Computer Netwerken

De basis uit `IN2605, EE8001, ET4359`

Deze samenvatting dient als basis voor *Networking* in het algemeen. Het bevat de core informatie uit de vakken Computer Netwerken, Telecommunication Networks en Advances in Networking. Pagina's uit de basis naslagwerken worden gegeven door de referentieletter en paginanummer. Referenties naar een hoofdstuk volgen met `H`. Verwijzingen naar gespecialiseerde vakken waar dieper ingegaan is op bepaalde stof zullen altijd plaatsvinden aan het einde van secties.

Basis naslagwerken:

`T` *Computer Networks*, door Andrew S. Taunenbaum (2010)

`M` *Data communications Networking*, door Piet van Mieghem (2011)

Introductie

De belangrijkste toepassingen van computer netwerken zijn research sharing en communicatie (zoals e-mail en VoIP). Research sharing kan in vorm van het client-server model of peer-to-peer (vaak op huisgebied). Het client-server (en peer-to-peer) model zijn een voorbeeld van een interactiemodel en zijn zogenoemd 'event driven'. Andere interactiemodellen zijn gecentraliseerd of distribueerd of via een *finite state machine*.

Naast interactiemodellen zijn er ook communicatie modi. *Broadcasting* en *point-to-point* communicatie zijn basismethoden voor digitale communicatie. *Unicasting* stelt de one-to-one relatie voor, waar één specificieke zender naar één specifieke ontvanger verzendt; *Multicast* bevat de one-to-many en many-to-many relatie; *Anycast* staat logischerwijs voor de one-to-*any* relatie.

Soorten netwerken

- Het Personal Area Network (PAN). `Bluetooth` opereert in PAN's.
- Het Local Area Network (LAN). De bekendste topologieën zijn kamer(/kantoor) niveau waarbij apparaten draadloos (of bedraad) verbonden zijn aan Access Points, welke onderling ook weer verbonden zijn. Voorbeelden zijn `IEEE 802.11` (WiFi, onbedraad) en `IEEE 802.3` (Ethernet).
- Een Metropolitan AN opereert op stadsniveau. Voorbeeld: Cable Television.
- Een Wide AN. Als je er naar kijkt lijkt het een over een continent uitgestrekte LAN, maar anders is het dat met lange kabels ze van verschillende bezitters zijn en dat routers verschillende netwerktechnologieën verbindt. WAN's zijn daarom ook internetwerken. Zo'n backbone, vaak een subnet genoemd, worden geopereerd door een Internet Service Provider (ISP).
- internet. Het Internet (let op de hoofdletter!) gebruikt de ISP netwerken om kantoren en huizen aan elkaar te verbinden.

Netwerken in lagen

Men gelooft dat het 'layering concept' meer voordelen dan nadelen kent. Een complex probleem wordt verdeeld over kleinere, overzichtelijke onderdelen die onafhankelijk behandeld en in parallel uitgevoerd kunnen worden. Informatie van de ene laag is onzichtbaar voor de andere laag - wat ook tot mindere performance kan leiden.

De twee basis modellen zijn het OSI- en TCP/IP model. Lagen zijn van elkaar onderscheiden door middel van interfaces. Elke laag werkt met zijn eigen service data unit en biedt services aan aan de onderliggende laag, zodat informatie in feite door de lagen heen kan reizen. Fig. 1 is een uitbeelding hiervan binnen het OSI model.

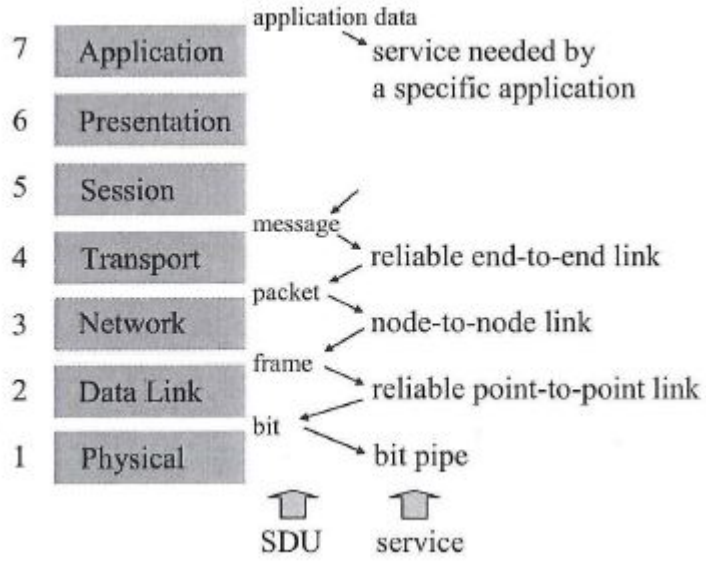

Figure 1: Services en SDUs in het OSI model (uit `M`)

.

Het TCP/IP model kent geen presentatie- en sessielaag en heeft de fysieke en data link laag samengevoegd. Het hart van dit model is de transportlaag, waar TCP en UDP uitgevoerd worden. IP bevindt zich in de netwerklaag.

Het is geconcludeerd dat de presentatie- en sessielaag zwakke concepten zijn. Het systeem voldoet met (minimaal) vijf lagen, welk concept zowel door `T` als `M` worden gevolgd. Meer informatie rondom deze twee modellen zijn te vinden op `T64-70`. De focus van `M` ligt bij de netwerk- en transportlaag.

Services, Protocols en Algoritmen

Het is belangrijk de bovenstaande termen goed te kunnen onderscheiden. Zoals hierboven beschreven, communiceren de verschillende lagen met elkaar d.m.v. services. Ook bestaan er primitieven als `SEND` en `LISTEN`.

Onder de services heb je vier mogelijkheden, waar twee paren gevormd moeten worden. Allereerst staat connection-oriented (waarbij eerst verbinding wordt gemaakt) tegenover connectionless (alles wordt direct verstuurd). Daarnaast heb je reliable tegenover unreliable service, waarbij we kijken of data acknowledged en herzonden wordt. Als voorbeeld, *datagram service* is unreliable connectionless.

Binnen netwerksoorten komt de term connection-oriented ook terug (`M4`). In een netwerk met connection-oriented forwarding zullen alle pakketen dezelfde route volgen (het pad is de 'verbinding'), waar bij connectionless forwarding ieder pakket een andere route zal volgen. Naast deze onderscheiding heb je circuit-switched en packet-switched netwerken. Bij circuit-switched netwerken zal via een *signalling* functie die een routing functie aanroept de verbinding worden opgezet. Circuit-switched netwerken zijn dan ook connection-oriented. Packet-switched netwerken delen de totale informatie op in verschillende pakketen (variabel in lengte) waarna deze pakketen individueel worden verstuurd. In packet-switched netwerken is de routing functionaliteit het belangrijkst. Ze kunnen zowel in CO (zoals TCP) als CL (zoals IP) mode opereren.

Uit `M13` volgt dat het woord *protocol* komt van de Griekse woorden 'eerste lijm'. Een protocol staat voor een set afspraken waar lagen van het zelfde niveau aan voldoen zodat communicatie mogelijk is. Het beschrijft een vorm van communicatie die gerespecteerd / gevolgd dient te worden (zoals in de politiek). Algoritmen beschrijven manieren om een bepaald doel te halen die uitgevoerd dienen te worden (zoals in een kookboek).

"The respect of a protocol could require no, one or many algorithms, depending on what we are speaking about. Algorithms you can use them locally for your own needs, without having to respect any protocol." (`M`)

De Fysieke Laag (TH2)

Signalen

Een signaal kan worden beschreven als een som van een oneindig aantal golven die elk een frequentie hebben die een veelvoud is van de grondfrequentie, volgens Fourier:

$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \sin(2\pi n f t) + \sum_{n=1}^{\infty} b_n \cos(2\pi n f t) \tag{1}$$

Een paar van de golven combineren kan al een voldoende benadering zijn voor de digitale uitkomst. Voor een draad geldt dat amplitudes tussen 0 en een bepaalde f_c onvervormd vervoerd kunnen worden. Deze breedte van frequentie die zonder té erg vervormd (*attentuated*) heet de bandbreedte. Deze breedte is onafhankelijk van de beginfrequentie. Op deze manier kun je naast *baseband* signalen ook opgeschoven *pass-band* signalen creëren die in álle wireless transmissies gebruikt worden. Voor een eindige bandbreedte, ruisloos kanaal geldt dat er een maximum data rate (signaalsneldheid) bestaat. Wanneer een signaal over V discrete levels bestaat, dan geldt Nyquist's theorie:

$$\text{maximum data rate} = 2B \log_2 V \text{ bits/sec} \tag{2}$$

Voorbeeld: $B = 3$ kHz en $V = 2$ geeft een max data rate van 6000.

Voor kanalen mét ruis heb je te maken met Shannon's Capacity:

$$\text{maximum data rate} = B \log_2(1 + S/N) \text{ bits/sec} \tag{3}$$

De Signaal-Ruis verhouding (S/N) wordt vaak gegeven in dB, waarbij $(S/N)_{normaal} = 10 \log_{10}(S/N)_{dB}$ (1000 = 30 dB).

Transport Media

Ondanks dat er tegenwoordig heel veel op disks past, is dat vaak niet 'the way to go'. Drie belangrijke soorten zijn Twisted Pairs (gednaliseerde kabels, tegenwoordig Category 6 UTP (*unschielded* TP), koperkabels (met grote bandbreedte) en powerlines. Andere terminologie is:

- full-duplex = tegelijkertijd verkeer in beide richtingen mogelijk

- half-duplex = beide richtingen mogelijk maar niet tegelijkertijd
- simplex = een one way street.

Draadloze communicatie maakt gebruik van het elektromagnetisch spectrum. Denk hier aan radio- en microgolven, infrarood en zichtbaar licht. Fiber gebruikt zelfs ultraviolet frequenties (`T127`). In draadloze communicatie worden bits/symbolen uitgebeeld door de amplitude, frequentie of fase van een signaal te moduleren.

Radiogolven, werkend in de VLF, LF en MF banden, zullen vooral op de lagere frequenties gebruikt worden, waar ze goed door obstakels gaan maar wel langzaam afsterven (ook wel *pathloss* genoemd). Microgolven worden veel gebruikt voor long-distance telefoon communicatie of televisiezenders, dat er een tekort aan spectrum is. De golven boven 100MHz kunnen goed recht gefocust worden, maar kennen wel *multipath fading* en kunnen niet door obstakels. Voor veel 'onbetaalde' stralen worden de ISM (industrial, scientific, medical) banden gebruikt.

Verdiepende informatie

- `ET4358` Wireless Communications:
  Karakteristieken van de fysieke laag, formulering van *pathloss* en *multipath fading*.
- `ET4169` Microwaves, Radar, Remote Sensing:
  Karakteristieken en gebruik van radio- en microgolven.

Modulatie

Het woord *modem* staat voor modulation-demodulation. Digitale modulatie is het converteren tussen bits en representerende signalen en is de brug tussen digitale data en de fysieke laag. Modulatie kan gedaan worden met line codes, waarbij we brandbreedte efficiëntie, clock recovery en DC balans tegen elkaar moeten afwegen. Verschillende methoden zijn met bijbehorende output te vinden op `T146`.

