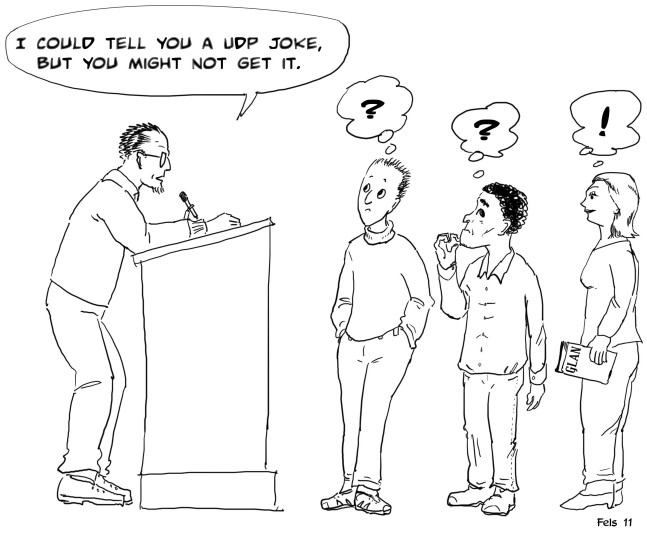
Kapitel 5

TCP/IP i teorin



5.1 Binära tal

1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, . . . Vi börjar med lite matte. Den här sifferserien av fördubbling-

ar, eller tvåpotenser, återkommer ofta i datorsammanhang. I binära tal (ettor och nollor) ger serien ovan värdet på varje siffra (bit) räknat från höger (minst signifikanta biten, least significant bit, LSB). Till exempel är 101=5=1*4+0*2+1*1 och $1000 = 8 = 1*2^3+0*2^2+0*2^1+0*2^0$. Räkneregler för binär addition är 0+0=0, 0+1=1+0=1, 1+1=10.

En annan talbas som används ofta är 16, hexadecimala tal. En byte är 8 bits, decimalt har en byte värden mellan 0 och 255, hexadecimalt mellan 0 och FF. Ett annat namn på byte är oktett. Se tabell 5.1 och lär dig gärna den utantill om du inte redan kan den. Att räkna om från binärt till hexadecimalt (och tvärtom) är lätt om man kan tabellen. Det är bara att dela upp bitarna i grupper om fyra. T.ex är FE25=1111 1110 0010 0101. Det har man nytta av för att förstå nätverk, vilket är det här kapitlets huvudämne.

Vill man låta datorn räkna om mellan olika talbaser kan man använda programmet `bc`. Bara skriv bc vid prompten och inne i programmet skriv t.ex. `obase=2` för att få utmatning i binärt eller `ibase=16` för att få inmatning i hex (out base och in base). Man avslutar bc med **ctrl-d**. Tänk på att hexadecimala tal måste skrivas som versaler, och att om du ändrar ibase så kommer den tolka allt, även kommande ibase=, i den basen. Det kan vara lättare att avsluta än att räkna om i huvudet, efter omstart är både ibase och obase 10 (decimalt). Nu blir det inte svårare matte än så här i den här boken, men binära tal återkommer.

5.2 IP, mask, gw

Din dators IP-nummer är adressen till den på Internet. IP-nummer är ett 32-bitars tal (med IPv4) och skrivs oftast som fyra byte (oktetter i viss litteratur). Alltså tal mellan 0 och 255. Till exempel 194.47.250.234 eller 130.235.20.3. Det finns några serier som är reserverade för interna nätverk, 10.x.x.x, 172.16.x.x till 172.31.x.x och 192.168.x.x. Dessa adresser finns inte ute på det stora Internet, utan används för privata nätverk ofta bakom NAT (mer om det senare i 7.5). Det är inte heller alla 4 miljarderna (2^{32}) möjliga siffrorna som går att använda av andra skäl.

För att kunna kommunicera via TCP/IP med andra datorer är det tre uppgifter som behövs: IP-nummer, nätmask och default gateway. Ett lokalt nätverk kallas LAN (Local

Tabell 5.1: binärt och hex

Dec	Bin	Hex
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Area Network), det är de datorer som är direkt åtkomliga via kabel och switchar. Varje dator på LAN:et har ett IP-nummer som börjar med samma siffror. Nätmasken anger vad som är LAN och vad som är Internet. Till exempel för en dator med IP 192.168.0.85 är nätmasken 255.255.255.0. Från kombinationen av IP och mask kan man utläsa att alla enheter på LAN:et har adresser 192.168.0.x, där x kan ha alla värden från 0 till 255. För att förklara det återvänder jag till ettor och nollor. Masken är alltid en följd av ettor följt av ett antal nollor. Totalt 32 tecken. Där det finns ettor är det Internet och där det finns nollor är det LAN. En tabell kan göra det klarare.

