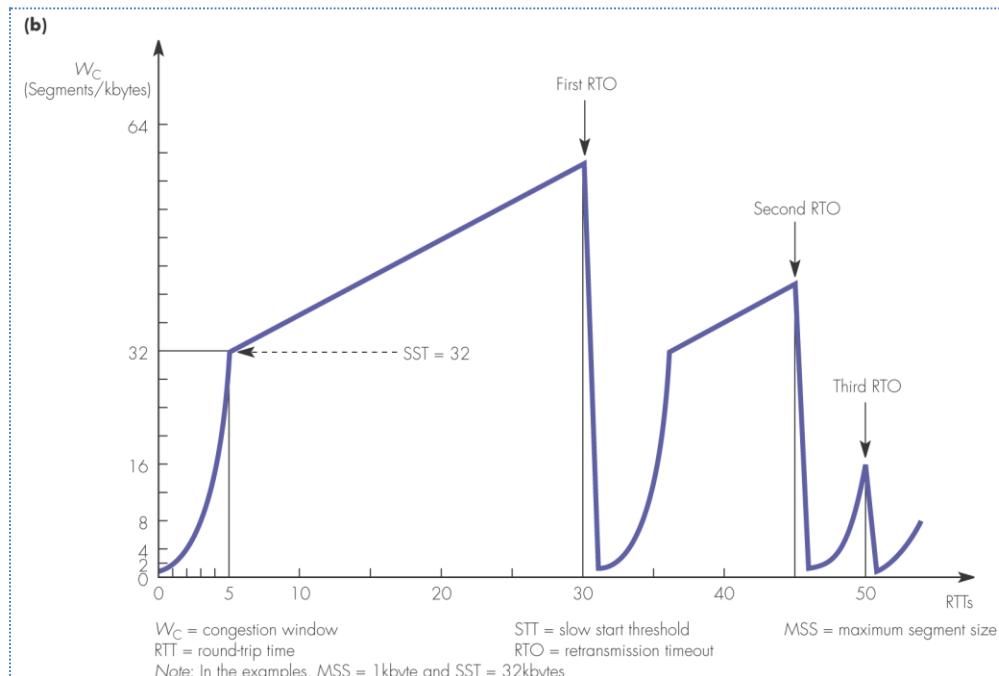
Herinner dat een TCP-ontvanger onmiddellijk een 'duplicate ACK' mag genereren bij het ontvangen van een segment 'out of order'. Deze dupl. ACK wordt onmiddellijk teruggezonden om de zender te melden welk 'sequence nummer' verwacht wordt.

Dit gebeurt bij elk pakket dat out of sequence wordt ontvangen.



Figuur 351 TCP congestion window aanpassing bij duplicate ACK's → Fast recovery.

Het gevolg van een *retransmission timer timeout* (RTO) is echter **drastischer** aangezien in dit geval er geen 'data flow' meer op het netwerk is. In principe is dit dezelfde werking als bij het ontvangen van een ECE vlag van de tegenpartij, zoals besproken bij Figuur 336. Explicit Congestion Notificatin vlaggen in TCP. Op p.316



Figuur 352 TCP congestion window aanpassing bij RTO's → slow start.

### ICMP SOURCE QUENCH

Een dergelijk ICMP-bericht is afkomstig van een router die congestie ondervindt (buffer overflow). Als dit binnenkomt zal TCP het behandelen als een RTO timeout en dus terug starten met *slow start* zoals te zien is in bovenstaande Figuur 352.

### 4.3.5 TCP AANPASSINGEN

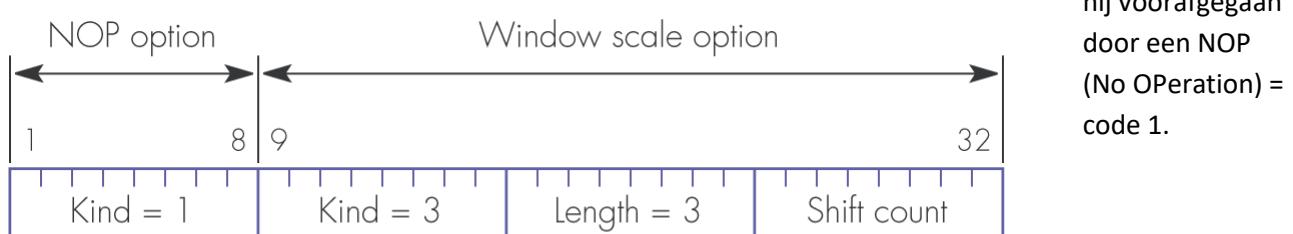
#### 'WINDOWS SCALE' OPTIE

Voor connecties met een groot '**bandwidth-delay**' product zal het window-mechanisme de belemmerende factor zijn in de **throughput** van de verbinding.

Bv. Voor intercontinentale verbindingen die een typische propagatie vertraging van 40ms kunnen hebben, liefst nog over fiber met hoge snelheden van bv. 155Mbps haalt men met het standaard window mechanisme feitelijk een throughput van 65535<sup>1</sup> bytes/40ms ... een ampele ... **1.6 Mbps !!!** of **1% effectief rendement**. Ook GBps verbindingen ondervinden deze restricties binnenin 1 LAN vanwege het niet ogenblikkelijk reageren van de TCP stacks, ACK's op ontvangen segmenten.

Gevolg is dat dit window mechanisme meer en meer DE beperkende factor wordt in TCP throughput. Ook al ligt het niet voor de hand dat 1 enkele TCP sessie een volledige link 'bezet' houdt, toch is het duidelijk dat deze window teller moet aangepast worden naar moderne normen, en liefst zonder de structuur van de TCP header aan te passen. Daarom is in RFC 1323 een **TCP-optie** gedefinieerd die een schaalfactor koppelt aan de window teller : de "**window scale option**".

Hij wordt, net zoals MSS, meegegeven bij het **opzetten van de verbinding** in het **SYN** segment. Hij kan verschillen in beide richtingen aangezien de 'geschaalde' window teller overeenkomt met de gereserveerde ontvangstbuffer. Deze optie is slechts 3 byte groot. Aangezien een TCP optie per definitie 32 bit is wordt



NOP = no operation option: used to pad the window scale option to 4

Figuur 353 TCP window 'scale option'.

De schaalfactor zit in het veld '**shift count**' en is **1 byte** groot. Hij mag zelfs maar waarden aannemen van **0 .. 14**. De echte window-tellers in de TCP stacks zijn **32 bit** variabelen. Ze worden nu berekend door de window size parameter uit de header (=16 bit) in deze variabelen te schrijven en vervolgens ze te 'schuiven' naar links, waardoor je ze vermenigvuldigt met  $2^{\text{shift count}}$ . Een shift count = 0 betekent GEEN vermenigvuldiging, = de oude definitie van de window pointers.

