Samenvatting Infrastructure

Noeël Moeskops

November 4, 2017

Contents

0 Voorwoord

0.1 inleiding

Deze samenvatting heb ik geschreven voor het tentamen 2017 blok 1 infrastructure. De hoofdstukken die hier beschreven zijn, zijn ook volgends de *reading guide infrastructe* van het HvA:

- Week 1: network architecture physical layer
  - KR 1.1 - 1.6: study with the exception of paragraph 1.3.2
  - KR 1.7: read this chapter to get an insight into the development of the Internet. You don't need to memorise the dates, events or people mentioned.
- Week 2: application layer application protocols
  - KR 2.1, 2.2, 2.3
  - KR 2.5: study with the following exception: You don't need to memorise the calculations in the section 'Scalability of P2P Architectures'. However, you should be able to compare the distribution time for a P2P architecture to a client-server architecture as shown in figure 2.23.
- Week 3: DNS transport layer
  - KR 2.4
  - KR 3.1, 3.2, 3.3, 3.5.1, 3.5.2, 3.5.6
- Week 4: network layer
  - KR 4.1
  - KR 4.2 - 4.2.4: study with the following exceptions: o For 4.2.3, you need to understand the role of the switching fabric in a router, but you don't need to memorise the specific properties of the different technologies that are described (memory, bus, interconnection network). o For 4.2.4, study only the first part (until 'input queueing') that describes where packet loss may occur in a router.
  - KR 4.3, 5.6
- Week 5: subnetting and link layer
  - Study the materials available on the VLO about subnetting
  - KR 6.1: study in its entirety
  - KR 6.2: study with the following exceptions: You should be able to describe what CRC is and how it is used, but you don't need to be able to do the calculations described in this chapter.

- Week 6
  - KR 8.1
  - KR 8.2 - 8.2.1: study the first part until (but not including) 'block cyphers'
  - KR 8.2.2: study the first part until (but not including) 'RSA'
  - KR 8.3
  - KR 8.6: study the first part until (but not including) 8.6.1
  - KR 8.7 - 8.7.1
  - KR 8.9

0.2 bijdragen

Dit document is geschreven door Noeël Moeskops. Met taal verbeteringen op Hoofdstuk 3 door Thomas Wiggers en Hoofdstuk 5 en 6 zijn ook door Thomas Wiggers geschreven.
Het orginele document kan je vinden op github: *https://github.com/Noedel-Man/infrastructure*

0.3 oeventoetsen

De vragen en antwoorden van de oeven toetsen.

0.3.1 vragen

week 1 Ontbreekt :(

week 2

1. Suppose Host A wants to send a large file to Host B. The path from Host A to Host B passes through three links, of rates R1 = 4 Mbps, R2 = 2 Mbps, and R3 = 1 Mbps.

   Assuming no other traffic in the network, what is the throughput for the file transfer?

2. HTTP is a protocol at which layer of the TCP/IP protocol stack?

3. SMTP-messages will result in message(s) in the transport-layer using which protocol?

4. What is the network application architecture model used by e-mail?

5. The transport layer takes care of transmitting data for different applications that run simultaneously on a host. (For instance, you may be using your browser to surf the web, while downloading files in the background using Bittorent.) In order to deliver the data to the correct application

process, the transport layer uses port numbers to identify the application process running on the host.

What port number is normally used by a webserver, using the HTTP protocol?

6. When your browser makes an HTTP request, it will specify a request method. What method is normally used to submit data to the webserver (for instance: data entered into a form on the website)?

7. Suppose a company has a relatively slow connection to the Internet, while its employees need to surf the web more and more to find relevant information. The company does not want to upgrade its Internet connection as that would be very expensive. What solution can the company implement to reduce traffic on its access link to the Internet and make sure its employees can surf the web faster?

8. This is a transcript of the beginning of a session in what protocol?

9. What is the oldest application protocol used to get e-mail from a server to a client?

10. kan niet ivm plaatje

week 3

1. Within an organisation, there can be many subdomains. For example, within the HvA we have: www.hva.nl, rooster.hva.nl, sis.hva.nl, webmail.hva.nl, moodle.informatica.hva.nl and the list goes on...

   There are different types of DNS servers. What type of DNS server is the source of information about the IP-addresses for all hostnames within an organisation?

2. DNS servers store resource records. There are different types of DNS records.

   What type of DNS record is used to store a domain and the address of a DNS server that knows the IP addresses for hosts in that domain?

3. In addition to translating hostnames to IP addresses, DNS provides a few other important services, such as host aliasing. Name one more service (other than translating hostnames and host aliasing) that is provided by the DNS protocol.

4. A computer network's transport layer could, in theory, offer many different services to the application layer. Some examples are:

   - Guaranteed minimum bandwidth for data transfer
   - Guaranteed maximum delay for data transfer
   - Reliable data transfer

- Confidential data transfer

In reality, which of these services are offered by the Internet's transport layer?

5. Which transport protocol provides a connectionless service?
6. Which TCP service is described below?

   This service allows the sender to match the rate at which it is sending data against the rate at which the receiving application is reading. In this way, this service enables the sender to avoid overflowing the receiver's buffer.

7. TCP Syn/Ack

   Consider the TCP segments exchanged between hosts A and B in the picture above. What is the Ack number for the communication from host B to host A? (where the red dot is shown)

   The communicated shown above represents a Telnet session, so one keystroke (one byte of data) is being sent at once.

8. The UDP header has only four fields. Name one of them.
9. When starting a Wireshark capture, you will notice that Wireshark records lots of different network activity on your computer. Suppose that you have a capture like shown below and, as we did in the practical lab, only want to see the HTTP traffic that was captured when you visited a website. How do you make Wireshark show only the HTTP traffic from the capture below and hide the traffic from other protocols?

   (a) Type the filter 'http' in the green bar in the area labelled 'A'
   (b) Double click 'HTTP' in the column 'protocol' in the area labelled 'B'
   (c) Click the triangle left of 'Hypertext transfer protocol' to show information about HTTP in the area labelled 'C'
   (d) Search for the text 'HTTP' in the area labelled 'D'

10. During the lab, we used the Telnet protocol to connect to a webserver and request an object.

    What is the correct HTTP request message to request the object: /wiki/Hypertext_Transfer_Protocol from the webserver: nl.wikipedia.org?

    (Keep in mind that during the lab you observed the regular behaviour of the HTTP protocol, just as described in the literature.)

    (a) GET /wiki/Hypertext_Transfer_Protocol
    HTTP: 1.1
    Host: nl.wikipedia.org
    (b) GET /wiki/Hypertext_Transfer_Protocol HTTP/1.1
    Host: nl.wikipedia.org
    (c) GET http://nl.wikipedia.org/wiki/Hypertext_Transfer_Protocol
    (d) GET nl.wikipedia.org/wiki/Hypertext_Transfer_Protocol HTTP/1.1

Week 4

1. The figure below shows a typical communication between a DHCP client and a DHCP server. What is the name of the DHCP message that is sent from the client to the server in the first step, marked with a 1?

2. A DHCP server can provide a client with an IP-address and lease time, but it can provide other information as well. For instance, DHCP will also provide the client with the subnet mask for the network. Name one more piece of information (other than the IP-address, it's lease time and subnet mask) that a DHCP server will typically provide to a client.

3. Suppose that a host wants to obtain an IP-address from a DHCP server. How will the host find the DHCP server in the network? Select one:

   (a) The DHCP server is always reachable at a fixed address in the network

   (b) The host will look up the address of the DHCP server in its iptables

   (c) It will send a broadcast message to all hosts in the network, asking for the DHCP server to identify itself

   (d) The host will request the IP address of the DHCP server using the DNS protocol

4. Which network layer function is described below?:

   - The network layer must determine the route or path taken by packets as they flow from sender to a receiver.
   - Algorithms are used to calculate the paths.

5. (niet mogelijk ivm plaatje)

6. Consider the address below:

   2002:4559:1FE2:0000:0000:0000:4559:1FE2

   This 128 bit address is an example of the addresses used by which network layer protocol?