Här är gränsen vid en jämn byte, men behöver inte vara så. Förr i tiden delade man in i klass A, B eller C beroende på om första nollan kom efter första, andra eller tredje punkten. Numera används ofta ett system som heter CIDR, Classless InterDomain Routing, där gränsen mellan ett och noll (Internet och LAN) kan vara vid vilken bit som helst. Men det är fortfarande aldrig någon etta efter första nollan.

Tabell 5.2: IP-nummer och mask

192	168	0	85
11000000	10101000	00000000	01010101
255	255	255	0
11111111	11111111	11111111	00000000

Detta ger 8 olika möjliga värden för varje byte, dessa visar jag i nästa tabell.

Tabell 5.3: Nätmask

dec	binärt	cidr
0	00000000	/24
128	10000000	/25
192	11000000	/26
224	11100000	/27
240	11110000	/28
248	11111000	/29
252	11111100	/30
254	11111110	/31
255	11111111	/32

Värdet i sista kolumnen är ett annat sätt att ange mask genom att efter ett snedstreck ange antalet ettor, i det här fallet för den sista oktetten (vilket är den vanligaste). Denna notation kallas också CIDR (uttalas cider) och används i många andra sammanhang såsom brandvägg och namnservrar. I CIDR-notation heter de tre privata näten 10/8, 172.16/12 och 192.168/16 och exempelnätet ovan heter 192.168.0.0/24. Masken /31 är ganska oanvändbar, och /32 är mest ett annat sätt att ange ett enda IP. Men alla de andra används i verkligheten.

När man kör TCP/IP med vanliga nätverkskort behövs utöver det egna IP:t en adress för hela nätet och en för broadcast. Dessa är nästan alltid de lägsta och högsta av de möjliga, och kan därmed räknas ut om man vet IP och mask. För vårt exempelnät blir nätets adress 192.168.0.0 och broadcast 192.168.0.255. Har man IP 82.23.134.140 och mask

255.255.255.240 (/28) så blir nätet 82.23.134.128 och broadcast 82.23.134.143. På ett /28-nät kan man ha 16 adresser, men eftersom lägsta och högsta blir upptagna är det 14 användbara IP. Se tabell 5.4 för detta exempel och hitta på egna exempel om det är oklart. En ledtråd är att skriva mask och broadcast under varandra binärt, där det är nollor i masken ska det vara ettor i broadcast. Ett annat tips är att använda kommandot `ipcalc`. Med IP och mask kan vi nå he-

Tabell 5.4: IP och mask igen

IP	82	23	134	140
IP	01010010	00010111	10000110	10001100
Mask	255	255	255	240
Mask	11111111	11111111	11111111	11110000
Nät	01010010	00010111	10000110	10000000
Nät	82	23	134	128
BC	01010010	00010111	10000110	10001111
BC	82	23	134	143

BC=Broadcast

la vårt LAN. Men Internet är ju ett nät av nät, och att endast kunna surfa till sina egna datorer är inte så kul. Därför har man en enhet (dator, router) på sitt LAN som har kontakt med Internet. Denna kallas av tradition default gateway, eller kortare gw (men egentligen är den inte en gateway). Det kan vara en dator med två nätverkskort eller en färdigköpt liten låda. Den har minst två IP-nummer, varav det ena ska ligga på samma nät som ditt LAN. Det allra vanligaste är att den har nästa nummer över vad nätet har, som 192.168.0.1 och 82.23.134.129 i våra exempel. Men den kan även finnas på .254 i ett /24-nät, och egentligen kan den ha vilket värde som helst inom nätet.

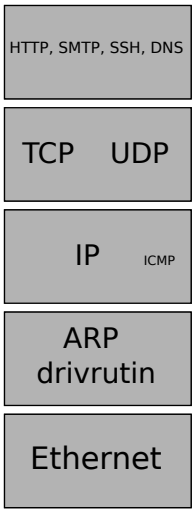
En dator som vill nå utanför sitt LAN jämför den önskade adressen med sitt eget IP och mask. Om det är utanför LAN:ets IP-område vet den att den ska skicka det till sin gw. T.ex om vår 192.168.0.85 vill gå till 192.168.0.126 skickar den broadcast på LAN, men vill den gå till 74.125.43.105 förstår den att den adressen inte går att nå lokalt utan skickar paketet till 192.168.0.1 som i sin tur skickar det vida-

re. Vid ethernet (vanliga nätverkskort) tillkommer ett lager som heter ARP, mer om det lite längre fram i 5.5.1.