- Shift count = 0 → geen shift → max. window = 65 535 bytes
- Shift count = 1 →  $W \times 2^1$  → max. window = 131 070 bytes
- Shift count = 14 →  $W \times 2^{14}$  → max. window = 1 073 725 440 bytes

Hou er rekening mee dat deze MAX. window size moet overeenkomen met de grootte van de ontvangstbuffer en bijgevolg waarden tot 1Gbyte redelijk zeldzaam zijn... .

<sup>1</sup> De window parameter in de TCP header is 16 bit groot en standaard gedefinieerd als 'byte teller'.

### 4.3.6 TCP OP EEN ONBETROUWbare NETWERKSERVICE.

Tot op heden zijn we ervan uitgegaan dat de segmenten van TCP aankwamen en zelfs in volgorde aankwamen. Dat zou zo zijn voor een betrouwbare netwerkservice, wat IP **NIET** is. Deze beide problemen ('unreliability' en 'nonsequencing') creëren problemen bij de uiteengezette mechanismen.

We bespreken 7 problemen :

- Ordening van de segmenten
- Retransmissiestrategie
- Detectie van dubbels
- Flow controle
- Opzetten van de connectie
- Afbreken van de connectie
- Herstelling van een crash

#### ORDENING VAN DE SEGMENTEN

Bij een onbetrouwbare netwerkservice is het mogelijk dat segmenten 'out of order' aankomen. TCP gebruikt de sequence nummers om het ontvangen segment in volgorde te plaatsen. (i.t.t. bvb. een segmentnummer zoals bij HDLC.)

#### RETRANSMISSIONSTRATEGIE

2 oorzaken kunnen leiden tot retransmissie. Als bij aankomst van een segment een checksum-fout wordt gedetecteerd wordt het segment verworpen, of het is ook mogelijk dat een segment niet eens aankomt. In beide gevallen zal de ontvanger hiervan geen ACK terugsturen naar de zender. Deze 'timeout' het segment en herzendt het.

Dit heet een 'positive acknowledge' schema. De ontvanger MOET elk goed ontvangen segment bevestigen met het ACK.nr. (en -vlag). Om efficientie-redenen moet dit niet gebeuren pér segment, maar kan cumulatief worden bevestigd. Bvb. In Figuur 347 op p. 330 zal de eerste ACK 3 segmenten tegelijk bevestigen.

Vraag : op welke waarde moet de timeout teller van een segment ingesteld worden ?

→ 2 mogelijkheden : een **vaste** teller of een **variabele**, adaptieve teller.

Een **vaste** teller, ingesteld op de kenmerken van het netwerk, is ongevoelig voor veranderende netwerkcondities. Is hij te kort ingesteld zullen verschillende onnodige retransmissies optreden, is hij te lang dan wordt de reaktietijd te traag. Ideaal wordt hij ingesteld op een waarde iets groter dan de **RTT** (Round Trip Time : zend segment – ontvang ACK). Maar deze RTT is **variabel**, zelfs onder gelijke netwerkcondities.

Maar ook een **adaptief** schema heeft zijn problemen : stel dat de ‘retransmission timer’ wordt ingesteld op een waarde iets groter dan het **gemiddelde** van de gemeten vertraging, RTT.

- De ontvanger kan cumulatief ACK'en ! → RTT = ???
- Als een segment wordt herzonden weet de zender niet of de ACK die hij hiervan ontvangt een bevestiging is van het oorspronkelijke, dan wel het dupliaat-segment.
- De netwerkcondities kunnen heel plots veranderen.

Daarom zijn er een aantal timers voorzien in het TCP transport protocol :

Retransmission timer	Retransmit een verzonden segment dat niet ge'ACKnowledged' is
Reconnection timer	De minimum tijd tussen het sluiten van een connectie en het openen van een andere connectie met hetzelfde bestemmingsadres (en – poort).
Window timer	De maximum tijd tussen ACK mét CREDIT segmenten.
Retransmit SYN timer	De tijd tussen opeenvolgende pogingen om een connectie op te zetten.
Persistence timer	Hoe lang duurt het alvorens een connectie wordt verbroken als er geen segmenten worden ge'acknowledged'.
Inactivity timer	Verbreek de verbinding als er geen segmenten worden ontvangen.

### DETECTIE VAN DUBBELS

Een verloren segment resulteert in een herzenden van een dupliaat zonder veel problemen. Als er echter een **ACK** verdwijnt zullen 1 of meerdere (→ bvb. bij cumulatieve ACK) segmenten worden herzonden en ontstaan er dubbels. De ontvanger moet deze herkennen op basis van hun sequence nummer.

We onderscheiden 2 gevallen :

1. Een dubbel wordt ontvangen **vóór** het sluiten van de verbinding.
2. Een dubbel wordt ontvangen **ná** het sluiten van de verbinding.

→ 1. De ontvanger moet veronderstellen dat zijn ACK verloren is gegaan en moet het dupliaat ACK'en. De zender mag niet in de war raken bij het meervoudig ontvangen van ACK's op eenzelfde segment. De sequence nummers moeten ver genoeg uit elkaar liggen om te verhinderen dat de modulo-telling ‘rondgedraaid’ is binnen de maximum leeftijd van een segment → 32 bit.

→ 2. Zie verder bij : Opzetten van de connectie op p. 340

## FLOW CONTROLE

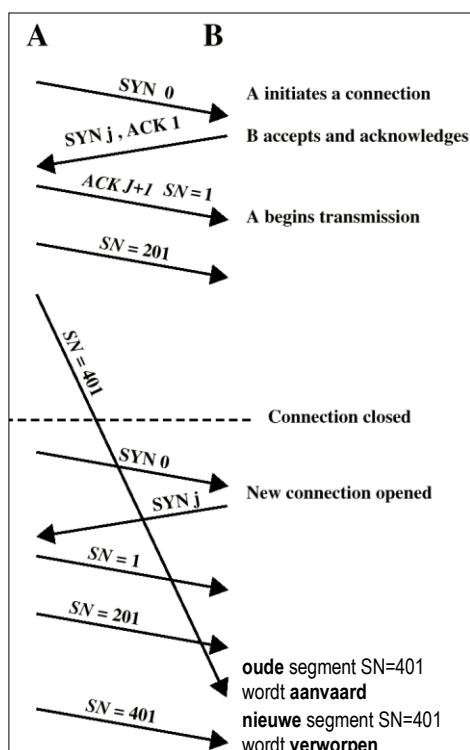
Als een ACK/CREDIT segment is verloren gegaan, zullen volgende ACK's het protocol terug synchroniseren. Als deze niet komen zal de zender zijn retransmissietimer aflopen en een datasegment herzenden, wat aanleiding geeft tot een nieuwe ACK. Er kan echter steeds '**deadlock**' optreden :

Stel B (=ontvanger) zendt een ACK met AN=  $i$  en W=0, om de ontvangst van databytes tot  $i-1$  te bevestigen maar tijdelijk het venster te **sluiten** wegens 'geen buffers beschikbaar'. Na een tijdje trekken de wolken voor B op en hij zendt een ACK met AN=  $i$  en W=  $j$ . Maar dit ACK-segment gaat **verloren**... . A (=zender) wacht om opnieuw te zenden en B denkt dat hij dit reeds heeft toegezegd → '**deadlock**'.