7. Which of the following is not a type of ICMP message?

   (a) TTL (time to live) expired

   (b) echo reply

   (c) echo request

   (d) bad sequence / acknowledgement number Correct

   (e) destination network unreachable

8. At home, you will typically be provided with only one IP-address by your Internet Service Provider. However, you may want to connect many devices to the internet, such as a desktop PC, laptops, smartphones, your gaming console, a NAS, smart TV or chromecast, and so on...

   Which method is used to connect all these devices to the internet, sharing the same public IP-address?

9. During last week's practical lab, we tried out three different Linux commands (tools) for querying DNS information. Name one of these commands.

10. Consider a TCP connection between Host A and Host B. Suppose that the TCP segments travelling from Host A to Host B have source port number 4012 and destination port number 96. What is the destination port number for the segments travelling from Host B to Host A?

Week 5

1. What is the decimal equivalent of the following binary number? 00011001

2. The subnet mask:

   255.255.255.128

   Can be written in CIDR notation as? Select one:

   (a) /16
   (b) /0
   (c) /25
   (d) /32
   (e) /255

3. Given is the following IP-address with subnet: 192.168.2.10/28.

   How many IP-addresses are in this subnet (including reserved addresses)?

4. How many devices can be provided with an IP-address, using the following Network-ID and subnet: 192.168.1.0/27?

5. Suppose you are a network administrator and want to create a subnet that will contain 1000 hosts.

   What is the smallest possible subnet that will allow you to do so? Select one:

   (a) /23
   (b) /24
   (c) /16
   (d) /22

(e) /0

6. Given is the figure below. At what stage (1, 2 or 3) does the link-layer add error checking bits, rdt and flow control to a datagram? Only enter the numerical value.

7. Suppose that a link layer protocol provides error detection by adding one parity bit to every 8 bits of data that are transmitted, using an even parity scheme.

   Which of the following data streams was received correctly?

   (For this question, you may assume that it is extremely unlikely for more than one bit error to occur during transmission.) Select one:

   (a) Data: 0 1 0 1 1 1 0 1, parity: 1

   (b) Data: 1 1 1 1 1 1 1 1, parity: 1

   (c) Data: 0 1 1 1 1 1 0 1, parity: 1

   (d) Data: 1 1 0 1 1 0 0 1, parity: 0

8. kan niet ivm plaatje

9. Which error-detection technique matches the following description?

   - This technique is widely used in today's computer networks
   - It is often used in the link layer, where dedicated hardware can perform the more complex calculations
   - The codes used in this technique are also known as polynomial codes and the calculations are done using modulo-2 arithmetic

10. What is the Linux command you can use to configure your client's network settings using the Dynamic Host Configuration Protocol?

Week 6

1. kan niet ivm dropdown.

2. When using public key cryptography, we want to make sure that we have the actual public key of the person (or organisation, device, etc.) with whom we want to communicate, and not the public key of someone else pretending to be that person.

   A special institution is responsible for verifying the identity of a person and binding a public key to them by issuing a signed certificate. What is the name of the institution that performs this job?

3. Which protocol on the transport layer is enhanced in security services by using SSL?

4. Virtual Private Networks (VPN) use a special protocol for their security, that operates at the network layer. What is the name of this protocol?

5. Firewalls allow some packets to pass and block other packets. A traditional packet filter examines each packet in isolation to determine whether is is allowed to pass or is dropped. Which of the following information can be used to base this decision on? Select one:

   (a) Source or destination port and IP address

   (b) Status fields in the headers, such as protocol type in the IP datagram field or TCP flag bits

   (c) What router interface the packet arrives on

   (d) All of the above

6. Firewalls can be classified in three categories. Name the category that is described below:

   The firewall does not examine individual packets in isolation, but tracks the state of TCP connections to make decisions about which packets to allow through or to block.

7. What is the name of the system described below?

   To detect advanced attacks, this system performs deep packet inspection and analysis. When it observes a suspicious packet or series of packets, it could alert the network administrators, so that the suspicious activity can be analysed further and appropriate action can be taken.

8. A company will typically run different servers in their network. Which of the following servers would you expect to find within the Demilitarized Zone (DMZ)? Select one:

   (a) The webserver, hosting the company's website

   (b) The server that hosts the test environment for application developers

   (c) The DHCP server that assigns IP-addresses to hosts in the network

   (d) The file server, hosting all the files that are created and used by employees

Week 7

1. What is an advantage of the Linux Operating System, when compared to alternative Operating Systems such as Microsoft Windows? Select one:

   (a) More software, such as games or highly specialised tools for image and video editing, is available for Linux

   (b) Linux is more efficient and stable

   (c) All software applications that run on Linux are free

(d) Linux is easier to use for inexperienced users

2. Given is the figure below where you can see (and not edit) the contents of the file test.c. Write out the complete Linux command that was used to get the output below. You do not need superuser access.

3. Given is the figure below. Write out the complete Linux command to move the file test.c to the directory SE1. You don't need superuser access.

4. Write out the Linux command that shows the full pathname of the current directory.

5. Write out the Linux command that is used to get a complete listing of the contents of the current directory, including permissions for files and directories, such as shown in the figure below.

6. Write out the complete Linux command to make a new directory with the name Infra in the current directory. (You don't need superuser privileges.)

7. Give one of the Linux commands used to forcibly end a process.

8. Write out the complete Linux command that gives the group 'docent' read and execute rights to the file test.c. Keep in mind that the owner must maintain its read and write rights. You do not need superuser rights.

9. Suppose you think VI is too complicated and want to use an editor that is easier to work with. To do this, you want to install the software package, named 'nano'. Write out the complete Linux command used to install nano.

   (Suppose you are logged in with an account that has sufficient rights to do this.)

10. On some Linux distributions, the command 'sl' can be used to display an animated picture of a train (see the screenshot below). Suppose you want to know more about the functionality of this command, and the command line options that are available.

    Write out the complete Linux command you would enter, to get the documentation for the 'sl' command.

0.3.2 antwoorden

Week 1 ontbreekt :(

Week 2

1. 1 Mbps
2. Application
3. TCP

4. client-server
5. 80
6. POST
7. proxy server
8. SMTP
9. POP
10. -

Week 3

1. authoritative server
2. NS record
3. mail server aliasing
4. C. Only reliable data transfer
5. UDP
6. flow control
7. 43
8. Source port
9. A
10. B

Week 4

1. discover
2. DNS server and first-hop router (or default gateway)
3. C. It will send a broadcast message to all hosts in the network, asking for the DHCP server to identify itself
4. Routing
5. B
6. IPv6
7. D. bad sequence / acknowledgement number
8. NAT
9. nslookup, dig
10. 4012

Week 5

1. 25
2. C. /25
3. 16
4. 30
5. D. /22
6. 1
7. A. Data: 0 1 0 1 1 1 0 1, parity: 1
8. b2
9. Cyclic Redundancy Check (CRC)
10. dhclient

Week 6

1. -
2. CA, Certification Authority
3. TCP
4. ipsec
5. D. All of the above
6. stateful packet filter
7. ids
8. A. The webserver, hosting the company's website

Week 7

1. B. Linux is more efficient and stable
2. cat test.c
3. mv test.c SE1
4. pwd
5. ls -l
6. mkdir Infra

7. kill
8. chmod 650 test.c
9. apt-get install nano
10. man sl

1 Computer Networks and the Internet

1.1 What is the Internet

1.1.1 A Nuts-and-Bolts description

Het internet bestaat uit verschillende lagen de **kern** en de **hosts/end systems** Eind systemen zijn apparaten die de **application layer** protocol verstaan. Hier uit kan je verstaan dat het laptops, mobiels, computers, tv's, game consoles etc. Deze zijn met elkaar verbonden door middel van **communicaton links en packets switches** (coax cable, radio etc.) Al deze dingen hebben verschillende **transmission rate**. Deze worden altijd afgebeelt in *bits/s. Meestal zie je mb/s of kb/s. Om van bit naar byte te gaan moet je het getal gedeelt door 8 doen.
De meest voorkomende vorm van packet switchers zijn **routers en link-layer switches**. routers worden meer in de netwerk core gebruikt. Het pad wat een pakketje neemt van van host A naar host B te komen word een **route of path** genoemd.
Een **Internet Service Provider (ISP)** geeft je een ip adres en verbind je met het internet. Je hebt hier verschillende lagen in. Je hebt lokale ISP (laag 3) laag 2 ISPs en in de core zitten laag 1 ISP.
Alle applicaties op je computer die verbindeng willen maken met een andere eind-systeem maken gebruik van een **application protocol** voorbeelden hiervan zijn http, https, ftp, skype, nfs etc. De belangrijkte protocolen voor het internet zijn **Transmission Control Protocol (TCP)** en **Internet Protocol (IP)** Het internet model wat nu in gebruik is heet daarom ook het **TCP/IP** model.
Internet standaarden zijn super belangrijk om te kunnen communiseren tussen systemen. Deze worden vaak gemaakt door het **Internet Engineering Task Force (IETF)**. Openbare vrijstandaarden noem je **requests for comments (RFCs)** daar mag iedereen aan werken; implementeren en wijzigen. Of het word goed gekeurd is een ander verhaal.

