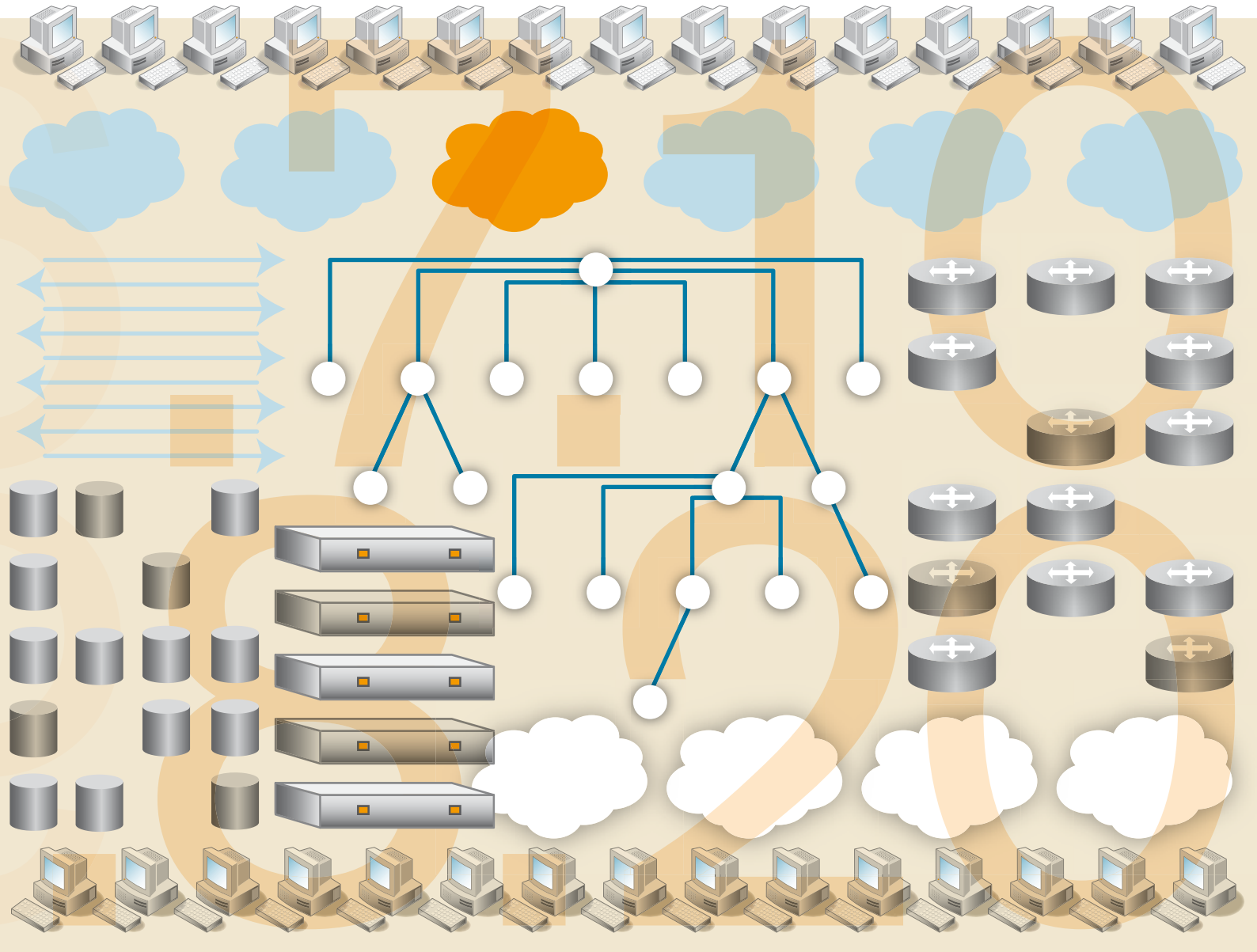
Håkan Lindberg

Introduktion till IP – Internet Protocol

En guide om hur Internettrafiken fungerar



Introduktion till IP – Internet Protocol

.SE:s Internetguide, nr 1
Version 2.0 2009

Författare: Håkan Lindberg
Redaktör: Lennart Bonnevier
Formgivning, layout och redigering: Gunnel Olausson/FGO AB
Omslagsillustration: © Camilla Laghammar
Renritning av figur: Camilla Laghammar
Andra upplagan, första tryckningen.
Tryck: Livonia Print, Riga, september 2009.
ISBN: 978-91-978350-0-8

.SE (Stiftelsen för Internetinfrastruktur) ansvarar för Internets svenska toppdomän. .SE är en oberoende allmännyttig organisation som verkar för en positiv utveckling av Internet i Sverige.