Hiervoor is er een '**window** timer' voorzien. Hij wordt ge'reset' bij elk (ACK-) segment dat een ACK.nr en window-veld bevat. Loopt hij af, dan moet het laatste herzonden worden, ook al dupliceert het een vorige ACK. Dit breekt de deadlock. Dit is niet te vergelijken met een retransmissietimer op datasegmenten (maar dan op ACK-segmenten) aangezien een retransmissietimer voor **ELK** datasegment wordt gestart, de windowtimer herstart bij elk ACK-segment mét 'credit-window'.

## OPZETTEN VAN DE CONNECTIE

Bij de 3-way handshake voor het opzetten van een verbinding, bvb. van A → B, zent A een SYN<sub>1</sub> naar B en verwacht een SYN<sub>2</sub> terug, zie ook Figuur 337 op p.318. Beide SYN's kunnen nu verloren gaan en worden uitgeteld door de retransmit-SYN teller. (moet > dan SYN<sub>1</sub> ↔ SYN<sub>2</sub> en van SYN<sub>2</sub> tot ACK van A).



Maar door het herzenden van SYN's ... ontstaan mogelijk SYN-dubbels! → A en B moeten deze dubbels herkennen en verwerpen eens de connectie staat. Daarom worden ook bij de SYN-segmenten een SEQuence nr. (SN) meegegeven.

In naaststaande figuur veronderstellen we dat seq.nrs. steeds op 0 beginnen na setup → als eerste databyte SN = 1. (voor de eenvoud zijn de ACK's weggelaten in deel 2 en de MSS = 200). We zien nu dat 'verloren' datasegmenten van een vorige verbinding de huidige reeks versturen.

Om dit te vermijden worden SN nummers bij setup ver uit elkaar gekozen en is er een reconnection timer.

Figuur 354 Het probleem van 'verloren' datasegmenten.

### AFBREKEN VAN DE CONNECTIE

Om dezelfde reden wordt bij FIN segmenten eveneens sequence nummers meegegeven. Stel dat A, die reeds in de ‘CLOSE WAIT state’ staat, zijn laatste datasegment opstuurt, gevolgd door een FIN segment. Maar ... het FIN segment komt aan **vóór** het laatste datasegment, B **sluit** de verbinding (want hij heeft reeds een FIN segment gegeven, zie Figuur 339 op p. 321) en verliest het laatste datasegment !

Als het FIN-segment nu een seq.nr. bevat kan B nog wachten op het ontbrekende datasegment alvorens te sluiten. Dit laatste datasegment én het FIN segment worden nog ge’ACK’ed door B waardoor A ook definitief kan sluiten.

### HERSTELLING VAN EEN CRASH.

Als een systeem (A) waarop TCP draait wordt ge’reset’ (power off) en terug opstart is hij alle tellers van al zijn actieve connecties kwijt. De verbindingen worden dan ‘**half open**’ omdat de andere kant (B) (nog) niet op de hoogte is van de crash. Deze andere kant B kan de verbinding afbreken wegens het aflopen van de ‘**persistence timer**’. Deze timer meet de tijd dat TCP wacht op een ACK van een opgezonden segment nadat dit segment het maximum aantal keren is herzonden.

Loop deze teller af dan wordt er verondersteld dat kant A is ge’crashed’ en TCP\_B sluit de verbinding en meldt zijn gebruiker de abnormale ‘CLOSE’.

Als systeem A nu faalt en snel terug heropstart, neemt A een ISN verschillend van de vorige seq.nrs. waar B nog op wacht → alle half-open verbindingen bij B zullen worden afgebroken door een RST *i* (van A) te geven op elk segment *i* dat A ontvangt van B. Als dit RST *i* segment terug aankomt bij B wordt de SN = *i* gecontroleerd, want het zou een ‘verloren’ RST segment kunnen zijn van een vroegere verbinding...

### 4.3.7 TCP IMPLEMENTATIE POLICY'S.

Sommige aspecten van het protocol laten een keuze toe, een benaderingswijze, hoe ze worden geïmplementeerd → ‘policy’s. Alhoewel 2 implementaties (met verschillende opties) perfect met elkaar moeten communiceren zijn er mogelijk belangrijke performantie-verschillen.

#### ZEND- EN ONTVANGST POLICY

In normale werking (geen ‘pushed’ data) is TCP vrij om de data, doorgekregen van zijn TS\_user, te bufferen in een Tx buffer en te verzenden wanneer hij dat opportuun vindt. Hij kan een segment maken voor elke brok data die hij ontvangt, of wachten tot er voldoende data in de buffer zit. Voor de ontvangende zijde is dit identiek. Het is een evenwicht tussen enerzijds het beperken van overhead (interrupts) naar de processor en het operating system toe en anderzijds de gewenste responssnelheid.

#### ACCEPT POLICY

Als datasegmenten uit volgorde aankomen heeft TCP 2 mogelijke opties :

1. **In-volgorde** : TCP accepteert alleen segmenten die in volgorde aankomen en verwerpt alle ‘out of order’ segmenten.
2. **In-budgetvenster** : TCP accepteert alle segmenten die in het ontvangstvenster, = het ‘window’, aankomen.

Ook hier is het afwegen of het netwerk gaat belast worden met het herzenden van reeds succesvol opgestuurde segmenten, dan wel de μP en OS met het in volgorde plaatsen van de ontvangen segmenten.

#### RETRANSMIT POLICY

TCP houdt een rij (‘queue’) van verzonden segmenten bij die nog niet zijn bevestigd. Hij moet ze kunnen herzenden als hij geen ACK heeft ontvangen binnen een tijd ingesteld door de ‘retransmission timer’. Hij kan nu 3 retransmissie-strategieën volgen :

1. **‘First only’** : Er wordt **1** retransmissie-timer bijgehouden voor de **volledige** queue. Wordt er een ACK ontvangen, dan worden de resp. segmenten verwijderd en de timer gereset. Verloopt de timer dan zal het segment **vooraan** in de queue worden herzonden en de timer gereset.
2. **‘Batch’** : Er wordt ook **1** retransmissie-timer bijgehouden voor de **volledige** queue. Wordt er een ACK ontvangen, dan worden de resp. segmenten verwijderd en de timer gereset. Verloopt de timer dan zullen **alle** segmenten in de queue worden herzonden en de timer gereset.
3. **‘Individual’** : Er wordt **ook 1** retransmissie-timer bijgehouden pér segment! Wordt er een ACK ontvangen, dan worden de resp. segmenten én de corresponderende timers verwijderd. Verloopt één van de timers dan zal het (=1!) corresponderende segment worden herzonden en zijn timer gereset.