1.1.2 A service Description

distributed applications zijn programma's die verdeelt zijn onder meer mensen dus games, sociale media, streaming services omdat ze meer system ding doen ofzo (pag 33.).
Een eindsysteem bind zichzelf aan een socket. een webserver bijvoorbeeld poort 80. Dit noem je een **socket interface**.

1.1.3 What is a protocol

Een protocol is een van te voren afgesproken manier om met elkaar te praten. Dit hebben computers nodig om succesvol met elkaar te kunnen communiseren. Elk protocol heeft zo haar eigen doel.

1.2 The Network Edge

Het internet bestaat uit **clients** en **host** een client is een systeem wat iets opvraagt van een host. En een host is een systeem wat informatie geeft aan een client op aanvraag. Een **data center** kan bestaan uit meerdere **end-systems/host** kan oplopen in de tonnen.

1.2.1 Access Network

Om met het internet te verbinden heb je natuurlijk ook een kabel nodig. Een **Digital subscriber line (DSL)** is zon soort kabel. ookal word DSL voor meer dan 85% gebruikt kan je veel sneller gaan met **fiber to the home (FTTH)**

1.2.2 Physical Media

fysieke media vallen in twee categorieen: **guided media en unguided media**. guided media is bedraad en unguided media in draad-loos.
Unshielded twisted pair (UTP) word vaak gebruikt binnen een gebouw. transfer rates gaan van 10 Mbps tot 10Gbps. Coax cabels zijn een guided **shared medium**, op een shared medium kunnen meerdere systemen aangesloten zijn met maar 1 kabel. terwijl de data niet shared is.
In communicatie worden er twee verschillende soorten satalite gebruikt. **Geostationary satellites** deze blijven op een plek boven de aarde zweven op 36.000 KM. En **low-eath orbiting (LEO) satallites** deze zijn een stuk dichterbij en staan ook nooit stil.

1.3 The Network Core

1.3.1 Packet Switching

Wanneer je een bericht stuurt over het internet word hij in kleine stukjes gehakt, dit noem je **packets** dit pakketje gaat door allemaal **packet switches** (routers en link-layer switches).
De meeste packetswitchers zijn **store-and-forward transmission** dit betekent dat ze eerst wachten totdat alle stukjes van een paket binnen is voordat hij het door verstuurt. Voor iedere link aan een router heb je ook een **output buffer/output queue** dit is een stukje ruimte waar de router pakketjes stopt voordat hij ze versuurt. Een pakketje kan dus een queuing delay oplopen. Wanneer een queue vol zit kan het pakketje worden gedropt en ontstaat er dus **packet loss**.
Er zijn ook verschillende **routing protocols** Voor routers om met elkaar te praten.

1.3.2 Circuit Switching

word niet behandeld. (pag. 55)

1.3.3 A Network of Networks

Omdat er meerdere lagen van ISPs bestaan word degene waar jij in contact mee komt de access ISP genoemd en de bovenste laag de global ISP. Je zou ook kunnen stellen dat de access ISP een **customer** is van global ISP en dat hij weer de **provider** is.
Een **PoP** is een verzameling van routers op een lokatie (op een niet access ISP level) die word gebruikt om meerdere access ISP te verbinden naar local of global ISP. Een super switch eigenlijk maar dan van routers. ISPs kunnen ook direct met elkaar contact leggen zonder hulp van een local of global ISP dit bespaart kosten en noem je **peer**.
Een random bedrijf kan ook een **Internet Exchange Point (IXP)** opzetten. Dit is wel letterlijk een mega switch waar iedereen verbinding mee kan maken om zo internet te delen. Er bestaan ook **content-provider networks** Dit zijn mega bedrijven zoals Google, facebook, Amazon etc. Die als tier 1 (global) ISP spelen.

1.4 Delay, Loss And Throughput in Packet-Switched Networks

Wanneer een pakketje bij een node is kan het vertraging oplopen. Dit kan **nodal processing delay, gueeuing dealy, transmission delay of propagation delay** zijn. Al deze dingen samen noemen we **total nodal delay**.

processing delay is de tijd dat het duurt voor een node om naar de header van een pakketje te kijken. en om te kijken waar het naartoe moet.

queuing delay is de tijd dat het pakketje in de rij moet wachten tot dat het verstuurt kan worden.

transmission delay Dit is de tijd dat het duurt om alle pakketjes op de link te pushen. Dit is niet de tijd van een node naar de andere.

propagation delay Dit is de tijd dat het duurt om een pakketje van een node naar de anderen te sturen.
traffic intensity word bepaald door $paketgrote * aantalpakketjes/transmissionrate$ (La/R). Als een queue van een node is kan hij het pakketje **droppen** dan is de data verloren, hij kan er ook voor kiezen om een ander pakketje te droppen.
Instantaneous throughput is de throughput in een moment. **average throughput** is de gemiddelde throughput.

1.5 Protocol Layers and Their Service Models

1.5.1 Layerd Architecture

Protocol Layering Het idee van protocol layering is dat je al je dingen in andere lagen zet. Iedere laag heeft dan end-points waarmee het een service bied aan de laag erboven. en een end-point waarmee het service vraagt aan de laag eronder. De inhoud van de layer kan veranderen als de end-points maar hetzelfde blijven. Dit zorgt voor een schaalbaal systeem.

Applicatie layer Dit is de layer die drijd op een eind-systeem protocolen zoals HTTP, FPS, Skype, IMAP etc. dit gaat oneindig door. data word hier een **message** genoemd

Transport Layer deze laag zorgt ervoor dat de data van de applicatie laag uit je computer gaat en dat inkomende data naar de juiste applicatie worden geleid. Deze laag kent maar twee protocollen: **TCP & UDP**. Data word hiet een **segment** genoemd

Network Layer Deze laagt zorgt voor de navigatie van node naar node. Dit kent ook maar een aantal protocollen: **IP, en not iets?** //TODO: protocollen. Data word hier een **datagram** genoemd.

Link layer dit is de laag wat letterlijk de data van node A naar node B brengt. Het gebruikt hiervoor de physical layer. Dit doet het met het **PPP** protocol. Data op deze laag worden **frames** genoemd.

physical layer dit is letterlijk een ethernet kabel of wi-fi verbinding.

1.5.2 Encapsulation

Wanneer je data verstuurt over het internet doe je aan encapsulation. Een beetje als een russiche pop met meerdere lagen. Een Message wordt in een segment gestopt. waarbij de transport layer alleen kijkt naar de header van de message. Hij voegt dan wat nuttige data toe aan de segment header en geeft het dan aan de network laag. die maakt er een datagram van en kijkt weer naar de header van de segment. Zo gaat het door tot aan de Link laag. Wanneer de Frame bij een switch komt word hij niet uitgepakt omdat deze ook alleen gebruik maakt van de link layer. Een router werkt echter op de netwerk laag, het pakketje word dan uitgepakt naar een datagram wanneer het bij een router is en weer ingepakt als de router het weer verstuurt. Wanneer het bij zijn eindbestemming aankomt word het weer helemaal uitgepakt tot een Message.