In line codes kunnen meerdere signaallevels gebruiken zodat meerdere bits tegelijkertijd verzonden kunnen worden. Deze combinaties heten symbolen. We spreken dan van *symbol rate* of *baud rate*. Er geldt: bit rate = baud rate $\cdot$ bits per symbol.
Voorbeelden van schema's zijn:

- **Manchester**, ofwel Non-Return to Zero, waarbij 1 het positieve en 0 het negatieve voltage is. Wordt geXORed met de clock.
- **4B/5B** waarbij een reeks van 4 bits wordt gemapt op 5 bits waarbij lange reeksen van 0000 worden voorkomen.
- Bij **8B/10B** wordt ook, door ook te werken met bipolar encoding, gerwerkt aan een gebalanceerd kanaal door te kijken naar disparity.

Bij passband signalen (voornamelijk draadloos) wordt er geknutseld aan een standaard golf. Dit kan met Amplitude/Frequency/Phase Shift Keying, met bekenste variant BPSK (Binairy Phase Shift Keying). In Quadrature Amplitude Modulation worden verschillende eigenschappen gecombineerd veranderd. De mogelijkheden hiervoor zijn af te lezen in een *constellatiegram*. Een voorbeeld is QAM-16, waarbij één punt 4 bits voorstelt.

Naast modulatie kennen we *multiplexing*. Bij FDM geschoven aan de passbanden frequentie, zodat de signalen niet intefreren. Bij TDM is iedereen om de beurt aan de beurt in tijd met tussendoor een *guard time*. Bij statical TDM is het statistisch bepaald wie de tijd krijgt, géén fixed schema.

Verdiepende informatie

- `ET4358` Wireless Communications:
  Verdieping van zowel modulatie- als multiplexing methoden.

De Data Link Laag (TH3)

De Data Link Layer (DLL) probeert ervoor te zorgen dat er een reliable, efficiënte communicatie van hele frames tussen twee machines bestaat. Vanaf nu kan je er vanuit gaan dat hetgeen wat hiervoor besproken is werkt; dit is namelijk het hele idee van het layerings concept.

De DLL verzorgt *unacknowledged connectionless service*, *acknowledged connectionless service* en *acknowledged connection oriented service*. Focus in de DLL is betrouwbare point-to-point verbinding. Daarnaast splitst de laag de lange bitstreams op in frames om hersturen en acknowledgen mogelijk te maken. Om de ontvanger de frames te laten herkennen kan allereerst byte count gebruikt worden. Hierbij geeft de eerste byte het aantal totale bytes aan. Echter, bij een bitflip kan dit fout aflopen.

Synchonisatie herstellen gaat beter bij *byte stuffing* i.c.m. *flag bytes*, die aan 't begin en eind van elke frame zitten. Om te voorkomen dat een flag byte binnen frames worden onderschept wordt er voor elke niet-geldige flagreeks een escape-byte geplakt (etc). Dit wordt gebruikt in PPP. Dit zogenaamde stuffen kan ook gebeuren op bitniveau. Neem flag = 01111110, dan voegt men een escape bit toe indien er teveel 1-tjes acher elkaar gevonden worden, ofwel 11111 $\rightarrow$ 11101. Wanneer je niet wilt stuffen zijn er nog 'coding violations'.

Ook verzorgt de DLL error en flow control. Dit tweede is er vooral om ervoor te zorgen dat de zender de ontvanger niet overdondert met data. Dit kan feedback-based of ratebased (met een vastgezette maximum rate).

Error Control (TH3.2 / MH3)

De DLL is verantwoordelijk voor de error control voor de hop-by-hop transmissies, terwijl in de transport laag de end-to-end reliability wordt gecontroleerd. Onder error control valt zowel detectie en correctie. Error correctie (*Forward Error Correction*) wordt toegepast op noisy kanalen, waar de kans groot is dat een herzonden frame ook fouten bevat. FEC is voor vele real-time applicaties de enige mogelijkheid wanneer er niet genoeg tijd is voor retransmissies. Veel non-real-time applicaties gebruiken error detectie in combinatie met retransmissies. Error detection vindt vaak plaats als de in bedrade netwerken, omdat herzenden sneller is dan fouten herstellen.

Error Correction Codes

In FEC wordt aan frames genoeg redundante data toegevoegd om de originele data te berekenen. Bij de redundante informatie komt deze voor in blokken, waarbij door deze r bits het correcte woord van lengte m gevonden kan worden. Met $n = m + r$ volgt als code rate (ofwel percentage nuttige informatie) $\frac{m}{n}$. Hoe kleiner de code rate, hoe meer redundante informatie en dus grotere kans dat foutief ontvangen informatie hersteld kan worden.

In `T` worden Hamming codes, binary convolutional codes, Reed-Solomon codes en Low-Density Parity Check besproken. De laatste drie kunnen gebruikt worden om burst errors te corrigeren. Hamming codes is een simpelere principe hieronder kort uitgelegd, met een goed voorbeeld in `T`.

De Hamming distance is de hoeveelheid bitposities die de geldige x aantal codewoorden van grootte n van elkaar verschillen. De code is error-correcting voor een lijst met $2d + 1$ afstand, waarbij d het aantal fouten is. Error-detecting kan t/m $d + 1$. Bij Hamming codes stellen de bits 1,2,4,8 (etc) de checkbits voor. Fouten kunnen hersteld worden door de een combinatie checkbits te XOR-en en te vergelijken met het bijbehorende bit. Zo wordt bit 11 berekend uit bit 1,2 en 8.

Error Detection

Error detectie gebruikt een operatie O waarvan de uitkomst moet overeenkomen met een afgesproken uitkomst. Indien de uitkomst afwijkt, zit er een fout in de ontvangen bitreeks. Een voorbeeld van deze operatie O is (mod 9), welke gebruikt wordt in de zogenaamde *nine-proof*(`M53`).
In beide tekstboeken worden drie error detectie methoden besproken:

- **Single Parity Checks**
  Hier wordt aan de data (of een blok data) een enkel *parity bit* toegevoegd dat aangeeft of de som van de data even of oneven is. Bij een correct getransporteerd frame geldt dan:
  $$\bigoplus_{j=0}^{k} c_j = \left(\bigoplus_{j=0}^{k-1} m_j \right) \oplus c_k = 0 \tag{4}$$
  Gegeven de codewoorden valt vlug te concluderen dat de hamming distance $d = 2$. De single parity check code kan daarom $d - 1 = 1$ bit error detecteren, maar geen errors corrigeren. Bij het gebruik van interleaving van parity bits kunnen burst errors gedetecteerd worden (ter breedte van de blokken) (`T231`).

- **Checksums**
  Een checksum is een uitbreiding van parity bits, waarbij er een groep redundante bits als checksum wordt gebruikt. In het Internet bestaat R uit een 16-bit checksum. Voor het codewoord C (inclusief R) geldt, indien deze correct is getransporteerd: $C \mod (2^{16} - 1) = 0$.

Voor R geldt, zeer gemakkelijk te berekenen met *1s complement*:

$$R = -\sum_{j=0}^{k-1} m_j \mod (2^{16} - 1) \tag{5}$$

- **Cyclic Redundancy Check (CRC)**
  Deze methode wordt in de meeste communicatienetwerken gebruikt. Hier wordt met polynomialen gewerkt. Er is een generator $G(x)$ waarmee de R portie bepaald kan worden. Dit gebeurt door de M portie te delen (zonder *carry*) door de bit representatie van $G(x)$. De remainder van deze operatie levert R, eventueel aangevuld door nullen om de correcte lengte te verkrijgen ($|R| = |G(x)| - 1$). Voorbeelden van CRC berekeningen vind je op `T234, M57`.

  De gebruikte generator bepaald uiteindelijk wat voor fouten er gedetecteerd kunnen worden. Beschrijven we de inverted bits als een polynomial $F(x)$, dan is de fout detecteerbaar indien $F(x)$ niet deelbaar is door $G(x)$. Elke generator die $x + 1$ bevat kan een fout detecteren met een oneven aantal inverted bits. Een andere veelgebruikte generator is $G(x) = x^{16} + x^{12} + x^5 + 1$, welke enkele, dubbele of triple errors in berichten korter dan 32752 bits kan detecteren.

Verdiepende informatie

- `ET4030` Error Correction Codes: ♥
  Uitgebreide behandeling van alle bovengenoemde (en meer) ECC's.

Retransmissie Protocols

Retransmissie protocols worden nageleefd wanneer frames niet aankomen of error gedetecteerd zijn (en niet gecorrigeerd worden). Het doel is: *Implement a reliable packet channel over an unreliable one.* De protocols uitgevoerd in de DLL zijn van belang in netwerken waarbij de kwaliteit van de connecties laag zijn, zoals in sensor netwerken. In de meeste datacommunicatienetwerken is de kwaliteit hiervan wel hoog, waardoor de error control daar vooral doorgevoerd wordt op de transportlaag, wat TCP belangrijk maakt. Dit kan vergeleken worden met *Utopia* protocol (`T`) waar er één data stroom is zonder verlies, met de ontvanger rustig wachtend op wat er komt en dit doorgeeft aan de Network Layer.

Retransmissie protocols zijn voorbeelden van acknowledgement schemes. Hierbij geldt wel dat zowel de data als acknowledgement frames duplicaten kunnen bevatten. Data kan bijvoorbeeld dubbel verzonden worden doordat de acknowledgement later dan de timer toestaat binnenkomt bij de zender. Het probleem van duplicaten wordt voornamelijk opgelost door het gebruik van *sequencenummers*.

Protocols waarbij de zender wacht op een positief acknowledgement voordat hij verder gaat met het volgende frame heten Automatic Repeat reQuest (ARQ) protocols, of Positive Acknowledgement with Retransmissions (PAR) (`T247`). Besproken protocols in beide tekstboeken zijn het Stop-and-Wait protocol, Go Back n protocol en Selective Repeat.

Stop-and-Wait

In het *Stop-and-Wait* protocol wacht de zender op elke `ACK` voordat een volgende frame wordt gestuurd. Hierom is werken met een sequencenummer van 0 of 1 voldoende, waarom het protocol ook wel het Alternating Bit Protocol wordt genoemd. Het protocol kan uitgebeeld worden als een finite state machine met vier states ((0,0) (0,1) (1,1) (1,0)).

De performance van dit protocol hangt van verschillende dingen af. Het blijkt dat Stop-and-Wait efficient is als de transmissie rate C klein is. Daarnaast verlaagt de efficiëntie als er fouten op gaat treden. De uiteindelijke formule is:

$$\eta_{S\&W;p} = \frac{l_{packet} - l_{header}}{t_{packet} + t_{ack} + 2(Ct_{prop} + t_{proc}) + \frac{p}{1-p}CT_{RTO}} \leq (1-p)\eta_{S\&W} \tag{6}$$

Waarbij T_{RTO} de retransmission time-out voorstelt en C de transmissie rate. Er volgt uit deze formule, naast bovenstaande geconcludeerde, dat langere frames de kans voor een succesvolle tranmissie verkleind $(1-p)$ en dus de efficiëntie ook verkleind.