De ‘first-only’ strategie is efficiënt voor het netwerk aangezien enkel de verloren segmenten (of de segmenten waarvan de ACK verloren is) worden herzonden. Het kan echter tot lange vertragingen leiden aangezien de timer voor het volgende segment pas wordt gestart als het vorige is ge’ack’ed.

De ‘individual’ strategie lost dit probleem op, ten koste van een complexere implementatie. De ‘batch’ strategie is minder efficiënt voor het netwerk, maar heeft geen last van lange vertragingen. Deze laatste, ‘batch’, is vooral interessant als de ontvanger een in-volgorde strategie volgt, aangezien hij de volgende segmenten toch heeft verworpen. In-budgetvenster ‘accept policy’s accorderen beter met ‘first-only’ of ‘individual’ retransmit policy’s.

#### ACKNOWLEDGE POLICY

Een datasegment dat ‘in sequence’ wordt ontvangen kan bevestigd worden volgens 2 strategieën :

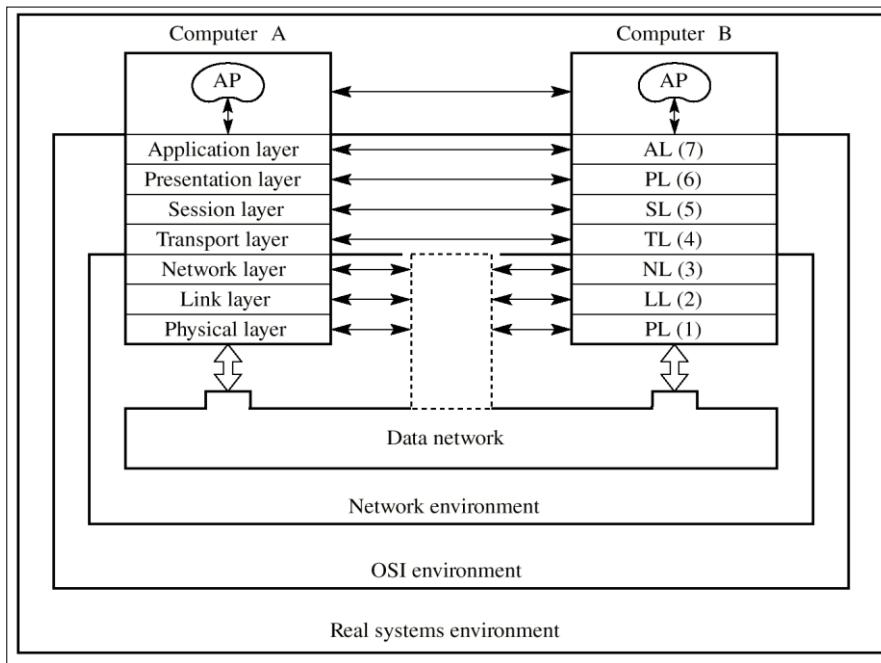
1. **‘Onmiddellijk’** : er wordt onmiddellijk een ACK segment verzonden, zonder data, met het overeenkomend ACK-nummer.
2. **‘Cumulatief’** : er wordt ‘genoteerd’ dat er een ACK moet teruggestuurd worden, maar dit gebeurt pas als er data de andere kant wordt opgestuurd → ‘piggy backing’. Om lange wachttijden te vermijden wordt een ‘window timer’ ingesteld die toch een leeg ACK segment laat verzenden als hij verloopt.

De eerste filosofie is simpel maar belastend voor het netwerk, waardoor vooral cumulatief wordt ge’ack’ed. Dit maakt het echter complexer om de RTT te schatten.

# 5 APPENDICES

## 5.1 OSI PROTOCOLS

### OSI model



Hier voor verwijzen we naar de inleiding van de cursus, p 28, waar we de indeling en functie bespreken van de 7 OSI lagen.

Figuur A-1 Het OSI model

### OSI begrippen : SAP 's

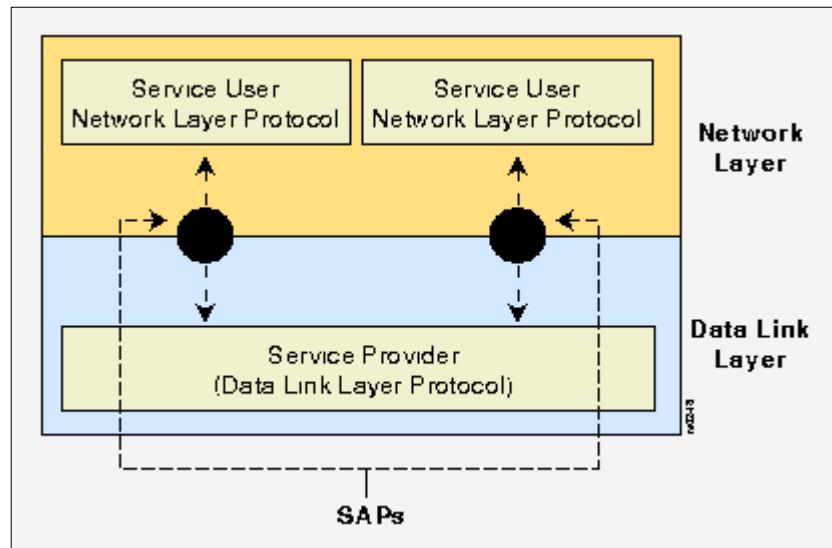
Een laag verleent services aan de bovenliggende laag, bepaald door het communicatieprotocol dat in een P2P verbinding staat met de remote computer.

De service die laag N biedt aan laag N+1 wordt bepaald door het **protocol** in laag **N** en de **services** aangeboden door laag **N-1**. De services worden aangeboden via een **SAP** : Service Access Point.

**SAP** : In de lagere OSIlagen dienen SAP's om de **fysische locatie** van een computer in het netwerk te identificeren.

Bvb.

- laag 2 : 6-byte MAC-adres
- laag 3 : 4-byte IP adres



Figuur A- 2 Service Access Points.

Aangezien in 1 computer ook **meerdere** applicatie-processen kunnen draaien, moet een volledig adres van een datapakket niet alleen het **fysisch** adres bevatten van de **computer** maar ook **adresinformatie** voor het verder sturen van het pakket **binnenin** de computer, de **NLPID** (Next Layer Protocol Identifier).

Bvb.