1.6 Network Under Attack

slechte software word **malware** genoemd. Je hebt hierin meerde varianten: **Virusesen** die je zelf op je computer werkt en **worms** die een via een lek in je systeem komen. Veel malware in **self-replicating** het versprijd zichzelf.
deniel-of-service (DoS) is dat je heel veel verkeer stuurt naar een host om zo zijn toegang tot het internet te blokeren. Dit kan op drie manieren *vulnerability attack* is een aanval speciviek gemaakt om een lek te gebruiken om toegang te blokeren. *bandwidth flooding* Met deze aanval stuur/vraag je mega veel en vaak data naar/van een host. *connection flooding* is dat je alle sockets van een host bezet houd door een (half) open connectie te behouden. Je kan ook een **distributed DoS (DDos)** uitvoeren doormiddel van een **botnet**.
Je kan ook op een lokaal netwerk **packets sniffen** dan inspecteer je alle pakketjes op een netwerk. Je kan ook je **ip spoofen** dan doe je net alsof je een andere IP hebt.

2 Application Layer

Dit hoofdstuk gaat over de Application Layer. waar eigenlijk al je netwerk programma's opdraaien.

2.1 Principles of Network Applications

Het principe van netwerk layers is dat je je maar meestal zorgen hoeft te maken over 1 of 2 lagen van het netwerk systeem. Wanneer je een applicatie schijft voor een eind-systeem hoef je dus alleen maar zorgen te maken over hoe de applicatie layer jou programma implementeert.

2.1.1 Network Application Architectures

Er zijn twee verschillende application netwerk architecturen; **P2P** (peer-to-peer) en **client-server** architectuur. Je bent niet gelimiteerd aan deze twee architecturen maar het zijn de enige die op dit moment bestaan. (je kan zelf iets nieuws verzinnen als je genoeg vrije tijd hebt).

client-server

Een client-server is de meest simpele en meest traditionele architectuur. Het gebruikt twee system een *client* en *server* De server moet op een statische plek staan en 24/7 beschikbaar zijn. De client daarin tegen kan dynamiche zijn (van IP verwisselen) en uit en aan gaan wanneer gewenst. De server neemt geen dienst af van de client.

Peer-to-peer

Een peer-to-peer verbinding word onder anderen gebruikt bij Torrents en video gesprekken. De verbinding word onderling gedaan zonder centrale server. Dit systeem schaalt ook veel beter omdat wanneer iedereen een bestand wilt downloaden het ook door meer mensen word geupload.

2.1.2 Processes Communicating

Wanneer een **process** wilt communiceren met een ander process op het Internet heeft het 2 dingen nodig, het adres van de het anderen end-systeem en **Sockits** of **ports** zij

2.2 The Web and HTTP

Tot 1990 werd het internet grotendeels door onderzoekers en scholieren gebruikt. Weinig mensen wisten er nog van. Later in 1994 werd de eerste webbrowser ontwikkeld en daardoor werd het internet onder de het normale volk ook ontamd.

2.2.1 Overview of HTTP

Hyper Text Transer Protocol (HTTP) word gebruikt om documenten (objecten) over het internet te versturen. Wanneer een object (index.html) verwijst naar een ander object (favicon.ico) word deze ook opgehaald. HTTP maakt gebruik van TCP en word beschouwd als een **stateless protocol** dat houd in dat de server geen data bijhoud van haar clients. Wanneer een client 4x hetzelfde bestand zou aanvragen zal de http server de 4e aanvraag hetzelfde behandelen als de eerste.

2.2.2 Non-Persistent and Persistent Connections

HTTP kent twee soorten verbindingen: Non-Persistent en Persistent.

Non-Persistent

Wanneer een client een object aanvraagt van een HTTP server gaan ze eerste een **three-way-handshake** doen. De client vraagt aan de server om een TCP verbinden optestellen en de server antwoord vervolgends met *OK*. Daarna vraagt de client een object aan (bijv. */mydir/index.html*) De server antwoord dan met het bestand *index.html* en de verbinden word verbroken.

De client leest het bestand en ziet dat hij nog 10 plaatjes moet laden. Hij gaat dan weer een three-way-handshake doen en het hele verhaal begint weer opnieuw.

Round-Trip Time (RTT)

RTT is het het tijd wat het duurt voor een pakketje heen en terug te sturen van client naar een server. Propegation delay heen en terug. In het vorige voorbeeld zijn er twee RTT. De eerste is wanneer de client connectie aanvraagt aan de server en antwoord krijgt. De twee is het bestand wat hij vraagt en ook krijgt van de server. Voor elke extra object krijg je bij een non-persisten verbinding **2** RTT's (een om verbinden vast te leggen, en een om het bestand te krijgen.)

Persistent

Een persistent verbinden is veel zuiniger met data dan een Non-persistent verbinden. Want het houd de TCP connectie open tot een bepaalde timeout tijd. Dus in ons voorbeeld met 1 html pagina en 10 plaatjes heb je dus maar 12 RTT's op een persistent verbinding (1 RTT voor TCP verbinden + html bestand + (plaatje * 10)) In plaatjs van 11 * 2 = 22 RTT's bij een Non-persistent verbinding.

2.2.3 HTTP Message Format

Er zijn twee verschillende HTTP bericht formaten, een request en een response. Hieronder volgt een voorbeeld van een request:

```
GET /tools/index.html HTTP/1.1
Host:  www.noeel.nl
Connection:  close
User-agent:  Mozilla/5.0
Accept-language:  nl
```

De request is geschreven in ASCII. De eerste zin is de **request line** de lijnen die daarop volgen zijn de **header lines**. Daarna komt er een wit-regel gevolgt door de **Entity body** maar die is nu leeg omdat het een request is en geen response of `POST`

Request line

De request kan bestaan uit een `GET, POST, HEAD, PUT` of `DELETE`

`GET` request een bestand, `POST` stuur data naar een server (de content staat dan in de body) `HEAD` voor debugging, vraag alleen de head van een response op. `PUT` plaats een bestand en `DELETE` verwijder een bestand. Het word daarna gevolgt door de locatie en de HTTP versie.

Header lines

`Host:` verklaard de host waar de request naar toe moet gaan, is nodig voor proxy's `Connection:` of het een persistant (`open`) of non-persistant (`close`). Data zoals `User-agent` en `Accept-language` worden gebruikt om verschillende data weer te geven per taal/browser.

HTTP Response

```
HTTP/1.1 200 OK
Connection:  close
Date:  Tue, 18 Aug 2015 15:44:04 GMT
Server:  Apache/2.2.3 (CentOS)
Last-Modified:  Tue, 18 Aug 2015 05:44:04 GMT
Content-Lenght:  6532
Content-Type:  text/html

data data data ....
```

In de Status line staat de HTTP versie gevolgt door de statis code en statis zin. (200 betekent dat alles goed is) `Connection:  close` betekent dat de server de TCP verbinding zal sluiten hierna. `Date` De datum en tijd dat de server het bestand van het file system heeft gekregen. `Server:` Allemaal server data, welke http server hij draaid, welke versie en welk OS. `Last-Modified:` handig voor caching enzo. `Content-Lenght` hoelang is de data en `Content-Type` wat voor soort data het is.

2.2.4 User-Server Interaction: Cookies

Omdat HTTP stateless is weet de server niet of de client vaker gebruikt maakt van zijn diensten. Daarom bestaan er Cookies, een Cookie is een string die de server kan zetten met de HTTP header: `Set-cookie: myCookieValue` Wanneer de client een nieuwe request maakt naar dezelfde server stuurt hij altijd in zijn HTTP header (`Cookie: myCookieValue`) zodat de server dit kan processen en client speceviek diensten aan kan bieden.

2.2.5 Web Caching

Je kan ook een caching server hebben. Dit is een server dat een kopie maakt van een bestaande server en dit op een andre plek host. Dit kan handig zijn als de normale server een hoge ping heeft en dus een lage response tijd heeft. Om de caching server te berijken moet je zelf in je browser (of andere applicatie) instellen dat je met de caching server verbinding wilt maken inplaats van de bestaande server.
Wanneer je een HTTP GET request stuurt naar een caching server kijkt hij of hij het bestand heeft en stuurt deze dan naar jou toe. Wanneer het bestand ontbreekt vraagt hij het zelf op bij de orginele server en stuurt het dan naar jou toe en slaat het ook meteen op.