Besöksadress: Ringvägen 100 A, 9 tr, Stockholm
Brevledes på .SE Box 7399, 103 91 Stockholm
Telefon: +46 8 452 35 00
Fax: +46 8 452 35 02
E-post: info@iis.se
Organisationsnummer: 802405-0190
www.iis.se

Förord

Det här är ett utdrag på över hundrafemtio sidor ur en bok som är cirka tre gånger så lång. Och det slog mig att på en vanlig dator, PC, Mac eller Linux, är det bara tre – fyra saker man behöver ställa in för att IP ska fungera. På något vis har jag lyckats skriva tre hundra sidor om tre inställningar…

Varför skriver man en bok om IP idag? Jag fick frågan på en kurs för en kort tid sedan. Jag vet att det finns flera andra böcker om IP. Men flera bygger på gamla förlagor och jag tyckte det behövdes en uppdaterad variant. I grunden är IP fortfarande ett mer än 25 år gammalt protokoll men användningsområdet har ökat och ändrats. Adressöversättning som var tänkt att användas i begränsad och temporär utsträckning har blivit standard. Det finns flera tusen dokument, RFC:er, som beskriver de protokoll som vi ibland kort och gott kallar TCP/IP. En bok blir mer tilltillgänglig. Vidare behöver alla som arbetar med IP-teknik ha en god grundförståelse för nätverksbaserad säkerhet. Vi bör inte behandla säkerhet i några påhängda kapitel på slutet.

Den här guiden är liksom min längre bok avsedd att kunna användas som lärobok. Trots det har jag lagt in en del personliga värderingar. Jag har ägnat mitt yrkesverksamma liv åt data- och telekom och jag tycker fortfarande att det är roligt. Jag gillar IP och tycker att det är en fantastisk konstruktion.

En kritisk läsare kan uppfatta delar av denna guide som lite väl positiva till teknik. Jag granskar inte kritiskt vad det finns för hot och problem med Internet eller IP. Jag har själv läst om hot och problem med informationssamhället i flera decennier. Men jag tycker att det Internet och IP medför till största del är av godo. Idéer kan flöda fritt över landsgränser. I stort sett kan vem som helst bli sin egen publicist och skapa webbsidor eller bloggar, Internet har beskrivits som en motor för demokrati. Vi kan bygga fantastiska verksamhetssystem i webbläsare som blir oberoende av vilket operativsystem som används. Stora användargrupper som inte

haft råd att ringa utomlands kan använda IP-telefoni och ringa till andra världsdelar för en fast månadsavgift. Och problem med spam får inte överskugga den fantastiska möjlighet som e-post inneburit.

Jacques Vallee tar i boken ”Det osynliga nätet” från är 1982 upp två huvudsakliga scenarier för det samhälle som han ser vara på väg. En ganska mörk bild med övervakning, IT-system som kräver experter, myndigheter som begår allvarliga misstag på grund av en övertro på sina IT-system. Men han målar också upp en bild av system som blir lättare att använda och hur möjligheten att kommunicera kan föra människor närmare varandra. Lägg märke till att boken kom innan persondatorer, grafiska gränssnitt eller TCP/IP hade slagit i någon större omfattning. Av dessa två bilder anser jag att vi ligger närmare den positiva. De som ser faror, hot och problem med IP får tyvärr läsa en annan bok. Själv gläds jag åt en diskussion jag hörde mellan mina barn häromåret: ”Vad har man datorer till om man inte har Internet?”. Internet har blivit något självklart och viktigt för min och nästa generation.

Säkerhet

Den här guiden handlar till stor del om nätverksbaserad säkerhet. Jag har försökt att skriva in säkerhetsaspekter inom varje aktuellt kapitel. Min erfarenhet är att vi behöver ta med säkerhetsaspekter i allt vi arbetar med och säkerhetsarbetet blir en integrerad del av utveckling och implementering. På samma sätt som nycklar och ID-handlingar är en del av vårt vardagliga liv tror jag att vi måste få med grundläggande funktioner för säkerhet inom all IP-baserad nätteknik.

Vem ska läsa den här guiden?