Sliding Window Protocols

Wanneer data twee kanten op gaat of we efficiënter te werk willen gaan, gebruiken we *sliding-window protocols*. Allereerst het principe *piggybacking*. Hiermee wordt bandbreedte efficiënt gebruikt door een acknowledgement mee te sturen met een dataframe - behalve als het te lang duurt, dan gaat het gewoon zelf. In het window van de zender staan de sequencenummers voor de verstuurde frames waarvoor nog geen acknowledgement is ontvangen, welke in 't geheugen blijven voor eventuele herzending. Aan de andere kant houdt de window van de ontvanger de nummers bij die hij kan ontvangen. Een window van 1 houdt (logischerwijs) in dat de ontvanger alles op volgorde wil ontvangen.

De window size van de ontvanger kan als volgt berekend worden (`T`). Het *bandwidth-delay product* BD is de bandbreedte maal een one-way transit (in seconden). Deze gebruiken we om te kijken hoeveel frames w er in de lucht kunnen zitten zonder dat we vastlopen.

$$\text{link utilization } \leq \frac{w}{1+2BD} \tag{7}$$

Neem $B = 50$ kbps, een enkeltje duurt 250ms, dan $BD = 12,5$ kbit (ofwel 12,5 frames van 1000 bits). Een link utilization van 100% geeft $w = 26$. Om overlap te voorkomen geldt nog: window size $= (\texttt{max\_seq} + 1)/2$.

Kort kunnen de twee besproken sliding-window protocols als volgt omschreven worden. In het *Go-Back-N* protocol gaan we terug naar dát frame waar het mis is gelopen en starten we vanaf daar met opnieuw verzenden (ofwel, $W_{reciever} = 1$). Bij *Selective Repeat* zal de ontvanger ook frames bufferen, waardoor slechts alleen het frame waar het mis ging herzonden hoeft te worden. Indien een frame incorrect ontvangen is wordt er een `NACK` verstuurd. De volgende (zogenaamde cumulatieve) `ACK` laat dan weten dat

alles tot aan het acknowledgement (dus alles wat gebufferd is ook) correct is ontvangen.

Wanneer we de drie besproken protocols vergelijken, dan volgt $\eta_{S\&W} < \eta_{gbn} < \eta_{SR}$ (`M72`). Alle efficiënties stijgen monotoon met een dalende error probability p. Daarnaast is Selective Repeat onafhankelijk van de karakteristieken van het kanaal (welke gerelateerd is aan BD). Tegelijkertijd geldt $C_{S\&W} < C_{gbn} < C_{SR}$, dus welk protocol te pakken is een typische engineering trade-off.

Medium Acces Control

In broadcast channels heb je last van het zogenaamde *channel allocation probleem*. Hier gaat de Medium Access Control (MAC) laag over, een sublaag van de DLL naast de Logical Link Control (LLC) laag. Eerder is *multiplexing* genoemd als oplossing voor channel allocation, maar statische oplossingen als FDM en TDM hebben hun tekortkomingen zoals heel veel delay en weinig efficiëntie.

Multiple Access kan op verschillende manier onderscheiden worden. Er kan gekeken worden naar de manier hoe men tot resoluties komt en of oplossingen conflict-vrij zijn of niet. Een goed overzicht van deze onderscheidingen vindt je op `M30`. Belangrijk aantekening is dat MAC protocols zich bezig houden met per-link communicatie. Routing protocols (transportlaag) gaan over end-to-end communicatie.

Ethernet

Local Area Networking (LAN) kan gezien worden als de basis *access network implementation*. Een groot besproken onderwerp is `IEEE 802.3` (Ethernet), waar CSMA wordt geïntroduceerd om het channel allocation probleem op te lossen. Logischerwijs heten netwerken die CSMA gebruiken ook wel Ethernets.

De meest standaard (en zeer populaire) LAN topologie is de ster topology, omdat hier een kabelbreuk slechts een enkele node aantast. Kleine topologieën kunnen vervolgens worden uitgebreid door gebruik te maken van repeaters, bridges en routers. Een repeater is hardware die simpel een signaal van de ene kabel doorstuurt naar de andere. Een bridge, ook wel Ethernet switch, controleert ook of het door te sturen signaal geen ruis of fouten bevat.

Ethernet maakt gebruikt van MAC adressen. Dit MAC adres bestaat uit een 24-bit *company ID* en een 24-bit *board ID*, welke uniek zijn voor interface card. Dit betekent dat Ethernet / MAC adressen fysieke ofwel hardware adressen zijn.

Classic Ethernet is een broadcast technologie met best-effort delivery intenties en *distributed access control* dat gebruikt maakt van CSMA/CD met binaire exponentiele backoff (hieronder besproken). Distributed access control indiceert dat de toegangsregels gedistribueerd worden geregeld en niet door een centraal apparaat. *Centralized multiple access* is er bijvoorbeeld in de Bluetooth standaard en kan je zien als een master-slave communicatie type. Best-effort delivery doelt erop dat er geen packet receipt notificaties zijn. De geïnteresseerde lezer vindt op `T300` het frame formal voor Classic Ethernet.

Naast Classic Ethernet is er ook switched Ethernet, dat gebruikt maakt van switches.

Bij een switch zijn - in tegenstelling tot een hub waar alle draden tot een centraal punt komen - alle computers langs verschillende wegen te bereiken. Hierdoor vinden collisions minder vaak plaats. Indien de kabel half duplex is, werken we met CSMA/CD op de normale manier. Een volgende verbetering was Fast Ethernet over twisted pairs, daarna gigabit Ethernet, met carrier extension en frame bursting en de 8B/10B encoding. En last, 10-Gigabit.

Verdiepende informatie

- `ET4394` Wireless Networking:
  Protocols als `IEEE` 802.11x (WiFi), 802.15.x (Bluetooth) en 802.15.4 (ZigBee).

Protocols

Bij de bespreking van MAC protocols gaan we uit van onafhankelijk verkeer, één kanaal, observeerbare collisions, tijd die continu of in slots is verdeeld en wel of geen *carrier sense*. MAC protocols zijn naast in het Ethernet direct toepasbaar in ad hoc netwerken. Ook moet je er rekening mee houden dat multiacces protocols geen reliable service biedt, maar dat dit geregeld wordt door de rest van de DLL (zoals de LLC).

Onderstaand worden de protocols in evolutie volgorde uitgelegd. Kort houdt deze evolutie het volgende in:

1. Het eerste random access protocol was ALOHA.
2. Een verbeterde versie was Slotted ALOHA toen de packet transmission tijd verdeeld werd in slots.
3. CSMA (Carrier Sense Multiple Access) is het protocol dat volgt door de introductie van carrier sense.
4. De toevoeging van een random backoff tijd geeft CSMA/CA.
5. MACA (ofwel CSMA/CA met reservations) volgt door de toevoeging van RTS, CTS en de NAV.

Naast deze basis protocols is genoemd dat Ethernet werkt met CSMA/CD. WiFi gebruikt collision avoidance en geen collision detection. Dit is omdat transceivers niet kunnen zenden en luisteren op hetzelfde moment. Er is een te groot verschil is in send power (100 mW) en receive sensibility (0.01-0.0001 mW).

ALOHA

Kortgezegd mag in het ALOHA protocol iedereen zenden wanneer hij maar wil, maar zodra er twee frames overlappen wordt het resultaat altijd weggegooid. ("*bad is bad*"). We zitten in een *contention system*; Bij mislukking volgt een time-out voordat we

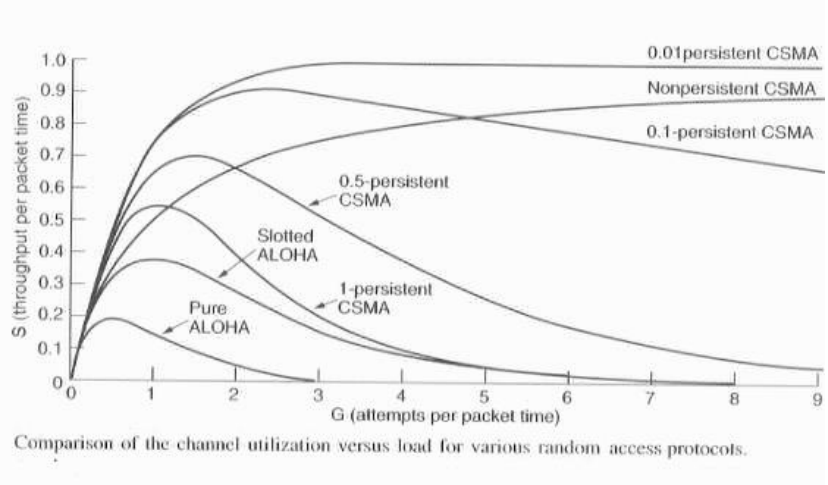

Figure 2: Performance van de verschillende basis MAC Protocols

herzenden. In Pure ALOHA is deze time-out ook in continue tijd, in Slotted ALOHA wordt er gewerkt met tijdslots.

De aankomst van pakketten kan gezien worden als een Poisson proces met parameter λ, het aantal pakketten dat per seconde binnenkomt, De kans dat er in een periode t geen enkel pakket wordt verstuurd is dan $\Pr[k=0]_{pure} = e^{-2\lambda t}$. In pure ALOHA mag een tranmissie op elk moment plaatsvinden. Hierdoor kan er aan twee zijden van de tranmissietijd (pakketlengte) een overlap plaatsvinden, waardoor er tweemaal de pakketlengte gewacht moet worden. Bij slotted ALOHA is er maar één mogelijk startmoment, dus volgt $\Pr[k=0]_{slot} = e^{-\lambda t}$.

In Fig. 2 is de throughput van ALOHA af te lezen, het aantal pakketen dat correct aankomt. Indien we niet kijken per packet time zoals in Fig. 2, volgt $G = \lambda t$. Nu kunnen we de throughput (*maximum channel occupation*) schrijven als hieronder:

$$S_{pure} = Ge^{-2G}, \qquad \text{met } S_{max} = \frac{1}{2e}, \ (\lambda = 0.5) \tag{8}$$

$$S_{slot} = Ge^{-G}, \qquad \text{met } S_{max} = \frac{1}{e}, \ (\lambda = 1) \tag{9}$$

Hieruit volgt dat we met slotted ALOHA de dubbele throughput (38%) kunnen bereiken.

CSMA

In Carrier Sense Multiple Access Protocols kan er aan het kanaal 'geroken' worden of er al iets verzonden wordt en dus concluderen of iets verzonden kan worden. De term *carrier sensing* refereert naar het waaarnemen van de "carrier" wave. Ook bij CSMA wordt er gewerkt met tijdslots. In de basis hebben we verschillende simpele varianten: Hierbinnen heb je verschillende simpele varianten:

- **(1-)persistent CSMA** Als het kanaal vrij is stuur je met een kans van 1, anders wacht je en stuur je daarna per direct.

- **nonpersistent CSMA** Als het kanaal vrij is sturen we en anders wachten we een random tijd en kijken we dan weer eens. Dit leidt tot beter gebruik van het kanaal, maar langere delays.

- **p-persistent CSMA** (op slotted kanalen) Met een kans p verzenden als het kanaal vrij is en wachten met een kans $q = 1 - p$.