Conditional GET

Het grote nadeel van een caching server is dat hij een out-of-date bestand kan hosten. Om dat te voorkomen checked een caching server eens in de paar weken of er geen nieuwer bestand bestaat. Dat gaat omgeveer zo: /newline Eerst vraagt de caching server het orginele bestand aan

```
GET /img/logo.png HTTP/1.1
Host: www.mylogoImages.com
```

De orginele server geeft een respone:

```
HTTP/1.1 200 OK
Date: Sat, 3 Oct 2015 15:39:29
Server: Apache/1.3.0 (Unix)
Last-Modified: Wed, 9 Sep 2015 09:23:24
Content-Type: image/png

data data data ...
```

Wanneer er een bepaalde tijd voorbij is en de caching server is bang dat zijn bestand out-of-date is checkt hij bij de server of er een nieuwere versie beschikbaar is:

```
GET /img/logo.png HTTP/1.1
Host: www.mylogoImages.com
If-modified-since: Wed, 9 Sep 2015 09:23:24
```

Wanneer er geen nieuwe versie is beantwoord de server met een 304:

```
HTTP/1.1 304 Not Modified
Date:  Sat, 10 Oct 2015 15:39:29
Server:  Apache/1.3.0 (Unix)
```

2.3 Electronic Mail in the Internet

E-mail bestaat uit drie delen: **user agents** (het client programma op het eindsysteem van de gebruiker), **mail servers** (de servers die mails ontvangen en versturen) deze hebben zowel een client als server rol en het protocol om mails te versturen **Simple Mail Transfer Protocol (SMTP)**

2.3.1 SMTP

//TODO

2.3.2 Comparison with HTTP

2.3.3 Mail Message Formats

2.3.4 Mail Access Protocols

2.4 DNS-The Internet's Directory Service

Servers hebben vaak twee adresssen waarmee je met ze kan verbinden: hostname *www.noeel.nl* en het ip adress *185.182.56.20*. Computers willen met het ip addres verbinden omdat dat ook echt iets betekent. Maar voor mensen is dat moeilijk te onthouden. Daarom bestaan er hostnamen die verwijzen naar het ip-adres zodat je die in kan typen in je webbrowser.

2.4.1 Services Provided by DNS

Wanneer jij verbinding wil maken met een hostname verstuurd je computer een verzoek naar een DNS server om het ip addres op te zoeken van de hostname. De DNS server geeft dan het ipaddres en je kan verbinding maken.
Een DNS server bied een aantal services:

Host aliasing Een sub domain (*sub1.sub2.domain.org*) word een **canonical hostname** genoemd deze verwijzen naar *www.domain.org* of *domain.org*

Mail server aliasing Je kan van elk domain naam ook een MX-record opvragen, dit is het addres van de mail server.

load distribution Je kan ook meerdere records van een soort (MX,A etc.) opgeven bij een DNS server. Hij kiest er dan per request een random uit zodat niet een server van je word belast.

2.4.2 Overview of How DNS Works

Er bestaan 4 verschillende DNS servers:

Root DNS servers Dit zijn servers die alleen een lijst hebben met TLD DNS servers. Wanneer jij een request maakt naar *www.google.nl* verwijst deze server je door naar een NL TLD DNS server

Top-Level domain server elk top-level domein naam (nl, com, org, de etc.) heeft haar eigen TLD DNS server. In deze server staan alle **Authoritative DNS servers** van elk domain naam onder dat TLD. Wanneer je een request doet voor het ip van google.nl geeft hij je de Authoritative DNS servers van google.nl.

Authoritative DNS servers Dit zijn de DNS servers die ook daatwerkelijk het antwoord voor je hebben. deze hebben alle records van een domain naam en geven je ook het ipaddres van je bestemming.

local DNS server Dit zijn vaak DNS servers van je ISP (Ziggo bijv.) Deze hebben een lijst van alle Root DNS servers en vragen aan hen het ip addres van je request. Hier maak jij dus ook verbinding mee.

DNS request

Wanneer jij een vraag naar een DNS server doet voor het ophalen van een ip address kan dat op twee verschillende manier gaan.

Various DNS servers

- Een eind-systeem maakt doet een DNS request voor *www.noeel.nl* naar een locale DNS server
- locale server doet een request naar de root server
- de root server leest het *.nl* gedeelte en returnt een adres van een NL TDL DNS server
- de locale server ontvangt de request en stuurt een DNS request voor *www.noeel.nl* naar de NL TDL DNS server
- de TLD server leest het bericht en returnt de authoritative DNS server voor noeel.nl
- de local server stuurt een bericht naar de authoritative DNS server en krijgt de record die hij verwachte

- de local server stuurt het ip-adres door naar het eind-systeem dat het eerder aanvroeg.
- profit!

Recursive queries in DNS Dit werkt op een anderen manier als de varios DNS server variant. inplaats van dat de server steeds een adres returned van een ander DNS server gaat hij zelf vragen aan die server vor het adres. Maar dit komt minder vaak voor.

DNS servers cashen ook al hun resultaten om verkeer te verminderen, en om sneller te zijn. Meestal word iets voor 2 dagen gecached. Maar als web-host kan je dit ook korter of langer instellen.

2.4.3 DNS Records and Messages

2.5 Peer-to-Peer File Distribution

3 Transport Layer

3.1 introduction and Transport-Layer Services

Een transport layer protocol zorgt voor logiche communicatie tussen applicaties van verschillende hosts.

3.1.1 Relationship between Transport and Network Layers

3.1.2 Overview of the Transport Layer in the Internet

Op de transport layer heb je maar twee protocollen **UDP (User Datagram Protocol) en TCP (Transmission Control Protocol)**. UDP levert geen vertrouwlijke data overdracht. Wanneer dat niet aankomt word het niet gecontroleerd. TCP doet dat wel. UDP heeft daardoor wel een kleinere header en kan data versturen zonder een handshake waardoor het bij sommige gevallen sneller kan zijn.
IP/TCP word als een **best-effort delivery service en als unreliable service** gezien. Dit houd in dat het zijn best doet maar geen beloftes doet, je data kan dus niet aankomen of vermist raken. TCP bied wel **reliable data transfer** door te controleren of aan pakketje wel in aangekomen. TCP maakt ook gebruikt van flow control, acknowledgments en onderanderen **congestion control**. Congestion control is service die TCP bied aan het internet om er voor te zorgen dat haar pakketjes niet het hele internet overspoelen. Wanneer TCP merkt dat een node maar 10 pakketjes per seconde kan handelen en hij er zelf 100 per seconde stuurt, zal TCP dit terug schroeven om zo het netwerk niet onnodig te belasten.

3.2 Multiplexing And Demultiplexing

Een applicatie bind zich aan een socket, dit is waarmee hij communiseert met de transport laag en dus verbonden is met het internet.
Wanneer de transport laag een segment van de netwerk laag ontvangt inspecteerd hij deze en stuurt het door naar de bijbehoorende poort. Dit proces noemen we **demultiplexing**. Andersom van applicatie naar network heet **multiplexing**.
TCP en UDP hebben beide twee header fields gemeen: **source port, destination port**. Een port is 16bit en kan dus van 0 tm 65535 zijn. van 0 tot 1023 worden **well-known port numbers** gemoend. Deze zijn eigenelijk al bezet door ouroude internet applicaties. TCP laat zichzelf identifiseren door middel van een four-tuple: source ip, source port, destination ip, dest. port.

3.3 Connectionless Transport: UDP

Voordelen van UDP:

Finer application-level control over what data is sent, and when Omdat de header van udp klein is stuur je geen dingen mee waarvan je niet weet

wat het is. Omdat udp ook bijna geen controle uitvoert op het pakketje word het bijna rouw doorgestuurt naar de netwerk laag.

No connection establishment Met UDP kan je meteen beginnen met versturen van data, het is niet nodig om eerst een **three-way hand-shake** te doen en/of om connecties open te houden. Wat bij TCP wel het geval is.

No connection state UDP is stateless, je hoeft niet voor het verzenden connectie te maken.

Small packet header overhead hier hebben we het al over gehad.

3.3.1 UDP segment structure

3.3.2 UDP checksum

udp heeft net als tcp ook checksums maar ze zijn een heel stuk toleranter. Wanneer udp merkt dat een pakketje niet klopt kan hij hem alsnog doorsturen (met flag) naar de applicatie laag.