Guiden är tänkt att vara av allmänt intresse för bredbandsanvändare, mindre företag, IT-intresserade etc. Fokus ligger på grundläggande funktioner för IP som arkitektur, funktion (och varför det inte alltid fungerar) samt översiktlig information om IP-baserade tjänster.

Målgruppen förutsätts ha datorvana och koll på ”bits och bytes”.

Men även bland datorvana användare finns det många som inte förstår varför subnätmasken ser ut som den gör, och varför man ibland kommer ut på Internet och ibland inte. Här hoppas jag att jag kan täppa till några kunskapsluckor.

Svenska, engelska, svengelska eller Swenglish?

Spårsmula eller cookie? Gateway eller router eller nätsluss? Jag har försökt att använda de begrepp som används i branschen med en viss övervikt åt datatermgruppens rekommendationer. Men ingen blir gladare av uttryck som ”standardväg” när vi i branschen pratar om ”default route”. Och uttryck som ”spårsmula” tycker jag bara förvirrar. Men jag slår gärna ett slag för fler svenska ord bara de fungerar bra, och undviker i denna text laptop, PDA etc.

Ett par ord har varit extra svåra att översätta. I IP-sammanhang använder man gärna begreppet ”host” om en dator eller nätutrustning som har implementerat IP. Jag har använt begreppet nod istället, en nod kan alltså vara en dator eller en router etc, begreppet säger inget om huruvida IP-paket skickas vidare. Paket är ett annat ord som är svårt. Det finns mer specifika begrepp som ram, datagram, PDU och SDU, men jag har valt ordet paket ändå och vid tveksamheter förtydligat som exempelvis IP-paket. Header fick stå kvar på engelska efter en del vånda, jag har helt enkelt aldrig hört någon i branschen säga ”IP-huvud”.

Tack till

Jag har skrivit på detta material i flera år och även hållit föredrag och kurser inom berörda områden. Till slut fick jag tid att sätta mig och skriva klart, till stor del tack vare ett stipendium från .SE:s Internetfond 2006. Följande personer har även hjälpt till med synpunkter, korrektur och glada tillrop: Yngve Sundblad KTH, Torbjörn Carlsson och Staffan Hagnell på .SE, Thomas Révész XEQ Systems samt Per Kjellberg B3IT AB. Tack till Camilla Laghammar och Gunnel Olausson som hjälpt till med illustrationerna respektive layout och redigering. Eventuella brister och felaktigheter i texten är dock givet-

vis mitt eget ansvar. Sista delen av texten tillkom samtidigt som familjen fick en ny kattunge: Truls. Så fort jag satt mig vid datorn har Truls spinnande lagt sig på tangentbordet och kommit med tilllägg som ”iihh8888”. Så jag bär fortfarande ansvaret men tillägnar denna guide Truls och skyller eventuella felstavningar på henne.

Sollentuna i december 2007

Förord till ny upplaga 2009

Tack till er som hört av er med uppmuntrande kommentarer och rättelser. Uppmuntrande kommentarer har som tur är övervägt antalet rättelser. Den nya upplagan innehåller flera nya avsnitt om IP version 6. De flesta bedömare är helt överens om att behovet av IPv6 ligger nära i tiden och kunskap om IPv6 behövs. Inför den andra upplagan har jag även ändrat och förtydligat några småsaker. Vad gäller Internets framtid skrev jag 2007 att vi kanske inte kommer att få vara anonyma på Internet så länge till. Här har det som bekant hänt en del med datalagringsdirektiv, IPRED och det som kallas FRA-lagen. Till den mer positiva sidan hör att mobilt bredband slagit igenom och blivit snabbare, billigare och mer lättanvänt. Ytterligare ett område där IP har blivit en möjliggörare.

Syftet med denna guide på drygt hundrafemtio sidor är fortfarande att utöka den till en hel bok som är tre gånger så lång. Arbetet fortskrider men det går långsamt då det får ske på min lediga tid.

Eftersom flera kommenterat kattungen Truls i förra förordet så måste även det avsnittet uppdateras. Truls sprang tyvärr bort innan sin ettårsdag. Istället fick familjen en ny kattmedlem: Maja. Maja har dessutom fått en kull kattungar men jag har blivit bättre på att hålla dem från tangentbordeiiii88888.