Bij p-persistent CSMA is de keuze van p van invloed op de performance van het systeem. Gegeven dat er N onafhankelijke stations zijn, verwachten we collisions indien $Np > 1$. Het pakken van $p < \frac{1}{N}$ is niet direct dé oplossing, omdat een grote N en dus kleine p leidt tot een vergrote wachttijd. Het volgt dat de kans op een sucesvolle transmissie (S) een binomiaal gedistribueerd is met parameter p.
Er geldt $\Pr[S] = \Pr[Y = 1] = \binom{N}{1} p(1-p)^{N-1}$ en $\max(\Pr[S]) = \left(1 - \frac{1}{N}\right)^{N-1}$.

Een uitbreiding op de CSMA wat betreft de wachttijd is het gebruik van een *binary exponential backoff* (ook gebruikt in Ethernet). Hierbij wordt de wachttijd uniform gekozen tussen $[0, (2^j - 1)]$, afhankelijk van het aantal j collisions. Deze aanpak past dynamisch het aantal actieve stations aan.

CSMA/CD

Een uitbreiding op CSMA is *collision detection*. Dit is een analoog proces waarbij de Power van het ontvangen signaal vergeleken wordt met het verzonden signaal. Hierboven was al genoemd dat dit proces niet uitgevoerd kan worden door antennas, waarom WiFi werkt met CSMA/CA.

In CSMA/CD heb je drie perioden: de transmissie periode, contention periode en idle periode. De contention periode kan één of meerdere collisions bevatten en heeft daarom dus een variabele lengte. Gegeven dat $p_s = \max[\Pr(S)]$ de kans is dat één van de N stations succesvol verzendt, volgt dat de kans dat de contentie periode Δ gelijk is aan k een geometrische random variabele is: $\Pr[\Delta = k] = p_s(1 - p_s)^k$.

Uit hierboven volgt $E[\Delta] = \frac{1}{p_s} - 1 \leq e - 1$. Een gemiddelde contention periode duurt nooit langer dan $e - 1 \approx 1.718$ tijdslots.

De efficientie van Ethernet hangt af van de verwachte lengte van transmissie en contentie. Op `M41` volgt:

$$\eta = \frac{1}{1 + f(v)\frac{BD}{E[L]}} \tag{10}$$

Waarbij $f(v) = \frac{v}{0.8c}$ en $v = 2(\frac{1}{p_s} - 1)$. Uit simulaties blijkt $v = 5$ voor Ethernet. Uit al deze formules blijkt dat de efficiëntie van CSMA/CD groeit wanneer er een grotere frame lengte ($E[L]$) wordt gebruikt en krimpt wanneer het netwerk (D) groeit. In de praktijk zien we dit terug in Ethernet omdat het netwerk opgedeeld is in subnets om een hogere efficiëntie te bereiken.

CSMA/CA

Voor draadloze kanalen is er geïnvesteerd in CSMA/CA, omdat enerzijds collision detection niet mogelijk is en anderzijds het draadloze kanaal met extra problemen kampt. In het *hidden terminal probleem* is er kans dat er collisions optreden doordat twee concurrerende terminals elkaar niet kunnen ontdekken omdat ze buiten elkaar range staan. In het *exposed terminal probleem* kunnen zendacties uitgesteld worden omdat een andere transmissie concurrent lijkt, maar deze een andere bestemming heeft.

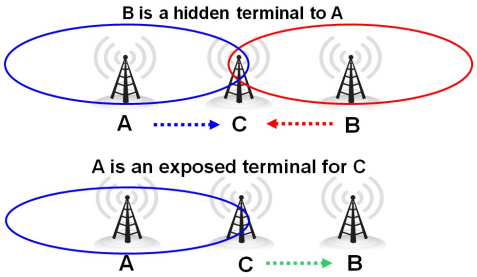

Figure 3: Vizualisatie van het *hidden en exposed terminal probleem*

Deze problemen worden verholpen door te werken met extra pakketen: `ACK`, `RTS` en `CTS`. `ACK`'s zijn acknowledgements dat transmissies zijn gelukt, maar is geen probleemoplosser. Dat is het `Request To Send / Clear To Send` systeem. Hier wordt door middel van kleine controleberichten eerst kennisgegeven aan het netwerk (en gewenste apparaat) dat er de zender wil zenden en het gewenste apparaat reageert of dit allemaal OK is of niet. Door de toevoeging van deze pakketen kunnen er feitelijk alleen collisions optreden tussen de controleberichten. Hierdoor gaat geen data 'verloren' en is de contention periode ook aanzienlijk kleiner.

Zowel de `RTS` als `CTS` pakketen bevatten meer informatie over de te komen data. Stations kunnen hun zogenaamde *Network Allocation Vector* updaten met deze informatie, zodat zij op de hoogte zijn van wat er kan komen en er op een slimme manier aan het medium kan worden gevoeld. Zo wordt het medium pas *idle* genoemd indien er minsten een DIFS (*distributed interframe space*) verlopen is. Tussen verschillende data pakketen zit een Short IFS, welke korte is dan een DIFS.

Een uitbreiding op MACA is MACAW, waarbij de berichtvoering is in de vorm `RTS-CTS-DS-DATA-ACK`. Daarnaast wordt er gewerkt met een back-off die op een *Multiplicative Increase, Lineair Decrease* wordt aangepast. MACAW valt in het categorieën systeem van MAC protocols onder *contention-based, sender-initiated, single-channel* protocols, net als de andere protocols hierboven besproken.

Verdiepende informatie

- `ET4394` Wireless Networking:
  ALOHA besproken als Markov Chain, CSMA/CA *DCF* gebruikt in WiFi.
- `IN4341` Performance Analysis:
  Performance analyse van ALOHA.
- `ET4388` Ad Hoc Networks:
  MAC protocols als Busy Tone MA, CATA en Hop Reservation MA.

De Network Laag (TH5)

De Network Layer zorgt voor services voor de Transport Layer, waarbij deze tweede niks van routers, topologieën e.d. wil weten, maar daarentegen wel van een uniformachtig netwerkadres. Deze service is of connectionless of connection-oriented. In het eerste geval werken we met een *datagram netwerk*, waar alles gewoon verstuurd wordt. In een connection-oriented service leggen we de route vast in een *virtual circuit*. Verschillen en overeenkomsten tussen datagram netwerken en virtual circuits staan op T379. Meest logische verschil is de behoefte van circuit setup die er in DN niet en in een VC wel is. Daarnaast worden de routes voor de datagram packetten onafhankelijk van elkaar vastgesteld.

Voordat we ingaan op de werking van het Internet, bespreken we routing algoritmen en protocols. Let hierbij op het verschil tussen *routing*, het bepalen van de route van source tot destination en *forwarden*, het opzoeken van de juiste lijn om de gevonden route te volgen.

Routing Algoritmen en Protocols

In de introductie van deze samenvatting kan de relatie tussen algoritmen en protocols naar voren. Een routing protocol probeert iedere node van het netwerk een consistente weergave van de topologie te geven, die ook up to date is. Het protocol neemt daarom de tijd variërende dynamieken van het netwerk mee (topology changes en traffic). Aan de andere kant is er het routing algoritme, dat op basis van een topology update de *forwarding table* voor elke node berekent. Ofwel, de uitvoering van een protocol kan gebruik maken van een of meerdere algoritmen.

Soorten Routing

Zowel binnen algoritmen als protocols zijn vier eigenschappen waar routing op verschillende manier kan worden aangepakt:

1. Bij *static* (non-adaptive) *routing* wordt er met offline computaties een goede oplossing gevonden voor kleine, geautorizeerde netwerken en statische netwerken. Aan de andere kant is er *dynamic* (adaptive) *routing* waarbij berekeningen afhankelijk zijn van de topologie en we werken met automation zodat 'second kind changes' en failures kunnen worden behandeld.

2. Routes worden op één centraal punt bepaald of op een gedistribueerde manier in resp. *centralized* of *distributed routing*.

3. In *source routing* worden complete paden berekend waarbij een pad gegarandeerd loop-vrij is en waarbij multiple constraints van Quality of Service mogelijk is. Tegenhanger is *Hop-by-Hop routing*. Elk Internet routing protocol is daar tegenwoordig het type van.

4. *Pre-computations* en *on-demand routing* spreken voor zich.

Routing Algorithmen (MH6)

In `MH6` worden Dijkstra's algoritme en het Bellman-Ford algorithme besproken voor het vinden van kortste paden. In `T` worden de routing algoritmen meer op soort besproken. Kortste pad algoritmen steunen op (1) dat een subsectie van het korste pad ook het korste pad is en (2) het relaxation principe.

Dijkstra's algoritme is het bekendste voorbeeld voor een *shortest path algorithm*. De kortste weg wordt afgewogen in afstand (hops) en op een greedy, recursieve manier berekend. Dijkstra's algoritme pakt vanaf de start node een onbezochte vertex met de laagste afstand en berekend via deze node de afstand naar iedere buur van deze vertex. Indien er op deze manier een kleinere afstand naar vertices worden gevonden, wordt deze afstand geupdate. Elke bezochte vertex wordt gemarkeerd (en dus éénmaal bekijken). Een voorbeeld van het algoritme vind je op `M153`.

Dijkstra's algoritme kan gebruikt worden om een 'shortest path tree' te creëren met de start node als root. De complexiteit van het algoritme is $O(N^2)$ omdat het vanaf de start node elke node langs loopt.

Het **Bellman-Ford** algoritme valt in de *distance vector routing* familie. In tegenstelling tot Dijkstra, is bij Bellman-Ford negatieve linkweights wel toegestaan, zolang er maar geen cycles van een negatieve lengte gevormd kunnen worden. *Note* wel (`T`) dat een DVR algoritme last heeft van o.a. het 'count-to-infinity' probleem, omdat een node niet weet of zijn route langs zichzelf loopt. RIP gebruikte Bellman-Ford's algoritme.

Bellman-Ford zou je kunnen beschrijven als een algoritme met een breath-first aanpak. Het algoritme bestaat uit max $h = N - 1$ ronden, waarbij de nodes op afstand (hops, weight niet meegenomen) h van de source node bekeken worden. In elke stap worden bij al deze nodes de links bekeken en de zogenaamde *relaxation* stap uitgevoerd; ofwel de kortste afstand tot de source node berekend en opgeslagen. Een voorbeeld van het algoritme staat op `M156`.

Het volgt dat Bellman-Ford sneller convergeert dan Dijkstra, maar de complexiteit geldt $O(NL) = O(N^3)$. Ook kan het algoritme uitgevoerd kan worden op lexicografische volgorde.

Naast deze twee algoritme heb je de *Link State Routing* (gebruikt in IS-IS (*Intermediate System-Intermediate System*) en OSPF) en *Hierarchical Routing* families. Het

eerste is een verbetering van DVR. D.m.v. het verzenden van packets wisselen routers afstandinformatie uit waarmee Dijkstra's truc gedaan kan worden. Van de packets worden sequencenummers en leeftijd bijgehouden om zo alles up to date te houden. Meer details vind je op `T392`.