- laag 2 : het **type** veld van ethernet of (8byte verder), van ethernet\_SNAP = 0800 voor IP ,
- laag 3 : de **protocol** byte in de header = 06 voor TCP,
- laag 4 : 2-byte **portadressen** = 25 voor SMTP

Zo zal een volledig adres van een applicatie bestaan uit de som van een aantal subadressen, SAP's, ook nog interlayer address selectors genoemd.

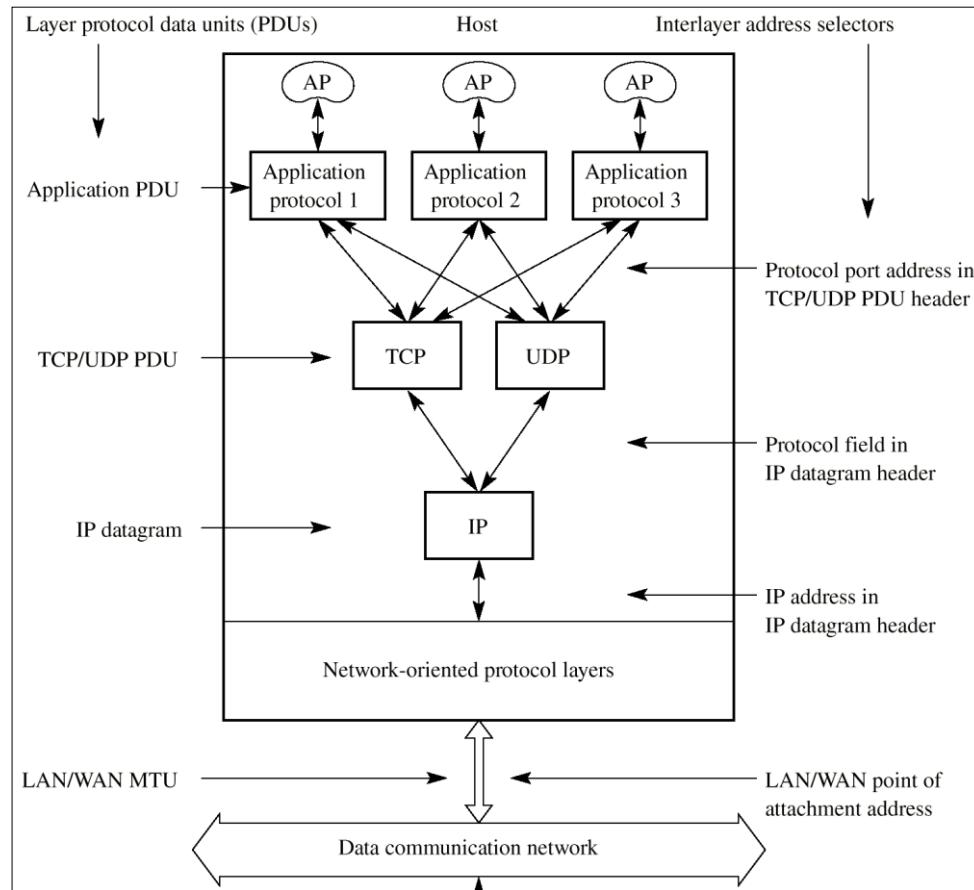
$$\text{AP adres} = \text{PSAP} + \text{SSAP} + \text{TSAP} + \text{NSAP}$$

Waar **PSAP** het SAP is tussen applicatie- en Presentatie-laag, **SSAP** tussen presentatie- & Sessielag, ... . NSAP is het wereldwijde netwerkadres. M.a.w. meerdere applicaties (processen) leiden tot meerdere SAP's, elk geïdentificeerd door een uniek adres.

Praktisch voorbeeld van SAP's : de **port** adressen (**TSAP**) van TCP en de **IP** adressen (**NSAP**) van IP.

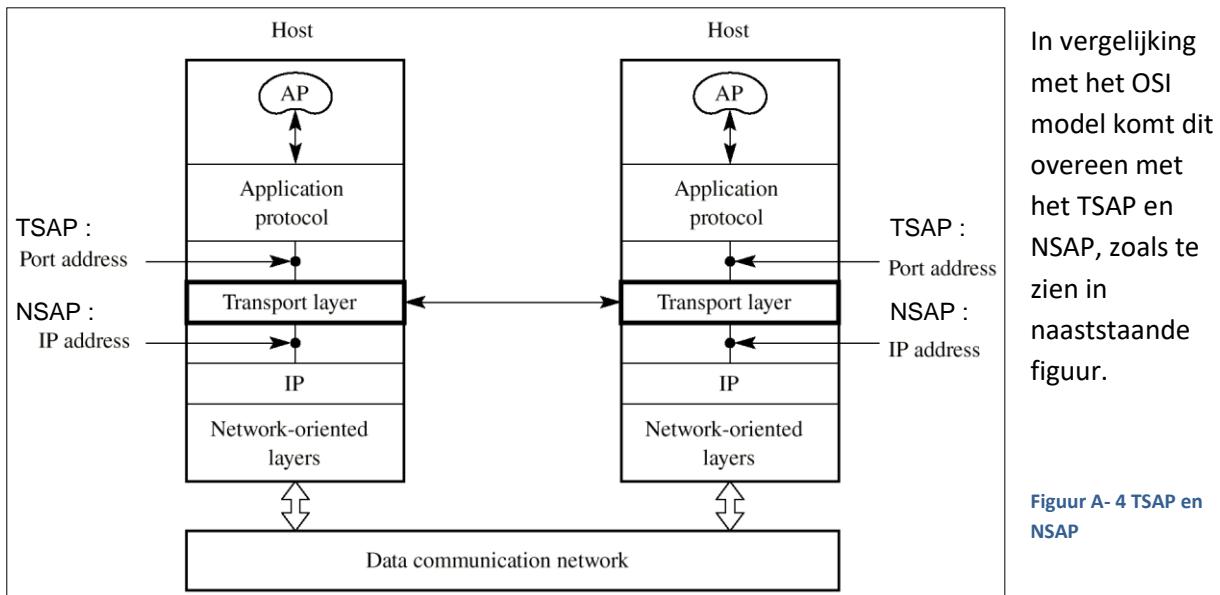
Op 1 host kunnen tegelijk meerdere applicaties draaien, elk met hun resp. protocol, bvb. SMTP, FTP, POP3, DNS, ...

Al deze applicaties gebruiken een transportprotocol (CO → TCP, of CL → UDP).