Sollentuna i september 2009
Håkan Lindberg

Hjälp oss att förbättra
Om du hittat fel eller har synpunkter på denna guide kan du sända dem till publikationer@iis.se

IP – grunder och arkitektur

- Kapitlet beskriver grunddragen i hur IP fungerar samt hur under- och överliggande nivåer fungerar. Arkitekturen i TCP/IP beskrivs och hur TCP/IP relaterar till OSI-modellen. Skiktade arkitekturer förklaras kortfattat.
- Lokala nätverk förklaras med fokus på modern Ethernetteknik. Samverkan mellan IP, LAN och WAN i form av protokollet ARP gås igenom.
- Kapitlet är tänkt att kunna läsas fristående.

I datorernas barndom fanns det knappast en egen bransch som kallades datakommunikation. Däremot har det alltid varit viktigt hur information kan överföras, ett fundament inom en dator är ju att information hämtas, bearbetas och lagras. Själva grunden för detta är informationsöverföring. Claude Shannon, som lade grunden för den matematiska teorin för hur datorer skulle kommunicera med sitt banbrytande verk ”The mathematical theory of communication” på 1940-talet, hade fokus på hur information ska överföras.

Idag är datakommunikation en etablerad bransch men den har förändrats helt på 20 år. Förr så handlade branschen mycket om hur olika system skulle kunna utbyta information via speciell utrustning: protokollkonverterare. Denna typ av lösning blev mer och mer komplex, om det fanns 15 olika system och ett nytt slog igenom på marknaden så behövdes ju 15 nya protokollkonverterare. Idag använder vi nästan alltid TCP/IP-protokollen och alla system kan utbyta information.

När alla system i stort sett pratar samma protokoll behövs sällan protokollkonvertering. Men om alla system kan kommunicera med varandra så blir ju det ett problem i sig. Ska vi ha kontroll på vilka system som pratar med varandra så krävs konfiguration och en helt ny bransch har uppstått: nätverkssäkerhet. Tekniskt sett kan min dator utbyta information med Pentagon och Säkerhetspolisen. Klart att de vill ha kontroll på hur jag kan komma åt deras system. Och jag kan

själv vara rätt intresserad av att begränsa deras möjligheter att komma åt min information. Det som skiljer våra system åt är ett par användarkonton med lösenord och kanske digitala certifikat. Vidare skiljs vi åt av ett regelverk, jag får inte ens försöka komma åt deras system och det finns förhoppningsvis även reglerat hur de får komma åt mitt.

Arkitektur och standarder

Standarder är viktiga. När jag på min dator startar en webbläsare och skriver in webbadressen www.bolag.se så förväntar jag mig att det ska fungera. Men för att det ska fungera krävs att en massa saker ska vara standardiserat. På något sätt sitter min dator ihop med webbservern så vi måste komma överens om hur kontakter och kablage ska se ut (fysisk nivå). Vi måste komma överens om adresser och namn så att jag hamnar på rätt ställe (nätverksnivå). Hur ska vi dela mediet med andra så att inte alla sänder på en gång (länknivån)? Webbsidan ska delas upp i en massa små paket. Hur stora ska de vara? Hur ska de hamna rätt? Hur ska de sättas ihop igen? Om ett paket kommer bort när ska det sändas om? De sista frågorna hanteras av transportnivån. Vi måste vara överens om hur bilder och tecken ska tolkas (presentationsnivå). Och eventuellt ska vi anpassa kodningstekniken efter förbindelsens kvalitet.

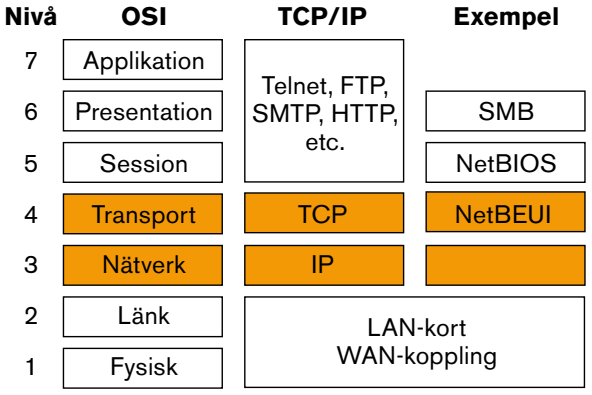
En fördel med en skiktad arkitektur är att vi bestämt på vilken nivå varje funktion ska utföras så att vi till exempel inte försöker komprimera data som redan är komprimerat.