3.4 Principles of Reliable Data Transfer

word niet behandelt. (pag 234)

3.5 Connection-Oriented Transport TCP

3.5.1 The TCP Connection

TCP is **connection-oriented** dat wil zeggen dat het eerst een actieve connectie moet hebben met een andere host voordat het pakketjes kan ontvangen of versturen. Dit noemen we een hand-shake. De client vraagt aan de host of hij verbinding mag maken, de host geeft antwoord dat hij het goed vind en de verbinding opent en de client reageert weer dat hij het bericht van de host heeft ontvangen.
TCP is een **full-duplex service** dat houd in dat hij dingen kan ontvangen en versturen tegelijkertijd, in tegenstelling to een walkie-talkie die maar half-duplex is, over. TCP is ook **point-to-point** dat wil zeggen dat er maar 1 ontvangen kan zijn en maar 1 verzender.
Wanneer TCP data krijgt van de applicatie laag doet hij dat in een **send-buffer**. En van tijd tot tijd stuurt hij dit door naar de netwerk-laag. De groote van de hoeveelheid dat hij pakt word bepaald door de **maximum segment size (MSS)** voor de netwerk laag. Deze word weer bepaald door de **maximum transmission unit (MTU)** dit is de maximale frame grote op de link laag. Dit is allemaal plus alle headers van de verschillende lagen en de data.

3.5.2 TCP segment Structure

TCP heeft net als UDP **source en destination port numbers** in zijn header staan. Hij heeft ook een **checksum**.

- **sequence number field en acknowledgment numberfield** (beide 32bit) word gebruikt voor het verfiëren van het pakketje. Elk pakketje dat verstuurt word met TCP heeft een **ack** en **seq** nummer daarmaa kan de ontvanger controleren of hij alle pakketjes wel ontvangen heeft. Als een host een pakketje stuurt met seq: 1 van 8 bytes, stuurt de ontvanger een pakketje terug met ack: 10 ($1 + 8 = 9$). Daarma wil hij zeggen dat hij alles tot 9 heeft ontvangen en klaar is voor pakketje 10.
- **receive window** (16bit) Dit veld word gebruikt voor flow-control. Hierin staat hoeveel bytes de ontvanger kan accepteren.
- **header length field** (4bit) In deze header staat hoegroot de header is.
- **options field** In dit veld onderhandelen de twee eind-systemen over de MSS.
- **flag field** Dit zijn allemaal mini flags van 6bit
  - **ACK** de ACK waar we het eerder overhadden.
  - **SYN & FIN** worden gebruikt voor het opzetten en afbreken van de connectie
  - **PSH** als deze aan staat betekent het dat de ontvanger het bericht meteen moet pushen naar de applicatie laag
  - **URG** geef aan dat de data "urgent" is. Dit wijst naar een **urgent data pointer field** (16bit), maar word nauwlijks gebruikt.

De **sequance van een segment** is belangrijk om de goed volgorde aan te houden. Pakketjes hoeven niet altijd op dezelfde volgorde te worden ontvangen.

3.5.3 Round-Trip Time Estimation and Timeout

word niet behandeld (pag 269)

3.5.4 Reliable Data Transfer

word niet behandeld (pag 272)

3.5.5 Flow Control

TCP bied **flow-control service**. Dit is een speed-matching systeem zodat de rate van data wat je verzend even hoog is als de bottleneck van je netwerk. Dit kan op meerde methodes. Een van die methodes noem je **congestion control** als je de snelheid op een "file" in het netwerk.

receive window word gebruikt door de verzender om te kijken hoeveel vrije buffer er nog is om dingen te verzenden.

4 The Network Layer: Data Plane

De netwerk laag kan je opdelen in twee delen: **data plane en control plane** In hoofdruk 4 gaat het meer over de data-plane.

4.1 Overview of Network Layer

4.1.1 Forwarding and Routing: The Data and Control Planes

Het heel doel van de netwerk laag is om data van een host naar de anderen te verplaatsen. Het bestaat uit

- *Forwarding* Het pakketje naar een de volgende host sturen.
- *Routing* De route bepalen die het pakketje moet volgen. Dit doet hij met **routing algorithms**

Een router heeft een **forwarding table** waar instaat via welke interface pakketje naar buiten moeten op basis van hun bestemming.

4.1.2 Network Service Model

De enige service die de netwerk laag bied is **best effort**. Anderen services die **niet** toepassen op de netwerk laag maar wel bestaan:

- *Guaranteed delivery*
- *Guaranteed delivery with bounded delay* zelfde als hierboven maar dan tijd gebonden
- *In-order packet delivery*
- *Guaranteed minimal bandwidth*
- *Security*

Switches op de link-layer (**link-layer switches**) baseren hun beslissingen op de header in de link layer. Switches op de network-layer (**routers**) doen die beslissingen op basis van de network-layer header.

4.2 What's Inside a Router?

Het router gedeelte van een router bestaat uit de volgende dingen:

- *Input ports* Dit is waar de pakketjes binnen komen. word uitgepakt tot datagram . Dan wordt gekeken waar de het pakketje naartoe moet en word dan (als hij de voorste in de queue is) Doorgestuurt naar de **Switching fabric**

- *Switching fabric* hier kunnen maar een aantal pakketjes tegelijkertijd opzitten (meer hierover later) maar de switching fabric stuurt ze naar de goede **output ports**
- *Output ports* Hier worden de pakketjes weer ingepakt naar de link-layer en dan physical layer en verstuurt over het internet.
- *Routing processor* Dit ding zorgt ervoor dat het pakketje doe goede richting opgaat. Bij ISP hebben ze zelfs een **software definded routing** (SDN?) zodat het vanuit een centrale plek gebruikt. Traditioneel gezien delen routers hun forwarding table met elkaar en ze weten ze waar hun pakketje naartoe moet.

De dataplane doet dingen op de nano-seconde en zit ook vaak in hardware omdat het zo snel moet zijn (forwarding etc.) Dingen in de **control plane** werken vaak op een normale CPU met software.
Er zijn twee verschillende manieren van forwarden in een router:

- *Destination-based forwarding* een pakketje naar een link sturen omdat hij dan zo snel mogelijk op zijn bestemming komt.
- *Generalized forwarding* Hij kan een pakketje ook naar een link forwarden op basis van zijn afkomst, inhoud of omdat hij niet weet waar hij naar toe moet gaan.

4.2.1 Input Port Processing and Destination-Based Forwarding

4.2.2 Switching

De **switching fabric** kan op drie verschillende manier werken:

- *Switching via memory* Dit is de simpelste methode, de data van de input port word gekopieerd naar de ram van de router en gekopieerd naar de output. Met deze methode kan je maar een pakketje tegelijkertijd processen.
- *Switching via a bus* Met deze methode word het pakketje naar alle outputs tegelijk verstuurt via een bus, maar met een label voor de goede output. De anderen outputs negeren het bericht. Ook maar een bericht tegelijkertijd.
- *Switching via an interconnection network (cross-bar)* met deze methode kan je meerde pakketjes tegelijktijd versturen zolang ze maar niet naar de zelfde input en/of output gaan. een pakketje van input A kan naar output X terwijl er ook een pakketje van input B naar output Y gaat. Cross bar is **non-blocking**

4.2.3 Output Port Processing

4.2.4 Where Does Queuing Occur

het queue van een router gebeurt na de route bepaling in de input ports en aan het gebin van de output ports. Wanneer die queue vol zit kan er **packet loss** voor komen als er nog meer pakketjes bijkomen. Die worden dan gedropt.

4.3 The Internet Protocol (IP): IPv4, Addressing, IPv6 and More

Er bestaan anno 2017 twee versies van IP (internet protocol): Ipv4 en Ipv6. IPv4 word het meest gebruikt maar heeft maar een hele kleine hoeveelheid adressen 32bit. Die van IPv6 zijn 128 bits.