Hierarchical routing, wat gebruikt wordt in het telefoonnetwerk, is opdelen van de topology en dan splitsen tussen 'naar de routergroep' en 'binnen deze groep' routeren. *Kamoun en Kleinrock* (1979) vonden het optimale nummer levels van een N routernetwerk als ($\ln N$), met dan ($e \ln N$) entries per router.

Minimum Spanning Trees

Naast routing algoritmen behandelt `M` Prim's algorithme en Kruskal's algoritme voor het vinden van een Minimum Spanning Tree. Een MST bestaat uit exact $N - 1$ links en wordt gebruikt om topology informatie te 'flooden' of om informatie te broadcasten.

In **Kruskal**'s algoritme, welke greedy werkt, worden alle links gesorteerd op weight en van lage tot hoge weight toegevoegd aan de MST om een verzameling van bomen (*forest*) te maken en deze uiteindelijk tot één enkele boom te vormen. Een link wordt alleen aan de MST toegevoegd als het een nieuwe boom oplevert, een boom vergroot of twee bomen verbindt. Indien de twee endpoints van een link al in dezelfde boom zitten, wordt de link dus niet toegevoegd.

Prim's algoritme gaat uit van een enkele boom die wordt uitgebreid, tegenover het samenvoegen van verschillende trees in Kruskal's algoritme. Prim's algoritme kan vergeleken worden met Dijkstra, waarbij de boom steeds uitgebreid wordt op het punt dat op dat moment de kleinste vergroting van de boom oplevert. Ook in dit algoritme wordt de relaxation stap uitgevoerd. Een voorbeeld is te vinden op `M169`.

Routing Protocols (`MH7`)

Basis principes die dichtbij routing protocols liggen zijn **flooding** en **topology changes**. Flooding is in zijn aard een zeer gemakkelijk principe. Het wordt gebruikt om berichten overal te krijgen, zoals routinginformatie of informatie over de topology changes. Ondanks dat flooding altijd het kortste pad kiest (en beloopt namelijk alle mogelijke paden), kost dit veel bandbreedte. Een verbeterde vorm is *Selective Flooding*. Hier wordt informatie gedistribueerd via de 'tree' van een netwerk.

Topology changes zijn er in twee soorten. Uit `M` volgt: **First kind** topology changes are due to failures and joins/leaves o nodes, which are slow variation on the time scale. These changes are all significant to start flooding. **Second kind** topology changes are rapid variations on the time scale and are coupled to the state of resources (thus, fluctuations on link weights). these changes are strongly related by the traffic load of the network and may be influenced by the flooding convergence time T.

Er zijn twee soorten protocols, intra-domain ofwel interior gateway protocols en inter-domain owel exterior gateway protocols. Bij de eerste vindt de routing plaats binnen eenzelfde *autonomous system*, waar bij de tweede verschillende autonomous systems

verbonden zijn en de routing tussen deze systemen plaatsvindt.

Open shortest path first (OSPF) is een intra-domain protocol. Zijn voorganger is **RIP** (Routing Information Protocol), een distance vector protocol dat gebruik maakt van Bellman-Ford in een gedistribueerde, asynchrone operatie. RIP kent het Bouncing Effect en Count-to-Infinity probleem. Eigenschappen van OSPF zijn:

- Het grote doel van OSPF is snelle (en consequent doorgevoerde) updates na topology changes.
- Elke entry in de tabel heeft een sequence nummer zodat oude berichten verwijderd kunnen worden.
- Convergence is stabilized na $N - 1$ hops, de maximale pad lengte.
- OSPF maakt gebruik van Dijkstra's algoritme. Bij meerdere shortest paths kan een router de load balanceren over deze routes.
- OSPF is een link state routing protocol, dat gebruik maakt van een synchronisatie procedure met secure flooding (o.a. hop-by-hop acknowledgements) om snelle en loop-vrije convergentie van de link state database te krijgen.

Het **Border Gateway Protocol** (BGP) is een exterior gateway protocol:

- BGP is het basis Internet protocol. Omdat het Internet in feite een verzameling van ASes is, vormt BGP de lijm tussen ASes (en is lekker complex).
- Wat er binnen een AS voor kortste pad wordt genomen is onzichtbaar.
- BGP is een distance vector protocol met path vectors, dit is het gehele pad in de vorm van een lijst met alle ASes.
- De grote routing tables worden uigewisseld via TCP.
- BGP werkt distributed. Het kortste pad wordt met behulp van policy info en link measures in een Node berekend.
- BGP lost het *stable path problem* (een consequentie van verouderde informatie) op.

Broadcast en multicast

Naast de uitgebreid besproken algoritmen en protocols worden ook verschillende oplossingen voor broadcast en multicast besproken. Zo is *Multidestination routing* genoemd als oplossing voor broadcast routing. Hier wordt een lijst van bestemmingen per tak gebruikt om zo een verbetering van flooding voor te stellen. In feite kan multidestination gezien worden als een oplossing waar meerdere packets dezelfde route nemen maar dat er maar één voor hoeft te betalen.

Andere broadcast oplossingen zijn om te werken met reverse path forwarding (i.c.m. een sink tree) of met een spanning tree. Multicast werkt over het algemeen met pruning. Mobile hosts daarentegen werken vaak met tunneling en een *care-of adress*.

Ook besproken is AODV (Ad-hoc On-demand Distance Vector) routing. Dit is een aangepast distance vector algoritme dat wordt neergezet voor iemand gaat zenden en up to date wordt gehouden met periodieke "Hello" broadcast berichten.

Verdiepende informatie

- `ET4389` Complex Network:
  Netwerk / Graph karakteristieken, zoals ook besproken in `MH6`.
- `ET4359` Advanced in Networking:
  Multicast Routing (niet in deze samenvatting uitgewerkt).
- `ET4388` Ad Hoc Networks:
  Ad Hoc Routing Protocols, zoals AODV.

Congestion Control (`TH5.3`)

Om opstopping te voorkomen en de goodput hoog te houden, kan dit gecontroleerd worden op verschillende manieren, hier beschreven van preventief (zoals **netwerk provisioning** (in schaal van maanden)) tot direct reactief:

- **Traffic-aware routing** is het meenemen van 'load' in berekeningen. Hiervoor heb je multipath routing nodig en een schema dat verkeer langzaam aan die andere kant op stuurt zodat we kunnen convergeren (Gallagher 1977).
- **Admission Control** is veel gebruikt in virtuele circuits (connection-oriented), waarbij verkeer doorgelaten wordt via het *leaky bucket* of *token bucket* principe (ook besproken in `MH9.4` als *input control*). In combinatie met traffic-aware routing kunnen er 'toegestane' VC bepaald worden (`T416`).
- **Traffic Throttling** is het ontlopen van congestion door (in zowel datagram als VC netwerken) machines z.s.m. op de hoogte te stellen van problemen. Dit kan met choke packets (als een soort `ACK`) of Explicit Congestion Notifications (ECN). Dit is het markeren van packets op punt van een congestion router. Betere resultaten met choke packets e.d. krijg je bij Hop-by-Hop Backpressure, waar er bij elke hop/router al aan gewerkt wordt 't te verminderen i.p.v. bij aankomst bij de zender. Een voorbeeld is te vinden op `T420`.
- **Load Shedding** is het netjes verwoorden van "Als je het niet aan kan, pleur het weg". Hierbij verschillen tactieken (nieuw is beter dan oud en vice versa). Om te bepalen vanaf wanneer ze dat moeten doen bestaat Random Early Detection. hierbij wordt een queue en threshold gebruikt.

Naast deze verschillende manieren voor congestion control worden er nog punten besproken als *overprovisioning*, *Quality of Service* en het tunneling principe.

Internet and IP (MH4)

Het Internet, begonnen als ARPANET, heeft vanaf het begin al gebruik gemaakt van TCP/IP en is vanuit een klein netwerk via de NSF T-1 backbone (en verder) uitgegroeid tot wat het nu is. Deze backbone heeft gediend als 'seed'; de originele motivatie was dat hiermee supercomputer faciliteiten van een afstand benaderd konden worden. Andere belangrijke achterliggende architecturische principes zijn (1) dat alle netwerken gelijkwaardig zijn, (2) niemand het Internet bezit en (3) het doel *connectivity* is. Bob Metcalf's statement luidt dat *'the value of a network increases with the square of the number of nodes'*, wat zeker op gaat voor het Internet.

De architectuur van Internet

De (hardware) architecture van het Internet (en dus de TCP/IP protocol suite) is de samenwerking tussen switches (behorende bij de DLL en FL) en routers. Het verschil tussen deze twee is dat een router naast de forwarding capaciteit (die switches ook hebben) ook berekeningen kan uitvoeren alsof het een computer is. Dit maakt een router een netwerk element dat internetworking levert, gebaseerd op een gelijk protocol als IP (gebaseerd op RFC 1958).

Zodra je verbonden bent aan een Internet Service Provider (ISP) is deze je network access point (NAP) naar het Internet. Een ISP voert voor een gedeelte service management uit, zoals authenticatie, accounting en authorizatie (AAA), maar ook sessie handling.

Vroeger vormden de backbone routers de bottlenecks, omdat alle grensverkeer via deze routers moest gaan. Inmiddels is de ATM technologie vervangen voor optische technologieën, en nu ligt het probleem in de edge en access routers.

IPv4

De header van IPv4 datagrammen is te vinden op `T457` en/of `M87`. Uitleg volgt op de pagina's erna. Met gebruik van de *total length* (16 bits) field en de *header length* kan de lengte van de data berekend worden, aangezien deze afhangt van de header lengte (die flunctueert i.v.m. de IP opties). De maximale grote (MTU) van een IP packet is $2^{16} - 1 = 65.535$ bytes. Het *procotol* veld kan de waarden 6 (TCP), 17 (UDP) en 1 (ICMP) aannemen voor de drie bekendste protocols.

De header van IPv4 is altijd een meervoud van 32 bits (de opties worden evt. aangevuld door *padding*). Hieruit volgt direct dat de lengte van IP adressen ook 32 bits lang zijn. IP adressen zijn geschreven in zogenaamde dotted decimal notatie. In 128.208.91.15 zijn vier groepen van 8 bits die ook te representeren zijn als 2 hexadecimale getallen per groep. In dit geval krijg je dan 80.D0.5B.0F.

Een IP adres bevat een vast blok voor de netwerk portie, een zogenaamde prefix - een vaste combinatie aan het begin van het adres. In het class-based systeem zijn er vijf verschillende klassen van IP adressen, uitgelegd op `M91`, die hiervan gebruik maken.

De grootte van de prefix wordt weergegeven met een /. Zo heb je met prefix /24 2^8 mogelijke adressen over. Tegelijkertijd zijn er wel 2^{24} /24 prefixen mogelijk (afhankelijk van de klasse en range volgt het echt aantal mogelijke prefixen). Door prefixen op te splitsen kan je verschillende subnetwerken opzetten, dit heet *subnetting*.

Classless Inter-Domain Routing (CIDR) is conceptueel gelijk aan subnetting, behalve dat de class boundaries niet beperken. Het is dan ook een opvolger van het class-based system. Het aggregeert adressen onder een bepaalde prefix, soms supernetting genoemd. Deze aggregatie is belangrijk voor border gateway of core routers, omdat CIDR helpt het aantal adressen in de routing entries te verminderen. CIDR bepaalt, wanneer er in een routertabel twee matchende prefixen gevonden zijn (neem /24 en /20), dat de gedetailleerde (/24) gevolgd wordt.