5.3 Stack: lager och inkapsling

För att förstå TCP/IP behöver man förstå både helhet och detaljer, och som författare är det svårt att välja i vilken ordning man ska berätta. Det blir lite upprepningar och många hänvisningar framåt och bakåt. Om det mesta är okänt kan du behöva läsa detta kapitel flera gånger och även andra böcker.

För TCP/IP används oftast en femlagersmodell, vars implementation kallas en stack. Modellen kan se ut som i figur 5.1



Figur 5.1: *Femlagersmodell för TCP/IP*

Nedersta lagret är det fysiska; ettor och nollor som representeras elektriskt eller med ljus. Nästa lager uppåt är det logiska länklagret, som i fallet med Ethernet består av drivrutin i kärnan och hårdvaruadress (MAC-adress) som via ARP kopplas till nästa lager IP. Detta lager, nätverkslagret är det jag har beskrivit i ovanstående stycke om IP, mask och gw. Ovanför finns transportlagret med TCP och UDP och ovanför detta finns applikationslagret med till exempel

HTTP, SMTP och SSH. Känner du dig förvirrad så är det normalt. Vad är ett lager, hur hänger de ihop, varför finns de? Jag försöker förklara.

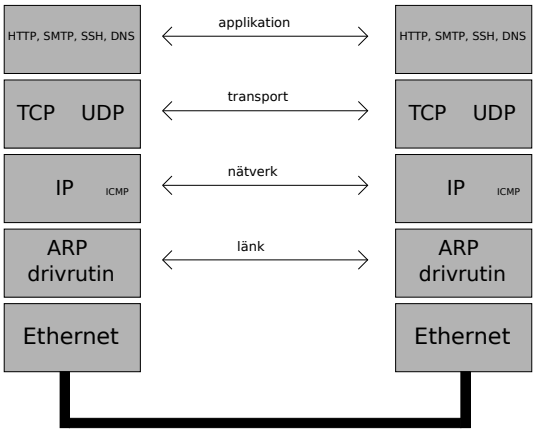
Eftersom jag har beskrivit IP tidigare börjar jag i mitten av modellen. På en dator som skickar data får IP-lagret information från antingen TCP eller UDP. IP märker ingen skillnad, det är bara indata. Men IP får även en adress att skicka datan till. Varje nytt lager neråt i modellen lägger till en header (ett huvud) med sin del av TCP/IP. För IP innehåller denna bland annat avsändarens och mottagarens IP-adresser. Header från IP och data från till exempel UDP skickas vidare till länklagret. Där läggs ytterligare en header till, med bland annat hårdvaruadresserna. Och i det fysiska lagret skickas all info ut på TP-kabeln. Fördelen med olika lager är att varje lager bara skickar data till det närmast nedanför eller ovanför på ett förutbestämt sätt, och behöver inte bry sig om vad de andra lagren gör. IP går att köra över nätverkskort (Ethernet), modem (PPP), fiberkabel (Sonet/ATM), brevduvor eller nästan vad som helst.

På den mottagande datorn går informationen nerifrån och upp. Fysiska lagret och länklagret skalar bort sin header. IP kommer se sitt paket exakt som det såg ut för IP-lagret hos sändaren. UDP kommer se sitt paket som ett UDP-paket och kan skicka det vidare till exempelvis en DNS-server i översta lagret hos mottagaren.

Varje lager ”pratar” på sätt och vis med varje annat lager på samma nivå hos sändare och mottagare, och tar bara emot och skickar data uppåt eller neråt i stacken. Till exempel ”pratar” en webbläsare HTTP med en webbserver på en annan dator, och skickar sina HTTP-meddelanden till kärnans TCP-lager. Det är lättast att förstå via figur 5.2.