4.3.1 IPv4 Datagram Format

Het IPv4 datagram bestaat uit de volgende onderdelen:

- *Version number* 4bits, staat het versie nummer van de IP.
- *Header lenght* 4 bits, beschrijft hoe groot de header is. Meestal 20 bytes.
- *Type of Service* (TOS) hier wordt beschreven wat voor soort data dit is. realtime, VOIP etc.
- *Datagram length* 16 bit, dit is de totale lengte van de datagram (header+data) meestal niet meer dan 1,500 bytes (max 65,535)
- *Identifier, flags, fragmentation offset* IPv6 heeft geen fragmentation
- *Time-to-live (TTL)* 8bit, In dit veld word aangegeven hoeveel hops dit datagram nog mag maken. Wanneer hij bij een nieuwe router komt word deze waarde -1 gedaan. Wanneer het 0 berijkt moet de router hem droppen.
- *Protocol* 8bit, Hierin staat welke transport-layer protocol deze datagram naar moet wordengepaast wanneer hij op zijn bestemming is. Dus TCP of UDP
- *header checksum* 16bit, checksum voor de header.
- *Source and destination IP addresses* 32bit p.s, hier staan de source en destination ip in allebij in een apart veld van 32bit.
- *Options* Hier kunnen not extra dingen staan, word bijna niet gebruikt. IPv6 heeft dit veld niet meer.
- *Data* De data van de datagram. (TCP, UDP, ICMP etc.)

4.3.2 IPv4 Datagram Fragmentation

Een Ethernet frame heeft een **maximum transmisson unit (MTU)** van 1,500 bytes. Maar somige wide-area link op het internet kunnen hebben maar een MTU van 576 bytes. Als dat gebeurt word het pakketje opgesplits in het maximale MTU van de bottleneck. Dit noem je een **fragment**. Het pakketje word daarna **niet** weer opnieuw groter gemaakt als hij de bottleneck uit is. Bij IPv6 stuurt de router een ICMP bericht naar de verzender om te zeggen dat hij een kleinere MTU moet doen. Dit hebben ze veranderd omdat het voor de router te process intensief is.

4.3.3 IPv4 Addressing

De connectie tussen een host en een link noem je een **interface**.
IPv4 addresen zijn 32bit, er kunnen dus 4 miljard unieke addresen bestaan. IPv4 word eigenlijk altijd opgeschreven in een **dotted-decimal notation** (193.32.216.9) elk getal tussen de puntjes is 8 bits, je zou dit ook in binaur kunnen opschrijven

$$11000001001000001101100000001001$$

Je kan in een netwerk **subnets** maken, subnets kan je onderverdelen per interface van je router(s). Zodat je je netwerk gescheiden houd. Als je dan een ip-range aan een bepaalde subnet wilt toekennen heb je te maken met een **subnet mask**, daarmee vertel je hoeveel bits je toekent dus *192.168.0.0/28* ken je de laatste 4 bit toe aan je subnet ($32 - 28 = 4$). Wat neerkomt op 16 addresen, daarvan zijn er altijd 2 gereserveerd. een voor je default gateway (router) en een voor de broadcast (voor DHCP bijvoorbeeld). Je houd er dus 14 over om je hosts op aan te sluiten.
De internet addres toekennings methode heet: **Classless interdomain Routing (CIDR)** van oorspring had je namelijk subnet classes. Waavan 10.0.0.0/8 classe A was, 10.0.0.0/16 B, 10.0.0.0/24 C. Maar met deze methode heb je of 255 (C) addresen of 65536 (B) of 16777216 (A) Terwijl je er als organistie maar vaak 1k nodig had.
ISP krijgen een blok aan ipaddresen en ddres staat niet vast en je kan dus altijd een andere krijgen. deze ook weer uit aan hun klanten. Van een locale ISP zoals Ziggo krijg je een **dynamic ip addres** dit a Wanneer jij verbinding maakt met een nieuw netwerk heb je een IP addres nodig. Die kan je krijgen van een **Dynamic Host Configuration Protocol (DHCP)** server, ieder netwerk heeft een of meerdere van dit soort servers . Je krijgt van een DHCP server een **temporary IP address** Deze heeft een lease-tijd en moet je om een bepaalde tijd weer vernieuwen bij je DHCP server. gelukkig gebeuren al deze dingen automatish want DHCP is een **zero-conf/plug-and-play** systeem.
Wanneer een hosts een ip-addres wilt moet hij doro een vier-stappen plan heen van DHCP, het zogenaamde **DORA**

- *DHCP server discovery* stap 1 is om de DHCP server te vinden. Dit doet een host door middel van een **DHCP discovery message** dit word naar

ip 255.255.255.255 gestuurd, het broadcast IP. Op 255.. staat geen host maar is bedoelt als broadcast, iedereen in dezelfde subnet ontvangt dit bericht. In dit bericht staat zijn ID van zijn request samen met nog wat info.

- *DHCP server offer(s)* De DHCP server ontvangt het bericht en stuurt een broadcast terug het netwerk op met een **DHCP offer message**. Hij doet een broadcast omdat de andere host nog geen IP addres heeft. In dit offer staat een IP addres die hij mag gebruiken. Samen met een **lease-time** (hoelang hij dit IP addres mag gebruiken)
- *DHCP request* De client stuurt een **DHCP request message** met het IP dat hij wilt hebben (de host kan meerde offers gehad hebben van verschillende DHCP servers)
- *DHCP ACK* de DHCP server stuurt een **DHCP ACK message** dat alles OK is.

4.3.4 Network Address Translation (NAT)

Omdat er maar 4 miljard IP addresen zijn en toch iedereen met zijn, computer, laptop, tablet en telefoon tegelijkertijd op het internet wilt gaan maken we gebruikt van **Network address translation (NAT)**. NATs worden toegepast op **private networks**. Je krijgt van Ziggo, KPN, xs4alll of elke andre ISP maar 1 IPv4 (soms ook een IPv6) addres. Maar je hebt ondertussen wel 20 hosts thuis staan die allemaal verbonden moeten worden. Je router maakt dan gebruikt van NAT, hij kens alle host binnen je netwek een ip addres toe van 192.168.0.0/16 of 10.0.0 .0/8 dit zijn gereserveerde adderesen voor NATs. Er bestaat geen public adderesen met deze waardes. Ze bestaat dus alleen in je thuis netwerk. Als je een pakketje verstuurt naar een host buiten je thuis netwerk pikt je router dit op en veranderd de source ip en de source port naar het **public IP** en een andere port. Wat je van je ISP hebt gekregen. Hij slaat alle connecties op in een **NAT translation table** (source-dest ip, port) zodat hij weet welk binnen komend paktje voor welk private ip zijn. Hij vervangt dan de port nr en dest ip en stuurt het pakketje weer door.
Wanneer je deel wilt nemen in een p2p (peer-to-peer) connectie of een ander soort verbinding waar het belangrijk is dat je poort nummer kloppen kan een host gebruik maken van **NAT traversal** tools en UPnP die configureren je NAT in je router dat hij niet aan de poort nummers mag komen. En dat hij alles van port x automatich forward naar een specivike host. Een NAT valt onder de categorie **middlebox**

4.3.5 IPv6

De belangrijkste veranderingen in IPv6 zijn:

- *Expanded addressing capabilities* ipv6 is 128bit inplaats van 32bit van v4. IPv6 ondersteund ook **anycast address** hiermee kan je een broadcast doen naar een groep public ip addresen.
- *A streamlined 40-byte header* ipv6 heeft een fixed 40 byte header
- *flow labeling* Je kan aangeven wat voor service je datagram is. (real-time, high priority etc)

Een Ipv6 datagram bestaat uit de volgende onderdelen:

- *Version* 4bit, de IP versie.
- *Traffic class* 8bit, hier kan je een hogere priority geven aan je datagram als het real-time data
- *Flow label* 20bit, zoals beschreven in flow labeling
- *payload length* 16bit, hoeveel bytes de header+data is
- *Next header* welke transport layer protocol word gebruikt, zelfde als IPv4
- *Hop limit* zelfde las TTL in IPv4
- *source and desination addresses* 128bit p.s. source en destination in een apart veld.
- *Data* de data.

IPv4 vs IPv6:

- *fragmentation/reassembly* v6 bied geen fragmentation. Als de MTU bij een link ineens kleiner moet word er een ICMP bericht gestuurt naar de verzender van het pakketje dat hij het opnieuw moet sturen met een kleinere MTU
- *Header checksum* v6 heeft geen checksum omdat het in de andere layers al meer dan genoeg voorkomen.
- *Options* v6 heeft geen opties.