CIDR behandelt ook het tekort aan IP adressen, doordat het een hogere flexibiliteit en beter gebruik van de address space mogelijk maakt.

Naast CIDR wordt ook NAT toegepast om het tekort aan adressen tijdelijk op te lossen. In NAT (Network Adress Translation) krijgt elk huis/kantoor een of een paar adressen als shared IP adress. Dit adres is publiek. Een gevolg van NAT is dat voor een server vele computers hetzelfde IP hebben. Onderlinge informatie wordt met unieke adressen uitgewisseld, waardoor er dus in principe IP adressen dubbel worden gebruikt in de wereld, maar deze zullen elkaar nooit 'tegenkomen'. Een uniek internet adres gebruikt binnen een groep mag nooit op het Internet verschijnen. NAT werkt i.c.m. de port behorende bij TCP.

In `M` wordt ook ARP en DHCP (Dynamic Host Configuration Protocol) genoemd. Een MAC adres is uniek en verbonden aan de network interface card. ARP vertaalt een IP adres naar een MAC adres. Hiernaast wordt DNS uitgelegd, maar dat is hier verbonden aan de transport laag. Het ARP protocol behoort officieel tot de DLL laag.

IPv6

IPv6 kent een header die versimpeld is ten opzichte van IPv4. De opdeling van fields is te vinden op `T476` en/of `M98`, met de verschillen hiertussen uitgelicht een pagina later. De drie versimpelingen zijn als volgt:

1. Er is een vaste lengte aan de headers gegeven, waardoor optionele elementen niet mogelijk zijn (en de header length field niet nodig is). In plaats daarvan wordt gewerkt met *extension headers*.

2. Er is geen *header checksum* meer. Dit reduceert processing costs. Error detection/correction wordt al opgelost door lagere lagen.

3. De hop-by-hop fragmentations fields zijn weggelaten. In IPv6 wordt er een 'unit of transmission' bepaald die over elke link past waardoor fragmentatie vervalt. Dit wordt gedaan met *path MTU discovery*.

Het enige 'nieuwe' field is de *flow label*. Deze wordt gebruikt door reeksen of packeten een bepaald label te geven om een bepaalde QoS-klasse aan te geven.

Echter, de nadruk ligt op de adressen. IPv6 lost namelijk in één klap het tekort aan IP adressen op omdat het werkt met adressen die 128 bits lang zijn (4×32). De notatie van deze adressen gebeurd met 8 hexadecimale getallen die een 16 bit nummer representeren. Als voorbeeld hebben we 1080:0:0:0:0:0:200C:417A, welke ingekort kan worden tot 1080::200C:417A. In IPv6 wordt CIDR op eenzelfde manier toegepast als in IPv4, en is de syntax dat / de prefix-lengte aangeeft.

Andere Protocols (`MH4.7`)

In de TCP/IP suite kan men gebruik maken van andere protocols, zoals `ICMP`, `Ping` of `Traceroute`. Een Internet Control Message Protocol bericht wordt ge-encapsuleerd in een IP packet en wordt gebruikt om de bron (als genoemd in het IP packet) te informeren wanneer er een fout is opgetreden bij een bepaalde router - bijvoorbeeld wanneer een bestemming niet bereikbaar is.

Traceroute maakt gebruik van ICMP om een sequence van IP adressen te vinden die het pad naar de bestemming voorsteld. Dit wordt gedaan door de TTL (time to live) vanaf 0 tot voldoende groot te vergroten en steeds de router die het ICMP bericht noemt op te slaan in de sequence.

Traceroute is een van de beste path measurement tool die we tot nu toe hebben, maar geeft geen reliable informatie, omdat de route kan veranderen door topology updates waardoor de uiteindelijke sequence niet meer klopt. Daarnaast is reverse route niet (altijd) gelijk aan de orginele route.

Verdiepende informatie

- `ET4359` Advanced in Networking:
  Bespreking van ATM

Transport Laag (H6)

De Transport Laag maakt zich druk over reliable, (cost-)efficiënte data transmissie service voor de users, normaliter applicatieprocessen. Deze laag maakt zich dan ook druk om het maken van verbindingen (tussen ports) en werkt met segmenten. Opgeteld is het dus zo dat het segment (met header en payload) ingepakt zit in een packet (waardoor er een packetheader bijkomt), welke weer ingepakt zit in een frame (waardoor er een framehader bijkomt).

De twee belangrijkste protocols in de transport laag zijn UDP en TCP. TCP verzorgt de Flow Control in het Internet. Binnen deze protocols wordt duidelijk dat we steeds dichter op de applicatie komen. Namelijk, het IP protocol levert slechts een IP packet van bron naar bestemming, maar weet niet hoe deze specifieke data afgeleverd moet worden bij een applicatie op deze bestemming. UDP levert een *unreliable connectionless* packet service, terwijl TCP *reliable connection-oriented* transport over de unreliable IP service aanbiedt.

Koppeling tussen de netwerk en transport laag

Wanneer data een machine binnenkomt via de netwerkadressen (NSAP), wordt de data doorgestuurd naar de tranposrt laag. Via een TSAP (*Transport Service Access Points*) (of Port) duidelijk naar welke proces/applicatie deze data moet. Dat vele applicaties geen vaste TSAP hebben, wordt opgelost met een portmapper of het *initial connection protocol*. Een verbinding verloopt in dat geval via een process server die op zijn beurt een port klaarmaakt met het juiste TSAP adres (`T529`).

De transportlaag is verantwoordelijk voor het maken, in stand houden en afbreken van verbindingen. Omdat deze informatie verwisseld wordt in packets en er van alles met deze packets kan gebeuren wordt als volgt gewerkt:

- Het systeem wordt afgebakend zodat er een grens aan geldige sequentienummers zijn voor men weer bij 0 begint te tellen. Het syteem wordt hier op aangepast zodat er een ongeldige regio van sequencenummers onstaat. Hierdoor kan er afgedwongen worden dat nummers niet te snel of te langzaam stijgen (`T533`).
- Een goede verbinding wordt opgezet met een “three-way-handschake”.
- Een verbinding verbreken heeft te maken met het “two-army” probleem en gebruikt ook een handshake, acknowledgements en timers (`T539`).

Congestion control is de gezamelijke verantwoordelijkheid van de netwerk- en de transportlaag, waarbij de netwerklaag zorgt dat de routers niet vollopen terwijl de transportlaag zich bezighoudt met gebruikers en bandwidth allocation. Een *faire* verdeling wordt berekend via de max-min fairness, waarbij geldt dat het verhogen van de bandbreedte voor een bepaalde flow de situatie voor de rest alleen maar slechter maakt (voorbeeld op `T551`). Een tweede punt is dat de verkleining van bandbreedte alleen werkt als de andere flows 'aan' staan, dat is wel zo eerlijk.

Ook het reguleren van de sending rate draagt bij aan congestion control. TCP gebruikt het congestie feedback systeem AIMD (Additive Increase Multiplicative Decrease). Geometrisch gezien is AI met 45° en MD met de lijn naar de oorsprong gericht. Dit zullen we hieronder terug vinden.

UDP: Het User Datagram Protocol

UDP (User Datagram Protocol) is hét protocol voor applicaties om ingekapselde IP datagrammen te verzenden zonder een verbinding te maken (ofwel het is een connectionless service). UDP biedt IP twee additionele services: *error checking* en *applicatie distinctie*. Het verzorgt dus geen flow control, congestion control of retransmissies.

De simpele header wordt gebruikt om de additionele services uit te voeren. De Port-nummers (beide 16-bits) zorgen voor de applicatie distinctie. De UDP Length en checksum samen met de IPv4 pseudoheader (protocol = 17) wordt gebruikt voor error checking.

Remote Procedure Call (`TH6.4.2`)

Al het programmeerwerk en verstoppen van lastige zaken in UDP is gedaan via Remorte Procedure Call (RPC). Client-Server RPC is een van de grote gebruikers van UDP. Daarnaast heb je het Real-time Transport Protocol, die via een socket-interface UDP gebruikt voor multimedia applicaties (`T565`). Dit werkt o.a. met timestamping om network delay variaties te voorkomen, maar kan ook verschillende stromen met elkaar synchroniseren. Het zusje RTCP (C voor control) verzorgt de feedback, synchronisatie en interface en gaat dus ook aan de slag met de issue wanneer men moet afspelen als men buffert en last heeft van *jitter* (variatie in delay van packets).

TCP: Het Transport Control Protocol (`MH5`)

TCP is de main workhorse voor Internet. TCP connections zijn full duplex en point-to-point en kunnen dus niet gebruikt worden voor multi- en broadcasting. *Sockets* zijn een combinatie van IP adres en portnummer (16 bit) om een machine te identificeren. Onder deze 1024 ports zijn zogenaamde privilege of well-know ports, zoals 25 (e-mail) en 80 (WWW). Een transport laag 'flow' is in de TCP/IP suite als een quintuple voor te stellen: twee IP adressen, twee ports (ofwel twee sockets) en het protocol veld dat UDP of TCP aangeeft.

Header

De TCP header (20 byte groot zonder opties) is te vinden op `T575, M113`. Het protocol werkt met 32-bit *sequencenummers*. Deze geven aan welk *octet* (want daar werkt TCP mee) de eerste is in de payload. Bijvoorbeeld, hebben we als sequencenummer 1001 en de TCP data bevat 8 octets, dan is het volgende TCP segment met sequencenummer 1009. Het *acknowledgementnummer* is het sequencenummer van het volgende (op volgorde) verwachte octet net als in SR ARQ protocols.

Meest belangrijke feature van TCP is de flow control, gebaseerd op sliding window techniek. Hiervoor speelt de *window size* een rol. Daarnaast beslist de software over de segmentgrootte, in combinatie met MTU discovery. Informatie hierover wordt gegeven in het additionele *option field*. De mogelijke flags zijn `URG, ACK, PSH, RST, SYN, FIN`. Deze spelen een signalling rol.

Verbindingen

TCP maakt verbindingen met een zogenaamde *three-way-handshake*. In de handshake worden de flags `SYN, ACK` gebruikt. Wanneer twee apparaten tegelijkertijd een handshake initiëren, levert dit ook een enkele verbinding op, doordat het apparaat dat een late `SYN` binnenkrijgt deze vraag wegwuift met een `RST` flag respons.

Verbindingen worden verbroken door een abort via de `RST` flag (net als hierboven) of via een *graceful close* (`M117`). Met de `FIN` flag geeft een apparaat A aan dat hij klaar is met het verzenden van data en de verbinding vanaf zijn kant wil sluiten. Een apparaat B kan dan nog wel data versturen en stuurt op zijn buurt een eigen `FIN`. Na zo'n bericht wordt er wel een `ACK` verwacht.