Där ser man en stack på varje dator på samma LAN. Informationen går från applikationslagret på ena datorn neråt i stacken och sen uppåt på den andra. Huvud läggs till på vägen ner och tas bort på vägen upp. Det som transporteras på LAN:et visas i nästa bild 5.3 på en ethernetram (frame). Applikationsdata är det som programmet vill skicka. Transportlagret lägger på en UDP-header med 8 byte. Nätverkslagret lägger på 20 byte IP-header. Länklagret lägger till 14 byte ethernet-header, och 4 byte kontrollsumma (CRC) på slutet.

I IP-headern finns IP-adresser för sändare och mottagare, kontrollsumma (men enbart för headern), antalet byte i hela paketet, vilket protokoll som transsportskiktet använder



Figur 5.2: *Hur datorer kommunicerar via TCP/IP*

Ethernet header	IPv4 header	UDP header	Applikationsdata (t.ex. DNS, NFS, RTP)	CRC
14 bytes	20 bytes	8 bytes	18-1472 bytes (i skalan här 58 bytes)	4

Figur 5.3: *Ethernetram*

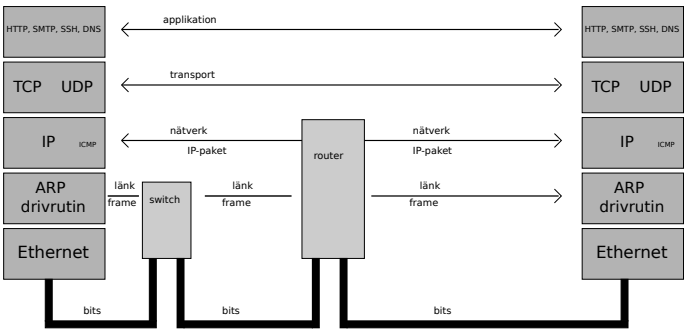
(UDP eller TCP) och ett TTL-värde (Time to live, det räknas ner för varje router som passeras). Det finns några fält till, men dessa är de viktigaste.

I UDP-headern finns portnummer för sändare och mottagare (lite annat också) Mer detaljer finns i 5.5.2.

I Ethernet-headern finns MAC-adress för sändare och mottagare på LAN:et. Alltså adressen till din router om det är ett paket som ska vidare ut på Internet, men om trafiken är till en dator på samma subnät (LAN) så är det dess MAC-adress som står som mottagare.

IP-paket levereras och adresseras alltid från respektive ändpunkt. Ska det från datorn 194.47.250.234 till 173.194.34.174 så är det dessa adresser i IP-headern hela tiden under paketets väg, från Datorföreningen i Lund till Google. Däremot varierar vilka adresser som används i länklagret för varje nät som passeras. Applikationsdatan eller UDP/TCP huvudet ändras inte under vägen, och det enda som förändras (normalt sett) i IP-huvudet är TTL och kontrollsumman.

Så här kan ett nät med två datorer, en switch och en router se ut.



Figur 5.4: *Nät med datorer, switch och router*

Switchen jobbar på fysiska och länklagret, den bryr sig enbart om MAC-adresser. De enheter den tar emot och skickar kallas ramar (frames).

Routern jobbar främst på IP-nivån, där den även har en egen adress. Men den har förstås även egna nätverkskort med varsin MAC-adress för att prata med switchen till vänster och datorn till höger. Det routern förändrar i IP är enbart TTL och kontrollsumma.

Switchar och routrars huvuduppgift är att välja väg på respektive nivå. Switchen har många fysiska portar och skickar bara till den där rätt dator är ansluten. Förut fanns det hubbar som skickade allt på alla sina portar, men de är sällsynta numera. Även routern är ansluten till flera nät, och väljer vart den skickar paketen beroende på destination på IP-nivå.

På det lägsta lagret är det bara ettor och nollor i någon fysisk representation. Och varken switch eller router förändrar något på transport- eller applikationsnivån.

En router med NAT (bredbandsdelare) gör mycket mer än att räkna ner TTL, den skriver om både IP-adresser och portar, mer om NAT finns i stycke 7.5, här handlar det mer om ”riktiga” routrar på företag och ute på Internet.

5.4 Protokoll

Du har nog redan en viss aning om vad ett protokoll för nätverkskommunikation är. Datorer är exakta maskiner som behöver väldefinierade regler för allt de ska göra. En stor fördel med protokollen i TCP/IP-familjen är att de är öppna standarder, vem som helst kan läsa vad varje etta och nolla ska betyda. Alla förslag på nya protokoll eller förbättringar av de gamla publiceras på Internet i dokument som kallas RFC:er (Request For Comments). De finns bland annat på http://www.rfc-archive.org/. Den första RFC:n kom 1969 och nu finns det över 6000. IP beskrivs i RFC791, ICMP i RFC792 och TCP i RFC793. Brevhuvud för e-post beskrevs först i RFC822, men nu är det RFC2822 som gäller för det. Jag kommer att hänvisa till flera RFC:er senare i detta kapitel.