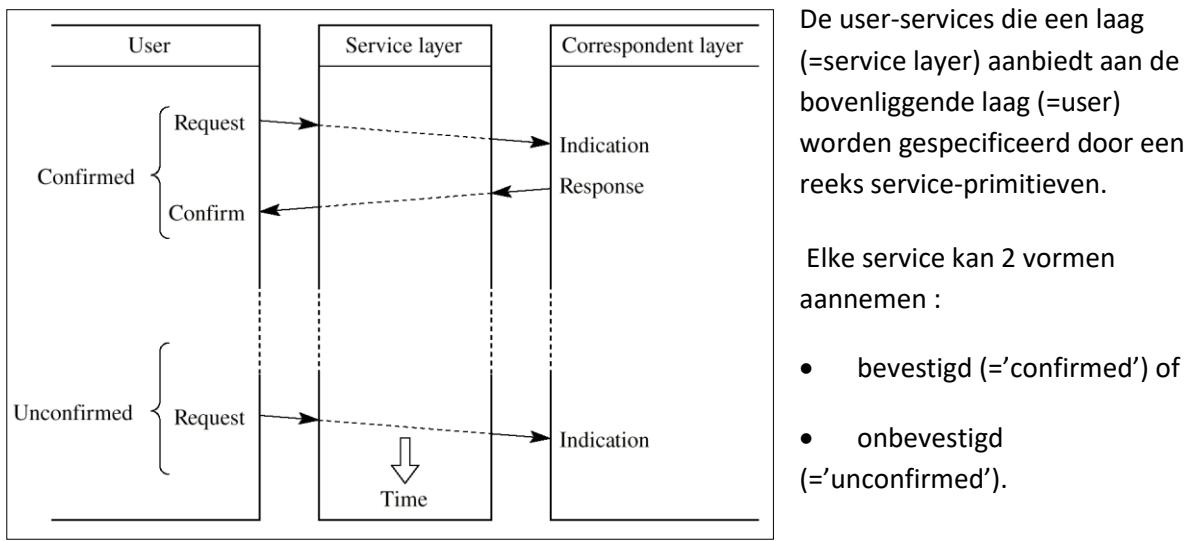


Figuur A-3 TCP/IP suite en zijn interlayer address selectors.

Onafhankelijk van het onderliggende netwerk vindt een IP datagram zijn weg naar het ES a.h.v. het IP adres, de NSAP. Daar binnengekomen ziet het in de IP header (protocol-byte) welk transportprotocol moet bediend worden. De transportlaag beslist a.h.v. het poortnummer in de TCP- (of UDP-) header voor welke applicatie de data bestemd was.

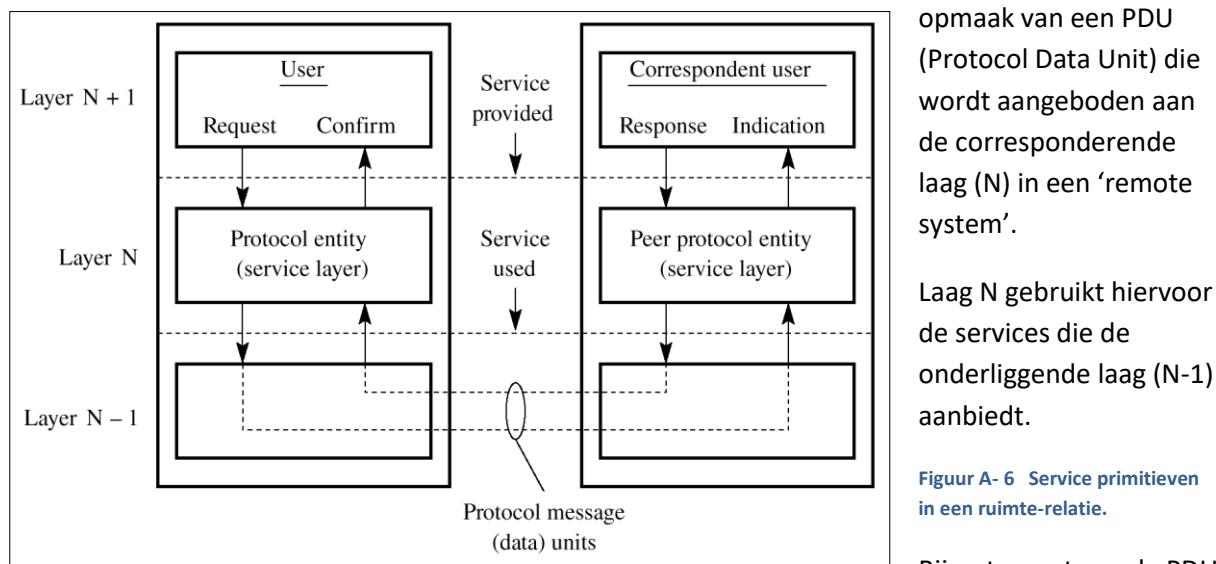


## OSI begrippen : service primitieven



Figuur A- 5 Service primitieven in een tijdsrelatie.

Normaal start een transfer als de user een **request** aanbiedt aan de laag (N). Dit resulteert in de



opmaak van een PDU (Protocol Data Unit) die wordt aangeboden aan de corresponderende laag (N) in een 'remote system'.

Laag N gebruikt hiervoor de services die de onderliggende laag (N-1) aanbiedt.

Figuur A- 6 Service primitieven in een ruimte-relatie.

Bij ontvangst van de PDU

door de corresponderende laag N in het 'remote system', creëert deze een **indication** primitief die hij doorspeelt aan *zijn* gebruiker, de correspondent. Voor *onbevestigde* service houdt het hierbij op.

In het geval van een *bevestigde* service zal de corresponderende gebruiker een **respons** primitief lanceren wat resulteert in het opmaken van een PDU door laag N die wordt verstuurd naar de 'initiator' via de services van de onderliggende laag N-1. Bij ontvangst van deze PDU zal laag N een **confirm** primitief doorspelen aan *zijn* gebruiker.

Één service, bvb. *connect*, bestaat dus uit een aantal primitieven, waarvan de naam zowel het **type** primitief, het **doel** van de service, als de **laag** aangeeft die deze service aanbiedt :

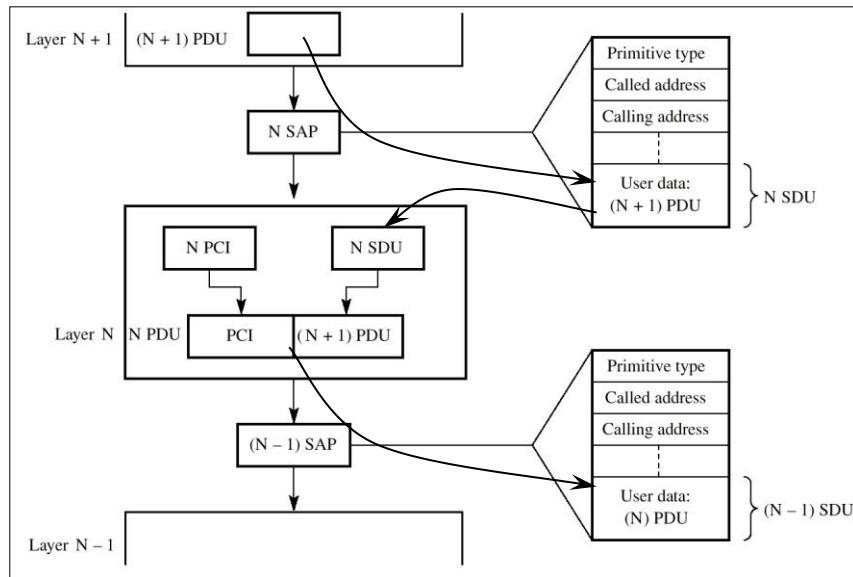
De standaardnotatie van de primitieven bevat dus:

1. de laag      2. Het doel      3. De primitieve zelf      4. Evt. parameters  
bv.      T\_CONNECT.request (called address, calling address, ... , user data)

Dit is een **vraag** van de **SESSIElaag** (5) (= de Transport Service gebruiker) aan de transportlaag om een **connectie** op te zetten met een 'remote' TS (Transport Service) -gebruiker, de remote sessielaaag.