Self-clocking (`MH5.4`)

Op `M119` wordt er gesproken over het *bandwidth-delay product*, de bandbreedte maal een one way transit, zoals ik ook al over sprak bij de introductie van de sliding window protocols. Extra punten die worden genoemd zijn dat voor de one way transit - ook wel *round-trip time* in de context van `M` - door de asymetrie van de paden en traffic load de end-to-end delay op de heenweg anders is dan van de terugweg. Daarnaast geldt het inzicht dat in het netwerk het aantal bits niet kan veranderen, waardoor een packet dus 'spreidt' in tijd wanneer een link een lagere rate heeft.

TCP is self-clocking. Dit houdt in dat het systeem zich automatisch aanpast op de beschikbare capaciteit (ofwel laagste linke rate). Er wordt voldaan aan de conservatie van packets, waar pas een nieuw packet verstuurd mag worden als een oud packet weg is (ofwel is acknowledged). Gegeven is dat een ontvanger niet sneller `ACK`'s kan genereren dan de packets door het netwerk reizen.

Ondanks dat self-clocking systemen zichzelf stabiel houden, zijn er twee nadelen. Allereerst is het opstarten van het systeem ingewikkeld, aangezien er `ACK`'s nodig zijn om de 'clock' te laten lopen. Daarnaast bouwen queues zich op bij bottlenecks wanneer we nog in de opstartfase zitten. Deze queues verdwijnen alleen indien de zender stopt met het versturen van (nieuwe) packets.

Flow Control

Om flow control te realiseren begruikt TCP verschillende technieken, namelijke de 'slow-start' in combinatie met congestion avoidance, retransmissies, fast-recovery en *additive increase, multiplicative decrease* (AIMD).

Slow-start wordt toegepast direct na de verbinding is bewerkstelligd, omdat een directe burst van data buffer overflow en dus packet loss en queues oplevert. In dit algoritme heeft de zender naast een advertised window (welke maximaal mogelijke grootte heeft) ook een congestion window $W = 1$. De zender volgt altijd de kleinste van de twee. In de 'slow-start' procedure verdubbelt W zich elke RTT. Ofwel, er is een exponentiele groei gereguleerd door `ACK`'s. Nadat een threshold (sstresh) is bereikt naderen we de congestion avoidance, waar $W = W + 1$ elke RTT, totdat de waarde van de advertised window bereikt is.

Note van `M121`: *Since the RTT is not a control parameter because TCP's self-clocking mechanism is based on acks, the rule is to increase W by $1/W$ on the receipt of a new ack. Indeed, a window of size W will generate at most W acks in one RTT and an increment of $1/W$ per ack will incrase the window by at most 1 packet per RTT.*

Een collision avoidance strategy moet voldoen aan twee eisen. Allereerst moeten de endpoints geïnformeerd worden over collisions. TCP gebruikt packet loss (time-outs) als een signaal dat de TCP pipe is congested of over-crowded.

Ten tweede moet er een regel zijn om de sending rate aan te passen wanneer een collision notificatie is geweest. Hiervoor wordt AIMD gebruikt, in feite het resultaat van de slow-start aanpak en fast recovery. Retransmissies worden geinitieerd door een RTO (retransmission time-out) of wanneer er drie duplicate acks ontvangen zijn. Na dit tweede volgt een *fast* retransmissie, gevolgd door fast recovery (`MH5.8`). Het is in deze fase waar de halvering van de congestion window W plaatsvindt. Bij een RTO volgt $W = 1$ en begint er een nieuwe 'slow-start' procedure.

AIMD leidt tot fairness tussen TCP bronnen door verdeling van beschikbare capaciteit. De grootte van W volgt een sawtooth figuur. Daarnaast kan de werking van alle besproken mechanismen hierboven als volgt afgebeeld worden.
Note: M5.8 - M5.11 volgt nog meer diepgaande informatie over TCP.

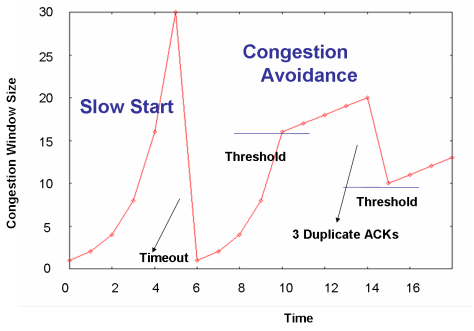

Figure 4: De grootte van de window size W gedurende verschillende fasen in TCP

Applicatie Laag (TH7)

In dit hoofdstuk bespreken we een aantal applicaties die werken door het bestaan van het Internet: DNS, e-mail en het World Wide Web.

DNS

Het Domain Name System (DNS) is er omdat het veel makkelijker is om `www.sitenaam.nl` te onthouden dan 128.0.2.54. Het systeem wordt net als IP beheert door ICANN en kent verschillende onderverdelingen, gescheiden door punten.

In eerste instantie kreeg ieder land zijn eigen domein (zoals `.nl`), maar het land `.tv` verkocht zijn domein. Kopen en doorverkopen heet *cybersquatting*.

Alle informatie wordt opgeslagen als een set resource records, waarvan verschillende types bestaan. Een A-type is bijvoorbeeld een IPv4 adres. Het type PTR (alias voor IP-adres) is er om reverse look-ups mogelijk te maken. Een DNS overzicht voorbeeld vind je op `T636`.

Om het systeem niet te laten overstromen is de wereld opgedeeld in zones en komt een computer via verschillende servers aan het antwoord. Deze procedure heet een *recursieve query*. De local server die alles nacheckt in deze query heet weer een *iteratieve query*.

E-mail

Kortgezegd zit e-mail als volgt in elkaar: Al je mail (je mailbox) is opgeslagen op de server van de Message Transfer Agent (zoals `google`); `gmail.com` bezoeken is simpelweg gebruikmaken van de interface. Hoe deze MTA deze in de juiste mailbox zet gaat met het *simple mail transfer protocol* (SMTP).

Jij werkt vanaf de User agent waar je verzending 'mail submission' heet. In het boek wordt Gmail genoemd als User Agent, maar bij het versturen heb je daar in feite niet meer mee te maken. De interface zorgt ervoor dat alles voor je wordt ingevuld (op de juiste plek) zodat er gelijk een goed pakket van gemaakt kan worden. Wat de ontvanger ermee doet heet message disposition.

Wat wel goed is om te weten is dat een appropiate vorm van een e-mailadres in de vorm `user@DNS-adress`. In het DNS moet dan dus wel een entry bestaan waarin staat dat ons adres daarwerkelijk ook met mail om kan gaan. Zoiets wordt dan in de message header gestopt. De te verzenden formats kan via de Internet Message Format of MIME.

MIME houdt ook rekening met non-ASCII mailtjes en voegt weer extra headers toe aan het mailtje. MIME staat voor multipurpose Internet Mail Extensions. MIME heeft vijf verschillende encodingsschema's en kan verschillende types aan, zie `T653`.

Het verzende van de mail, gedaan door SMTP, bestaat uit twee delen: mail submission en mail verzending. Bij het eerste vertrekt de mail van user agent naar de MTA. SMTP werkt zonder authenticatie (het invullen van FROM)m, dus vaak werkt men met Extended SMTP (`T658`). De client en server praten heen en weer en de client geeft eerst zijn header (hierin staat voor wie de mail is en wat er in staat) en vervolgens de tekst. De te gebruiken extensie is AUTH. Na de mail submissie stuurt de MTA hem via SMTP door naar het eindadres. Dit gebeurt via een samenwerking met DNS. Het ophalen van de mail uit mailboxes gaat via IMAP, Pop3 en webmail.

WWW

Het WWW (met o.a. fireforx als interface) werkt met URL's (Uniform Resource Locators. Een URL bestaat uit de volgende onderdelen:

1. Het protocol (of *scheme*)
2. De DNS naam van de machine waar de pagina op staat
3. Het unieke pagina pad om de pagina te vinden.

Een voorbeeld van een url is: `http:`[1]`//cs.washington.edu`[2]`/index.html`[3].

Via de server wordt, gegeven een URL, via TCP connecties de pagina opgehaald. Naast URL's bestaan er ook URI's (identifiers)., waardoor je een pagina kan openen zonder dat je weet waar vandaan. Dit is handig voor pagina's die veel verschillende kopieën op servers hebben. `Google` is daar vast een voorbeeld van.

Aanvullende informatie die een server meegeeft over een pagina is bijvoorbeeld het MIME type. Het kan zijn dat de browser het MIME type niet herkent (text/html natuurlijk wel), waarna de browser twee opties heeft: Een plugin of helper-applicatie. Het runnen van een plugin gebeurt binnen de browser, waardoor deze bereik heeft tot de pagina en dus de appaerance kan aanpassen. Indien een helper-applicatie nodig is opent de computer deze buiten de browsers. In dit geval opent de computer bijvoorbeeld Word of Excel.

De server werkt in eerste instantie hoe je verwacht, maar om de efficiëntie op te krikken worden servers multithreaded gemaakt. Op deze manier kunnen ze meerdere processen tegelijkertijd aan (`T674`). Daarnaast werken ze met caches, het tijdelijk opslaan van informatie om laden sneller te maken.

User informatie wordt bijgehouden met behulp van cookies. Een *nonpersistent* cookie verdwijnt als je de browser afsluit, persistent cookies hebben een *expire date/-time*. een cookie verwijderen kan doordat de server dezelfde cookie opnieuw stuur met een expire date in het verleden. Een voorbeeld van een cookie is het stoppen van artikelen in een winkel wagentje.

Statische webpagina's worden gecreëerd door HTML, tegenwoordig al versie 5.0 (features op `T685`), daarnaast wordt CSS (Cascading Style Sheets) als style sheet voor

de pagina's gebruikt. Tegenwoordig zijn dynamische pagina's (hele programma's via het web!) meer in trek. Gebruikers hoeven geen applicaties te installeren en opgeslagen informatie is overal op de wereld te bereiken - dit is ook bekend als *cloud computing*. Voor dynamische pagina's is PHP populair (server-side) of javascript/dynamisc html (client-side), wat overigens niets met Java te maken heeft.

Het HyperText Transfer Protocol (HTTP) is een simpel request-response protocol dat normaliter over TCP gaat, maar het is wel een echt protocol voor de applicatielaag. Het hergebruikt TCP connecties voor meerdere requests (heet persistent connections of reuse). Hierbij zullen na de setup responses steeds sneller komen door de "slow start" van de TCP congestion control afdeling. Daarnaast kun je ook pipelinen. HTTP kan meer dan alleen pagina's laden, want het heeft methodes, zoals "head", waarbij alleen de pagina's header wordt gelezen. Deze methodes (of requests) kunnen worden uitgebreid met request of response headers, waarmee additionele informatie kan worden uitgewisseld.

Nogmaals, http gebruikt caching.

Media Applicaties

voor het streamen van audio (en wideo) wordt ook het converten en sampelen genoemd. Daarnaast zijn er encoding en decoding algoritmen voor compressie. Wanneer de gedecodeerde output verschilt van de geëncodeerde input, is het systeem *lossy*, wanneer dit identiek is, is het syteem *lossless*. Voor de lossy systemen geldt dat een kleine toelating van informatieverlies veel voor je compressie ratio kan betekenen! Voorbeelden van compressie algoritmen zijn MP3, AAC en MP4. Compressie kan via waveform coding (is 'letterlijk') of via perceptual coding. Hierbij worden maskers gebruikt om sommige tonen uit je gehoor te bannen.