Eftersom alla tillverkare av operativsystem och hårdvara kan läsa de öppna specifikationerna kan olika tillverkares system ”prata” med varandra utan problem. Innan TCP/IP började dominera hade olika tillverkare ofta egna protokoll: Novell hade IPX, IBM SNA osv. Men som någon har sagt: *”Protokollkrigen är över, TCP/IP vann”*. Det är inte helt sant, det finns några proprietära (företagsägda) protokoll kvar. Men IP och TCP dominerar. Internet bytte till IP första januari 1983. Nätet hette fortfarande Arpanet då, och bestod av ca 400 datorer som alla bytte protokoll den dagen.

Ett protokoll kan ofta delas upp i huvud (header) och data. Huvudet har fördefinierade fält med bestämd betydelse. Det som skickas på nätet är ettor och nollor enligt någon konvention, och hur dessa tolkas beror på vilket protokoll som används.

Protokoll handlar alltså om både syntax och semantik. Och det är noga med detaljerna – är ett enda tecken fel så slänger mottagaren meddelandet utan att meddela sändaren. Med TCP märker sändaren det och skickar igen, men inte alltid med UDP.

5.5 ARP, UDP, TCP

5.5.1 ARP

Som jag nämnde i 5.3 används MAC-adresser på länknivån och IP-adresser på nätverksnivån. Dessa måste kopplas ihop på något vis. I version 4 av IP görs detta med ARP, Address Resolution Protocol.

ARP använder länklagrets broadcastadress för att skicka en fråga till alla enheter på LAN:et. En nystartad dator kan skicka ut ”Vem har IP 192.168.0.1?”, och den som känner igen den adressen svarar ”IP 192.168.0.1 finns på 00:1B:21:AF:B3:22”. Egentligen är det bara fält i ett ARP-huvud, men kollar man kommunikationen med t.ex. tcpdump ser man rader som liknar:

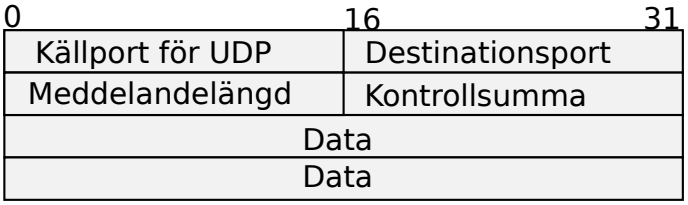
```
ARP, Request who-has 192.168.3.1 tell 192.168.3.3
ARP, Reply 192.168.3.1 is-at 00:1b:21:af:b3:22
```

Du kan kolla själv med:

```
sudo /usr/sbin/tcpdump -n -i eth1 arp
```

(Byt ut mot lämpligt interface, och ta bort -n om du vill slå upp namnen i DNS)

5.5.2 UDP



Figur 5.5: *UDP-header*

I figur 5.5 finns en header för ett av de enklaste protokollen, UDP (User Datagram Protocol). UDP ligger på transportlagret, ovanför IP. Huvudet består enbart av fyra fält som alla består av 16 bitar. Resten är data. Bit 0 till 15 anger källans port, bit 16 till 31 anger destinationsporten. En IP-adress leder oss till rätt dator, men för att bestämma vilket

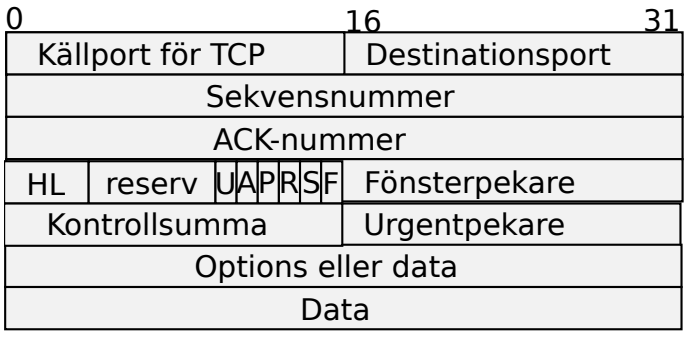
program på den datorn som ska ta emot datan används en abstraktion som kallas port. Portar är alltså egentligen bara ett 16-bitars tal (0-65535) som ingår i UDP-huvudet (eller TCP-huvudet för de protokoll som använder TCP som transportlager).