En standard för kommunikation är OSI, Open System Interconnection. Arbetet har bedrivits av ISO sedan 1980-talet och syftar dels till att ange en referensmodell med sju nivåer och dels till att ange standarder inom alla sju nivåer. I praktiken har TCP/IP ersatt OSI som praktisk tillämpning av en öppen och oberoende standard för kommunikation. OSI har till stor del blivit just en referens.

TCP/IP har också blivit ett praktiskt bevis på att skiktade arkitekturer fungerar. Det tar lite tid att göra saker i olika skikt men resultatet blir flexibelt och modulärt. Genom att beskriva gränssnittet mellan till exempel nivå två och tre så kan vi enkelt byta ut ett

protokoll på nivå två eller tre mot att annat, gränssnittet är detsamma. Idén är inte konstig, få av oss vet hur en fax fungerar men vi vet hur vi ska använda den – hur gränssnittet fungerar.

Kostnaden för flera skikt blir en hierarki som kan bli omständlig. En programmerare bör inte skriva kod som går direkt ut på en kommunikationsport och börjar skicka ettor och nollor. Istället blandas protokoll från flera skikt in och ska lägga till overhead och komplexitet. Vi får overhead på overhead vilket tar plats från vår egentliga nyttolast. Att kommunicera över IP blir en kompromiss. Hur mycket information ska varje nivå tillåtas lägga till (hur lång header)? Hur långa paket? Hur ofta ska vi skicka kvittenser? Hur ofta ska vi sända om vid problem? Ska vi utveckla en teknik för att skicka till exempel audio över trådlös länk i 700 MHz-bandet så går det att hitta en mer effektiv teknik än att använda TCP/IP?

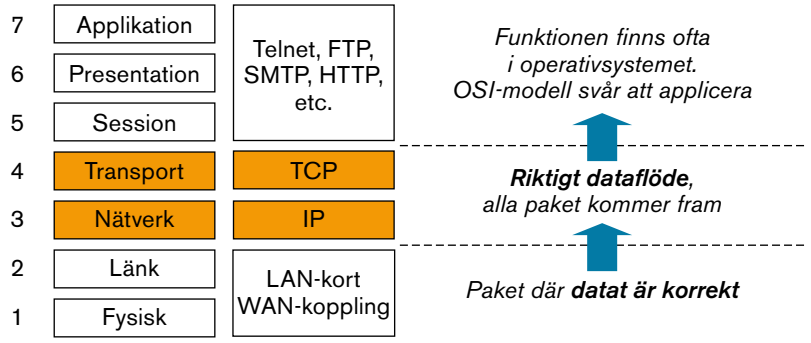


OSI-modellen och TCP/IP bygger bägge på skiktade arkitekturer. TCP motsvarar nivå fyra enligt OSI och IP nivå tre.

Men med TCP/IP får vi en mängd fördelar. Alla slags applikationer kan kommunicera över alla slags lokala nätverk och fjärrförbindelser. IP blir en gemensam nämnare för modern kommunikation. När vi kommunicerar via TCP/IP får vi en mängd funktioner på köpet för att hantera omsändningar, varierande fördröjning, adressering och routing. Ena gången kommunicerar vi med en server på vårt lokala nät, vi har gott om bandbredd, kort fördröjning och ett fåtal överföringsfel. Nästa gång kommunicerar vi med en server i en annan världsdel, för att hitta dit krävs drygt trettio routrar. Vi har ont om bandbredd, lång fördröjning och vart och vartannat paket försvinner på vägen.

TCP/IP ger en möjlighet för alla slags program att kommunicera över alla slags WAN (globala nät som ägs av operatörer) och LAN (lokala nätverk som Ethernet).

I modern datakommunikation kan IP lita på att nivå två hittar överföringsfel i varje länknivåpaket. Det förekommer bitfel (att en nolla felaktigt tolkas som en etta eller tvärtom) på grund av brus, svaga signaler och andra störningar. Men på nivå två lägger vi till en checksumma som beräknas baserat på de bitar som överförs. En mottagare beräknar checksumman och jämför med mottaget värde.



Tjänster från nivå två respektive fyra.

Om de inte stämmer så kastas paketet. Jämför checksumman med sista siffran i personnumret, den kan bara visa att något är fel och inte användas för att rätta felet.

På nivå tre, nätverksnivån, lägger vi in funktioner för en global adresseringsfunktion. Routrar arbetar på nivå tre med IP-adresser för att kunna skicka paketen till rätt mottagare.