Bij video ligt het ingewikkelder, omdat er nu ook dingen geregeld moeten worden voor de pixels. Het boek verteld een hele historie en legt de JPEG standaard helemaal uit. Bij MPEG, ook uitgelegd in het boek, werkt men met drie soorten frames (`T729-730`):

1. Intracoded (still) frames
2. Predictive frames, ofwel frames die nét iets verschillen, maar waarvan niet alles verstuurd hoeft te worden
3. Bidirectional frames, waarbij ook naar toekomstige frames worddt gekeken.

Het streamen van video is makkelijker via een media server dan live. Om om te gaan met packet loss in de applicatie kan je werken met erasure of interleaving. Bij erasure wordt een pakket $P = A + B + C$ meegestuurd zodat een verloren pakket berekend kan worden en vervolgens kan worden herzonden. Bij interleaving wordt een pakket verzonden met de oneven msec en een pakket met de even msec, zodat er bij verlies niet direct iets gemist wordt.

Daarnaast wordt er gewerkt met een buffer, zoiets als UDP-RTP.

Security & Cryptography (TH8)

Verdiepende informatie

- ET4191 Security and Cryptography:
  Behandeld o.a. DES, Rijndael en Authenticatie in details.

Cryptography Algoritmen

In de basis van cryptografie wordt het verschil gelegd tussen een *cipher* en *code*. Codes, waarbij elk woord wordt vervangen door een woord of symbool, worden tegenwoordig niet meer gebruikt. Bij ciphers wordt een plaintext d.m.v. een key in een bit-for-bit/character-for-character stappenplan omgezet naar ciphertext. Het algemene model, met intruders meegenomen, vind je op T785. Er geldt $D_K(E_K(P)) = P$. Kerckhoff's principe zegt dat je het algoritme publiek moet houden, maar de sleutel geheim. *Security by obscurity* werkt niet. Hoe moeilijk het is om de sleutel te vinden verhoudt zich exponentieel met de sleutellengte.

In de basis zijn er substitutie en transpositie cyphers. De eerste werd al door Julius Caesar gebruikt. Letter voor letter substitutie heet een monoalphabetic substitutie cipher en kan tegenwoordig door naar di- en trigrams en statistieken te kijken simpel worden gevonden. Bij de transpositie cypher wordt de tekst opgedeeld en de kolommen achter elkaar gezet in de volgorde van de sleutel. Ook deze methode kan gebroken worden, als de analyst bekende woorden weet en afstanden tussen letters kan traceren.

Wat onbreekbaar is, is de One-Time Pad, waarbij de tekst wordt geXORed met een onetimelijn. Echter, het versturen van zo'n pad is lastig.

Principe 1: Berichten moeten redundantie kennen.
Principe 2: Er moet een methode zijn om aanvallen af te slaan.

Tegenwoordig zijn de cryptografie-algoritmen heel complex. Allereerst heb je symmetric-key algoritmen, met in particular block ciphers, waar een n-bit blok plaintext een n-bit blok ciphertext produceerd. Gelocaliseerd in de software of hardware heb je enerzijds P-boxes, waardoor nummers/bits geshuffelt kunnen worden, of S-boxes, waarbij voor en na de shuffle een decoder en encoder (bv 3:8) zit. Deze twee gecombineerd levert een product cipher.

Een eerste voorbeeld is DES (Data Encryption Standard), maar inmiddels kan dit alweer gekraakt worden. DES werkt met een 56-bit key (T799). Iets sterker is triple-

DES, dat als E-D-E werkt. Indien $K_1 = K_2$ is triple-DES compatibel met duo-DES. Een volgende stap is AES en daarbij Rijndeal.

Omdat DES en AES in feite block ciphers (monoalphabetic/substitute ciphers) zijn met grote characters, geeft elke zelfde invoer een zelfde uitvoer. Om dit te voorkomen zijn er verschillende cipher modes (vanaf p806) zodat het niet mogelijk is de code te misbruiken (zoals in electronic code book mode). Cipher modes zijn een manier van hoe block encoding aan elkaar wordt geplakt. De eerste manier is block chaining mode. Hierbij wordt een stuk plaintekst P_1 geXORed met een Initialisatie Vector en dan pas geëncodeerd, waarna deze uitkomst wordt gebruikt met P_2 in plaats van de IV. Op de terugweg gaat het precies andersom.

Chaning heeft als nadeel dat decriptie pas kan als een heel 64-bit block arriveert. Cipher Feedback mode gebruikt daarom een shift register van de codes (ook wel met een IV). Echter, het nadeel is dat een one bit inversie enorme problemen oplevert (namelijk de volgende x bits als het register breed is).

Stream cipher mode heeft hier geen last van, omdat zij een losse key stream gebruiken die immuun is voor transmissie error (zie one-time pad).

Bij counter mode kan men wél bij elk blok dat hij/zij wil doordat daar de IV wordt hergebruikt.

In 1976 werd voorgesteld dat ook sommige sleutels bekendgemaakt moeten worden. Zo ontstonden public-key algoritmen, met als grote voorbeeld RSA. In kort geldt: $D(E(P)) = P$, waarbij E_K publiek is maar D_K geheim. D is níet af te leiden vanuit E en E is niet te breken met een "chosen plaintext attack".

RSA is eigenlijk te langzaam voor publiek gebruik en wordt nu gebruikt voor het distribueren van one-time session keys voor symmetric-key algoritmen. Precieze uitleg staat op `T813-814`.

Authentication

Een van de belangrijkste gebieden van beveiliging is authenticatie. Allereerst is er de symmetric-key signature, waarbij een Big Brother instantie wordt gebruikt die iedereen zijn geheime sleutel kent om zo zeker te weten dat de zender bestaat en zegt wat hij zegt en deze verificatie doorstuurt naar de ontvanger. Om Big Brother te ontlopen, wat in realtime realistischer is (aangezien Big Brother niet bestaat), zijn er public-key signatures. Dit houdt in dat de zender zijn geheime sleutel indirect meegeeft op de volgende manier $P \to D_A(p) \to E_B(D_A(P)) \to D_B(E_B(D_A(P)))$. Vervolgens is E_A nodig om P te verkrijgen, waarmee bewezen wordt dat het bericht van A komt. In feiten kan élk public-key algoritme hiervoor gebruikt worden, met RSA als reguliere oplossing.

Een derde is Message Digests, een has functie die $MD(P)$ uniek maakt en er dus niet geldt $MD(P') = MD(P)$. Op deze manier is de combinatie P, D_A en $MD(P)$ uniek.

Het moeilijke zit hem in het verkrijgen van public-keys. Immers, als ze van een homepage komen, kan er zo iemand de homepage faken en zijn sleutel geven om zo tussen beide te komen. Omdat sleutels via een key distributie centrum ook geen oplossing is (stel, dit centrum heeft een virus), werkt men met certificaten die bewaard worden op een Certification Authority. Deze CA kan checken of de sleutel is van wie claimed dat 'ie van hem is.

Voor de certificaten is X.509 een aangeboden standaar, diens velden op `T827` te vinden zijn. Met versie 3 van X.509 mogen DNS namen gebruikt worden.

In de wereld zijn er meer CA's en is er een heuse Public-Key Infrastructuur, waarvan de hierarchie op `T829` te vinden is. Via de boomstructuur kan een RA certificeren dat een CA echt is *und so weiter*. Een collectie van zulke certificaten heet een "chain of trust" of "certification path". Daarnaast slaan browsers een stuk of 100 keys op, zogenaamde *trust anchors*.

Certificaten kunnen ook worden ingenomen, welke op de Certificate Revocation List komen.

Hier volgen drie voorbeelden van security / authentication toepassingen:

- **IPsec**. Enerzijds is er transport mode hebben, waar er een IPscheader (ofwel een authenticatie header) tussen de TCP en IP header wordt geplaatst. Anderzijds tunnel mode, waar het hele pakket verpakt wordt in een beveilig pakket met een eigen header. Voor de AH uit de transport mode verwijs ik je naar p834.

- **Firewalls**. Ook dit is een methode om 'good' bits door te laten en de 'bad' bits af te weren, welke werkt als een packet filteren. De inhoud van packets wordt bekeken of deze voldoet aan gestelde criteria.

- **VPN's** (virtual private networks). Deze kunnen d.m.v. kabels en tunnels worden opgezet, dus ook via het internet.

Authentication (met ook de sleutels) protocols zorgen ervoor dat na een reeks berichten A en B zeker zijn van elkaar's identiteit. Veel authenticatie protocols zijn zogenaamde "challenge-respone" protocols. Bij een gedeelde geheime sleutel kan een challenge R_A beantwoord worden met $K_{AB}(R_A)$ waarbij K_{AB} de zender B identificeerd. Dit gebeurt in vijf stappen (`T847`). Minder moet je niet doen, want dan kan Trudy haar reflection attack uitvoeren als er teveel info tegelijkertijd wordt verzonden. Een ander goed voorbeeld is HMAC (wat ook gebruikt wordt in IPsec).

Bij *Diffie-Hellman* wordt er een shared key gemaakt door de uitwisseling van n (een priem, net als $(n-1)/2$), g, $g^x \bmod n$ tegenover $g_y \bmod n$ met de gedeelde sleutel $g^{xy} \bmod n$. Helaas is deze manier niet bestand tegen de "man-in-the-middle attack".

Een volgende stap is een key distributie centrum te raadplegen, waarbij de simpelste vorm nog kwetsbaar is voor een response attack. Het Needham-Schroeder authentication protocol (1978) maakt gebruik van een multiway challenge-response, met de zwakte dat Trudy een oude session key kan misbruiken. Dit probleem is opgelost in 1987; Otway en Rees deden het op een kortere manier (`T856`). Als laatst wordt public-key cryptography genoemd op `T859` en dat werkt!

Andere Security

Voor e-mailbeveiliging wordt Pretty Good Privacy (PGP) van Zimmerman (1995) gebruikt, die dat gratis ter gebruik heeft gesteld. Het systeem dat op `T862` wordt getoond is heel belangrijk. PGP gebruikt voor encryptie de block cipher IDEA (Int. Data Encryption Algorithm) met 128-bit keys, RSA voor key management en MD5 voor data integrity. Voor het RSA-gedeelte zijn 4 sleutellengtes in gebruik: 384, 512, 1024, 2048 bits, met de laatste twee onbreekbaar.

Het resultaat van een bericht kan gevonden worden op p863. Het bericht en de signature is encrypted met IDEA en deze en het message key onderdeel krijgt Base64 over zicht heen. PGP ondersteunt naast zijn eigen key ringen ook X.509.

Het web wordt dagelijks geterroriseerd, zo ook DNS met DNS spoofing, waarbij de DNS server wordt misleidt een vals IP adres op te slaan. Tegenwoordig is er DNSsec, met services waar de data vandaan kwam, public key distributie en transactie/request authenticatie. De DNS records worden opgeslagen in Resource Records Sets. Deze sets groeperen de records en zorgen dat deze getekend zijn door de eigenaar.

Een laatste voorbeeld is Secure Sockets Layer, om zo het maken van verbindingen ook veilig te maken. HTTP over SSL heet HTTPS (`T873`).