Wanneer je een IPv6 pakketje over het internet verstuurt is de kans groot dat hij bij een router komt die IPv6 niet ondersteund. De laatste router die wel v6 ondersteund stopt dan de gehele v6 datagram in een IPv4 datagram en die adraseert hij naar de eerste node op de route die wel v6 ondersteund. Deze router leest het pakketje dan, pakt het uit en verstuurt het orginele v6 pakketje alsof er niks gebeurt is. Dit process noemen we **tunneling**.

5 The Network Layer: Contol Pane

5.6 ICMP: The Internet Control Message Protocol

Het Internet Control Message Protocol (ICMP) wordt gebruikt door hosts en routers om network-layer informatie met elkaar uit te wisselen. ICMP wordt vooral gebruikt om errors te melden. ICMP wordt verzonden in IP datagrams, net als UDP of TCP segments. De ICMP quench message wordt bijna niet gebruikt, omdat TCP zelf congestion control heeft.
Het programma traceroute maakt gebruik van ICMP TTL expired messages om de route te bepalen. Traceroute stuurt UDP segments met een onwaarschijnlijke port number en een TTL van n om de naam van de $n$de router in de route te krijgen. Omdat de destination host een port unreachable message terug stuurt weet traceroute dat het eind berijkt is.

Table 1: ICMP message types

ICMP Type	**Code**	**Beschrijving**
0	0	echo reply (to ping)
3	0	destination network unreachable
3	1	destination host unreachable
3	2	destination protocol unreachable
3	3	destination port unreachable
3	6	destination network unknown
3	7	destination host unknown
4	0	source quench (congestion contol)
8	0	echo request
9	0	router advertisement
10	0	router discovery
11	0	TTL expired
12	0	IP header bad

6 The Link Layer and LANs

6.1 Introduction to the Link Layer

Elk apparaat dat een link-layer protocol draait wordt een **node** genoemt. Hosts, routers, switches en WiFi access points zijn dus allemaal nodes. De verbinding tussen nodes wordt een **link** genoemt. Een link gaat van node tot node. Over links worden **link-layer frames** verstuurd, die network-layer datagrams dragen (encapsulate).

6.1.1 The Services Provided by the Link Layer

De link laag kan de volgende services aanbiden:

- *Framing.* Bijna alle link-layer protocollen pakken de network-layer datagrammen in als een link-layer frame.
- *Link access.* Het medium access control (MAC) protocol beschrijft hoe de frames door de link worden gestuurd. Dit is vooral van belang bij WiFi omdat er dan meerdere nodes op een link zitten.
- *Reliable delivery.* Een link kan leverings zekerheid bieden. Dit zie je als de link onbetrouwbaar is zoals bij WiFi.
- *Error detection en correction.* Dit vermindert het aantal retransmissions. Dit wordt verder behandeld in 6.2

6.1.2 Where Is the Link Layer Implementend

De link laag is geimplementeerd in de **network adapter**, ofwel **network interface card (NIC)**. Deze network adapter verzorgt het in- en uitpakken van de frames en voert eventueel de error detection en correction uit. Het grootste gedeelte van de link laag is geimplementeerd in hardware.

6.2 Error-Detection and -Correction Techniques

Bit-level error detection and correction zijn twee services die veelal door de link laag wordt geleverd. De transport laag bied deze service ook, maar minder uitgebreid. Door fouten in de link kan er soms een bit of meer geflipt worden. Om deze te detecteren en te herstellen zijn een aantal technieken, waarvan er hier drie behandeld zullen worden. De error detection techniek is niet water dicht. Er kunnen nog steeds **undetected bit errors** optreden.

6.2.1 Parity Checks

De simpelste vorm van error detection is een enkele **parity bit**. Bij een even parity scheme zorgt de parity bit ervoor dat het aantal 1s in heb bericht even is. Mocht er een bit geflipt zijn tijdens transport dan is dat te detecteren omdat het aantal 1s nu oneven is. Dit is weergegeven in figuur 1.

Figure 1: parity check (even)

data bits	**parity bit**
0111000110101011	1

Als er twee bits geflipt worden zal dit niet worden gedetecteerd. Omdat dit voor kan komen wordt onderandre de **two-dimensional parity** scheme gebruikt. Hierbij wordt gebruik gemaakt van een parity rij en colom. Deze techniek bied daardoor niet alleen error detection, maar als er een bit geflipt is ook error correction. Dit is weergegeven in figuur 2.

Figure 2: 2D parity check (even, no errors)

1	0	1	0	1	1
1	1	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

Figure 3: 2D parity check (even, errors)

1	**0**	1	0	1	1
1	***0***	**1**	**1**	**0**	**0**
0	**1**	1	1	0	1
0	**0**	1	0	1	0

Het detecteren en herstellen van fouten wordt **forward error correction (FEC)** genoemd. FEC vermindert het aantal retransmissions en ook de latency omdat er niet op de retransmission gewacht hoeft te worden.

6.2.2 Checksumming Methods

De **Internet Checksum** is een error detectie techniek. Door de berichten van d-bits op te delen als integers van k-bits en die op te tellen en het 1s complement hiervan te nemen kan de checksum worden opgesteld. De checksum wordt door de verzender in de header mee gestuurd. De ontvangen doet vervolgens hetzelfde met de ontvangen data (inclusief checksum). Als het resultaat hiervan een 0 bevat is er een error.
Bij TCP en UDP wordt de checksum over de header en de data berekend. Bij IP wordt de checksum alleen over de IP header berenkend omdat TCP en UDP hun eigen checksum hebben.

6.2.3 Cyclic Redundancy Check (CRC)

De **Cyclic Redundancy Check (CRC) codes** is een techniek om errors in berichten te detecteren. Het voordeel van CRC is dat deze meer errors kan detecteren dan parity bits of checksums. Het aantal errors dat het kan detecteren

wordt bepaald door de lengte $r+1$ van een door de nodes afgesproken rijtje bits genaamt de generator. Dit aantal errors is r. Als er meer dan r errors zijn dan kunnen deze gedetecteerd worden met een kans van $1 - 0.5^r$. CRC is echter wel zwaarder om te berekenen. CRC wordt daarom veelal op de link laag in hardware toegepast, omdat deze sneller is dan software. CRC codes staan ook bekend als **polynomial codes**.

8 Security in Computer Networks

8.1 What is Network Security?

Een veilige verbinden kan bestaan uit de volgende eigenschappen van **secure communication**

- *Confidentiality* Het bericht kan alleen worden gelezen door de ontvangen er en de verzender. Dit kan je krijgen door gebruikt te maken van **encrypted**
- *Message intergrity* Je wilt er zeker van zijn dat het bericht wat je ontvangt ook precies hetzelfde is wat jij hebt verstuurt.
- *End-point authentication* Je wilt er zeker van zijn dat de ontvanger ook echt is wie hij zecht dat hij is.
- *Operational security* Je wilt dat je netwerk veilig is om op te werken.

8.2 Principles of Cryptography

Waarneer je blanko tekst versuurt staat dat bekent als **cleartext/cleartext** je kan met een **encryption algorithm** je data encrypten, dan staat het bekent als **ciphertext**
Om data geencrypt te versturen maak je gebruik van **keys** Als je data encrypt met een key, de data verstuurt naar een ontvanger en de ontvanger gebruikt dezelfde key om de data te decrypte noem je dit **symmetric key system**. Het probleem hiervan is dat je eerst een key moet afspreken.
Met **public key system** heeft iedere host 2 keys, een public en een private. Je private key deel je met niemand. Deze gebruik je om berichte te decrypten (of encrypten maar dan komt later). Je public key iedereen bij. Iemand kan dus een bericht encrypten met jou public key die kan dan alleen worden gedecrypt met jou private key.

8.2.1 Symmetric Key Cryptography

Monoalphabetic cipher is een encryptie dat je een letter door een andere letter vervangt. drie verschillende aanvallen op encryptie:

- **Ciphertext-only attack** Met deze aanval heb je toegang tot de ciphertext zonder identicatie wat het bericht zecht.
- **Known-plaintext attack** Nu heb je de cipertext en weet je (gedeeltelijk) wat er in het bericht staat, makkelijker te decrypten dus.
- **Chosen-plaintext attack** Nu kan de aanvaller text incrypten opdezelfde methode als het oorspronkelijke bericht.

Polyalpabetic encryption is dat je op basis van de positie van de zin de letter een andere letter maakt.

8.2.2 Public Key Encryption