Paket kan komma bort på nivå två och tre. För att kunna erbjuda ett korrekt dataflöde lägger vi till nivå fyra, transportnivån. TCP tar emot data från överliggande nivå, delar upp det i lagom stora paket och numrerar dem. Mottagande TCP-stack använder dessa sekvensnummer för att kontrollera att allt data kommit fram och vid behov begära omsändningar.

Två viktiga principer vid skiktade arkitekturer är också:

- Vi får inte hoppa över en nivå. Nivå fyra ska inte prata direkt med nivå två. Behöver detta ske får det ske via nivå tre.
- Sändare och mottagare kommunicerar med motsvarande nivå till exempel TCP till TCP. En TCP-stack på en maskin ska alltså inte ställa frågor till ett Ethernetkort på en annan maskin.

Dessa två är principer som man eftersträvar. TCP/IP är å andra sidan utvecklat av pragmatiker så om det behövs gör man någon gång små avsteg från dessa principer.

Allt över IP och IP överallt

IP arbetar på nätverksnivå, protokollets viktigaste uppgift är routing (vägval). Med hjälp av IP-adresser kan rätt mottagare och avsändare adresseras och routrar tittar på denna adress för att avgöra hur paket ska skickas vidare.

Routrar använder mottagarens IP-adress för att avgöra hur ett IP-paket ska skickas vidare.

När en router skickar ut ett paket på ett gränssnitt kan den få problem med paketets storlek. Routern kan ha tagit emot ett paket med längden 1 024 byte men den länknivå som utgör nästa hopp accepterar inte större paket än 512 byte. Routern måste vid sådana tillfällen dela upp paket i mindre fragment och motsvarande process kallas fragmentering. Varje fragment hanteras som ett eget paket men det märks upp så att mottagaren kan sätta ihop de olika fragmenten till ett IP-paket igen. Exakt hur fragmentering fungerar behandlas i kapitlet IP-nivån.

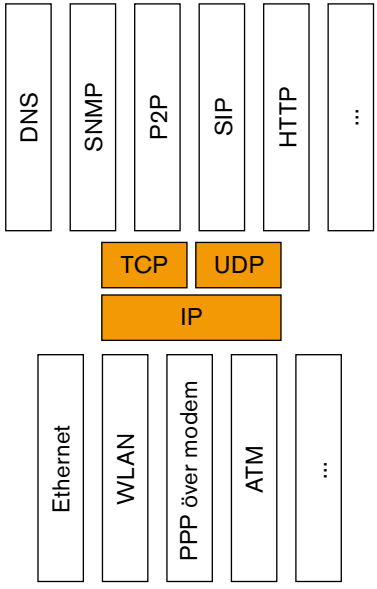
Den här avsnittet heter inte ”Allt över TCP/IP och TCP/IP överallt” utan ”Allt över IP och IP överallt”. Det finns tillfällen då vi inte vill använda TCP, protokollet UDP (User Datagram Protocol) har funnits med sedan starten. Nya transportnivåprotokoll har också utvecklats som till exempel SCTP. Och IP kan även bära andra nivå tre-protokoll. Men vi har mycket att vinna på att använda en gemensam adressering. IP handlar ju ytterst om att vi sätter en siffra på saker och denna siffra kan vi använda för att hitta fram. Mycket blir enklare om vi gör detta med en gemensam standard: IP.

IP har en struktur på sina adresser och det är detta som gör protokollet routbart. Vill vi bara ha en unik adress så kunde vi använt MAC-adresser (Media Access Control, se nästa avsnitt), men i så fall skulle routrarna fått arbeta hårt. Med IP-adresser behöver routrarna inte veta varje enstaka IP-adress, istället arbetar man huvudsakligen med nätnummer. En router som ska föra paket vidare till IP-adresserna 195.6.7.8 samt 195.6.7.250 använder normalt det faktum att de tillhör samma nät 195.6.7.0. Detta gör att routingtabellerna kan kortas ner betydligt. För att detta ska fungera måste IP-adresser delas ut och administreras centralt. Dessutom måste i princip en IP-adress som används ute på Internet vara unik.

IP-adresser är strukturerade så att de kan slås ihop till nätverksadresser, vilket ger hanterliga routingtabeller.

För att förstå IP:s genomslagskraft kan man dels hänvisa till att

En publik IP-adress delas ut centralt och är unik.