Een van de **parameters** kan handelen over de **kwaliteit** van de service. Deze wordt vooraf opgelegd of onderhandeld. In bovenstaand voorbeeld zijn de '**called address**' en '**calling address**' de SAP's die gebruikt zullen worden in de connectie, bvb. de **poortnummers** bij TCP/IP. Het '**user data**' veld is meestal een **pointer** naar een geheugenbuffer waar de data (=PDU van de bovenliggende laag) opgeslagen is. Zie ook Figuur A- 9.

### OSI begrippen : interacties tussen lagen.



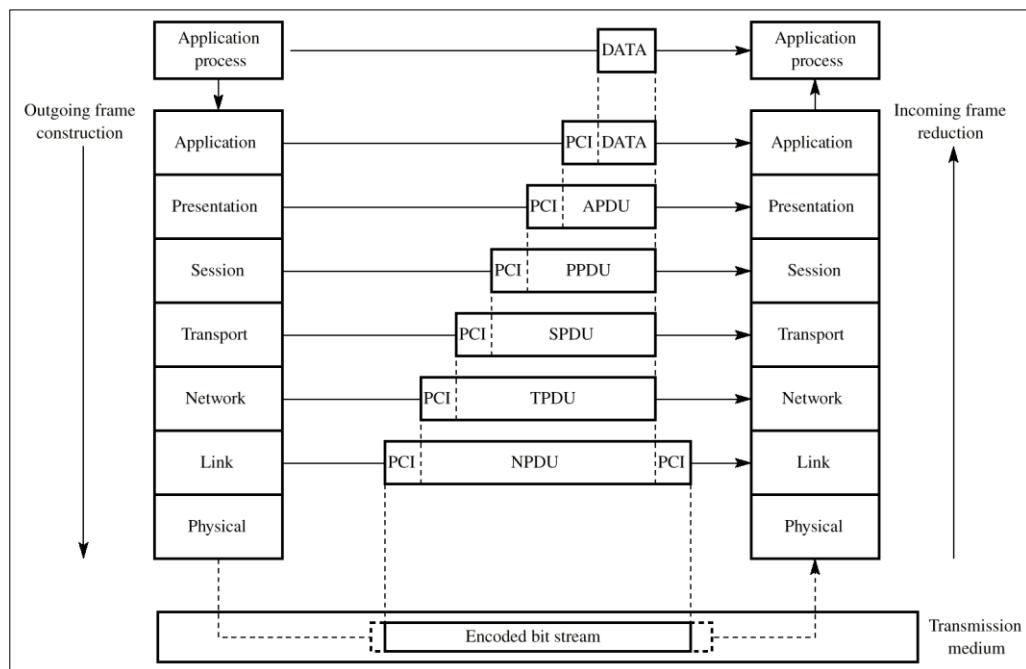
Deze data, aangewezen door een service primitief, wordt voor laag N : SDU (Service Data Unit) genoemd : ze moet ge'serviced' worden door de ontvangende laag N. Aangezien dit overeenkomt met een PDU van de bovenliggende laag kan men stellen dat een (N)SDU = (N+1)PDU.

Figuur A- 7 OSI-interacties tussen lagen d.m.v data units

Bij ontvangst van het primitief, via de N-SAP verbinding, interpreteert Laag N de bijhorende parameters en stelt zelf een PCI (Protocol Control Information) header samen die hij toevoegt aan de SDU om, tezamen, **zijn** PDU te vormen.

OPM. Bij het definiëren van een SAP gebruikt men vaak de letter van de laag waarop de SAP betrekking heeft. Zo is **TSAP** het SAP naar de Transportlaag, m.a.w. voor TCP/IP de '**port**', **NSAP** is de SAP naar de **Netwerklaag**, m.a.w. het **IP**-adres. (Niet te verwarren met NSAP van Figuur A- 7 waar NSAP algemeen als SAP van laag N wordt gebruikt.)

Samenvattend kunnen we stellen dat het datagedeelte van de boodschap groeit naarmate de lagen worden afgelopen, aangezien elke laag zijn eigen PCI toevoegt. In de corresponderende gebruiker zal elke laag (enkel) de overeenkomstige PCI lezen en interpreteren en het (gereduceerde) dataveld doorgeven aan de bovenliggende laag.



Figuur A- 8 De samenstelling van een frame door de verschillende lagen.

We zien dat de oorspronkelijke data per laag een PCI-header krijgt om de corresponderende lagen P2P te verbinden. Men spreekt ook wel van het enveloppe-systeem waar op elke verpakking de relevante informatie staat. Niet alleen het ontvangende systeem, maar ook repeaters, bridges, switchen, routers, ... moeten het pakje enveloppen open maken tot hun resp. functieniveau.

Op de datalink-laag wordt naast de header ook een staartje toegevoegd : de CRC voor foutcontrole. Het is dit totaal dat wordt gecodeerd en opgestuurd met toevoeging van preamble, SFD en EFD.

De Primitieven moeten in volgorde aankomen, zoniet wordt de verbinding verbroken. Bij CO transport moet eerst een verbinding worden opgezet alvorens er data kan worden verstuurd. We geven hiernaast een voorbeeld van transportlaag primitieven en hun parameters, CO zoals TCP.

Primitive	Parameters
T_CONNECT.request .indication	Calling address Called address Expedited data option Quality of service TS_user data
T_CONNECT.response .confirm	Responding address Quality of service Expedited data option TS_user data
T_DATA.request .indication	TS_user data
T_EXPEDITED_DATA.request .indication	TS_user data
T_DISCONNECT.request	TS_user data
T_DISCONNECT.indication	Disconnect reason TS_user data

Figuur A- 9 T\_services primitieven met de resp. parameters voor CO verbindingen.

## OSI: protocolfuncties.

- **Algemeen :**

De belangrijkste protocolfuncties zijn hieronder opgesomd. Het komt erop neer dat error- en flow-controle, link-setup, ... e.a. functies ook in andere dan de datalinklaag voorkomen.