Utöver två portnummer anger UDP-huvudet meddelandets totala längd och en frivillig kontrollsumma.

Vissa protokoll använder standardportar, till exempel 53 för DNS. Källporten kan ha ett godtyckligt värde, men om destinationsporten är 53 vet mottagaren att paketet ska skickas vidare till programvaran som agerar namnserver. När namnservern svarar använder den källporten från sändarens UDP-paket som destinationsport, och givetvis den ursprungliga frågarens IP-nummer som destination på nätverksnivå. Totalt behövs fyra uppgifter för att identifiera en förbindelse: sändarens IP och port, plus mottagarens IP och port. Att man använder portar medför att varje dator kan utföra många olika tjänster. Detta brukar kallas multiplexing och demultiplexing.

5.5.3 TCP

UDP, eller IP, har ingen kontroll av att paketen kommer fram, eller att de kommer i rätt ordning, eller om de kommer dubbelt. För detta behövs TCP. Förkortningen står för Transmission Control Protocol. TCP är mera komplicerat än UDP, och jag går inte in på alla detaljer. Först visar jag headern i figur 5.6



Figur 5.6: *TCP-header*

Även TCP har portar för att veta vilken process informationen ska skickas till. En TCP-förbindelse kan unikt identifieras med ett fyrtal: IP-nummer och port på avsändare och mottagare, dessa kallas även sockelpar (socket pair). En TCP-förbindelse är alltid dubbelriktad, om det inte kommer data mer än åt ett håll kommer det ändå paket åt andra hållet som bekräftar vad som har tagits emot. Dessa kallas Ackar (Acknowledgements). I headern ser du en flagga som heter A eller ACK, den biten är satt om paketet är en ACK. Dessa paket kan även skicka annan data. TCP är en ström av bytes, sekvensnumret anger var i strömmen man är nu. IP är ett opålitligt protokoll, paket kan både försvinna, komma i fel ordning och komma dubbelt. Men det hanterar TCP genom omsändning och diverse timers.

Portnummer för vanliga tjänster är

Tabell 5.5: Portar

SSH	22
SMTP	25
HTTP	80
HTTPS	443

Flera kan du se med `less /etc/services`.

5.6 HTTP och SMTP

Många applikationsprotokoll är textbaserade. Det gäller för såväl HTTP som SMTP. Det går att simulera en webbläsare genom att ansluta till port 80 med telnet:

```
[ao@a ~]$ telnet www.df.lth.se 80
Trying 194.47.250.11...
Connected to www.df.lth.se.
Escape character is '^]'.
GET /

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Datorföreningen vid LU &amp;amp; LTH</title>
```

Vissa servrar kräver mer än GET, t.ex. version av http (1.1) och vilket hostname man ansluter till. Och ibland får man statuskod 302, att dokumentet är flyttat. Men att telnetta till port 80 är ett enkelt sätt att testa att en webbserver svarar.

5.7 IPv6

IPv6 är en nyare version av IP. Där är adresserna 128 bitar långa, och anges med hexadecimala siffror. Till exempel 2a00:1450:4010:c03::67

Två kolon anger att adressen innehåller enbart nollor däremellan. Den fullt utskrivna adressen för denna dator (ipv6.google.com) skulle bli
2a00:1450:4010:0c03:0000:0000:0000:0067 Alltså 8 grupper av 4 hexadecimala siffror. Men inledande nollor i varje grupp kan utelämnas och på max ett ställe kan nollor bytas ut mot kolon.

Med IPv6 är nätmasken ofta /64. Alltså är varje subnät mycket större än hela IPv4-Internet. Det gör att man inte behöver använda NAT.

I IPv6 är headern enklare än i IPv4. Broadcasting används inte, enbart multicast.

5.8 Lästips

Två lättlästa böcker på svenska är:

- Maria Kihl. *Datakommunikation : en inledande översikt*. Studentlitteratur, Häftad 2006, e-bok 2010.
- Maria Kihl & Jens A. Andersson. *Internet*. Studentlitteratur 2008.

Standardverk på engelska är:

- Douglas E. Comer. *Internetworking with TCP/IP.* Addison-Wesley 2005.
- Kevin R. Fall & W. Richard Stevens. *TCP/IP Illustrated.* Addison-Wesley 2011.