Andere functies zijn :

- *Multiplexing en de-multiplexing*: een onderliggende laag ondersteunt meerdere connecties van een bovenliggende, zie ook Figuur A- 3.
- *Segmentering – reassembling* : één ontvangen SDU wordt gesplitst in meerdere te verzenden PDU's bij de zender en omgekeerd, gereassembleerd bij de ontvanger.
- *Blocking en deblocking* : het omgekeerde van segmentering : meerdere SDU's worden gezamen in 1 PDU verzonden.
- *Sequencing* : controle van de pakketvolgorde
- *Reset* : terugzetten in de begintoestand, vb. voor HDLC de tellers = 0
- *Protocolkeuze en onderhandeling* : op elke laag moeten nog afspraken gemaakt worden welke opties gebruikt worden, QoS, ...
- *Adressering* : alle SAP's moeten een uniek adres krijgen, velen hebben een standaardadres maar daar kan van worden afgeweken.

- **De 7 lagen :**

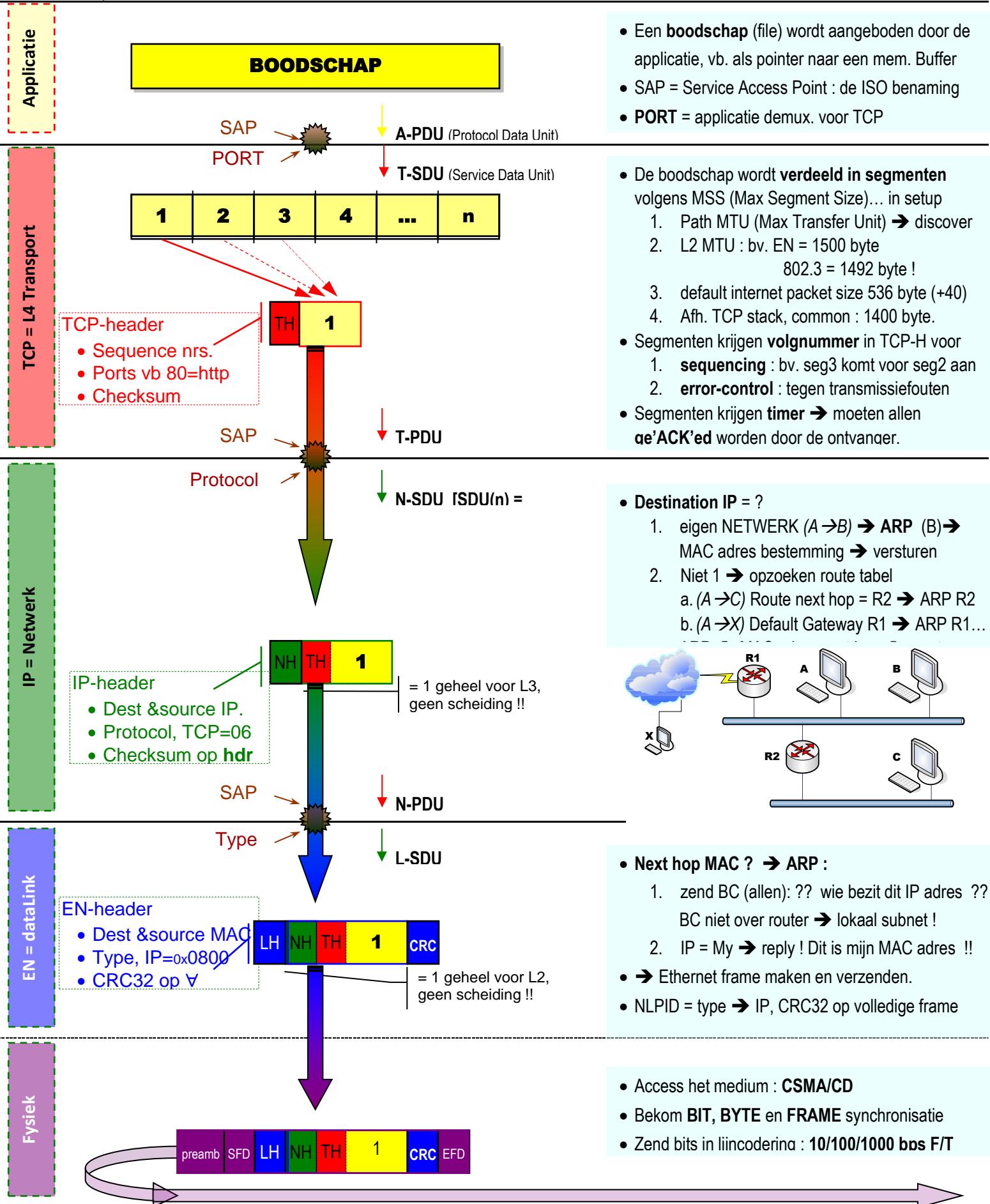
De belangrijkste functies van de lagen zijn reeds vermeld in de inleiding, Figuur 15 op p.29. We geven hier enkel een aantal uitbreidingen op.

- **Datalinklaag** : wordt bij LAN's **opgesplitst** in LLC en **MAC** laag, waarbij deze laatste de gezamenlijke toegang tot het medium regelt.
- **Netwerklaag** : zorgt ervoor dat datagrammen van één node naar een andere gaan, mogelijk over tussenliggende nodes (I.S.) → route uitstippelen → **routerings-** of 'relay'-functie van de netwerklaag. Als sommige routes een kleinere pakket-size ondersteunen → **fragmenteren** en later terug **reassembleren**.
- **Transportlaag** : gaat over het netwerk heen → is een 'end to end' protocol, is netwerk-onafhankelijk. Zorgt ook voor de **sequencing**, het efficiënt gebruik van de onderliggende lagen en de afgesproken **QoS** zoals max. vertraging, min. throughput, ...
- **Sessielaa**g : verzorgt het opzetten en afbreken van een **connectie**, zet **synchronisatiepunten** om bij fouten op terug te vallen, bepaalt het soort dialoog : simplex, half- of full duplex,
- **Presentatielaag** : staat ook in voor **compressie** en **beveiliging** van de gegevens. De verantwoordelijkheid over de syntax van bv. email ligt in zijn bevoegdheid : uuencode, mime,...

- **Applicatielaag** : meest ingewikkelde/uitgebreide laag. Het voorziet de gebruiker van een reeks 'netwerk'services' zoals file transfer (FTAM), directory services (X500), mail (X400), ...

## 5.2 OVERZICHT TCP/IP OVER ETHERNET

### 5.2.1 TCP / IP OVER EN. : ZENDEN VAN EEN BOODSCHAP



## 5.2.2 TCP / IP OVER EN. : ONTVANGEN VAN EEN BOODSCHAP