IP har en central roll om alla slags applikationer ska kunna kommunicera över alla slags WAN och LAN.

arkitekturen och protokollen är väldigt generella. Som många andra nätverkstekniker drar IP också nytta av något som brukar beskrivas som Metcalfe's lag nyttan ökar med kvadraten på antalet användare. Det här är ingen exakt lag (hur mäter man nytta?) men den förklarar varför en teknik som blir dominerande tar över en marknad helt. De som väljer att inte använda tekniken eller kommer in sent får ett kraftigt underläge. Det gäller inte bara IP utan även GSM och Ethernet.

Givetvis finns det brister i Internetprotokollen. Men vi kan jämföra TCP/IP med engelska. Ska vi titta på vilket språk som är första språk för flest antal människor borde vi lära våra barn kinesiska. Som alla naturliga språk innehåller engelska oregelbundenheter och konstigheter vilket det tar många år att lära sig. Engelska är dessutom ovanligt svårt att stava, ordlistor kom tidigt och man var inte alltid så konsekvent. Det lär finnas 14 sätt att stava sh-ljudet, varav några finns i orden shoe, sugar, passion, ambitious, ocean och champagne. Det lär finnas 38 stavningar som uttalas "air" eller "aire" eller "heir". Och så har till exempel ordgruppen "ough" fått en mängd olika uttal som through, though, thought, tough, plough och två – tre till. Trots det är engelska ohotat som internationellt språk för närvarande. Alla försök att hitta på nya språk för internationellt utbyte har misslyckats trots fördelar som fonetisk stavning och regelbunden grammatik.

Den främsta konkurrenten till IP som protokoll på nätverksnivå är IP själv. Idag använder vi IP version fyra (IPv4). Version sex (IPv6) finns framtagen och används i vissa delar av Internet. IPv6 är tänkt att kunna samexistera med IPv4 men det finns skillnader i hur protokollen fungerar. Det finns även principiella problem när Internet pratar två nätverksprotokoll. Hur ska vi till exempel

hantera om jag har IPv4 och vill skicka e-post till en server som bara har en IPv6-adress?

Funktioner på TCP- och UDP-nivå

IP-nivån ser till att paketen kommer fram till mottagaren, och protokollet ICMP (Internet Control Message Protocol) hjälper till om problem uppstår.

För att få ett korrekt dataflöde behövs dock lite mer. IP är förbindelselöst, vi skickar data till mottagaren utan att kontrollera om mottagaren är redo att ta emot data eller om den överhuvudtaget går att nå just nu. Om vi vill ha en förbindelseorienterad överföring får vi lägga till TCP (Transmission Control Protocol). TCP handskakar innan den egentliga dataöverföringen sker, tre paket utväxlas mellan klient och server och på så sätt kan vi vara säkra på att bägge parter är redo att överföra data.

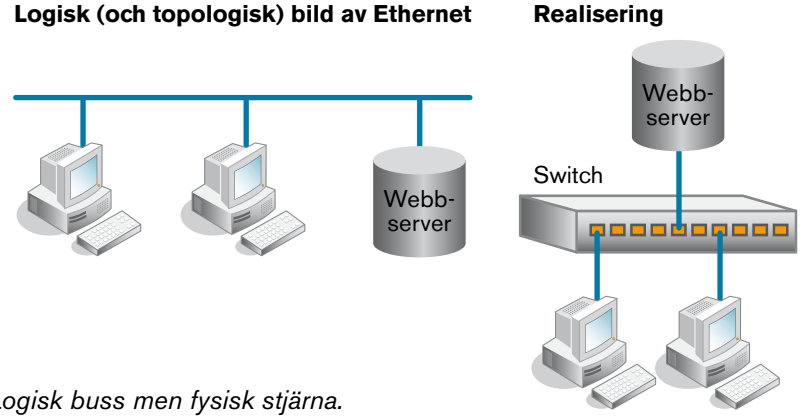
TCP lägger även till så kallade portnummer. Men hjälp av dessa kan flera applikationer dela på TCP/IP-stacken. Portnumret talar om vilken applikation (eller program eller process) som ska ta emot datapaketet. Samma server kan därför både vara webbserver (och svara på port 80) och telnetserver (port 23). Även på klienter används portnummer, det är därför vi kan surfa till flera servers på en gång eller överföra flera filer mot olika servrar simultant.

Det är även TCP:s uppgift att dela upp dataflödet i lagom stora delar och numrera dem. Med hjälp av dessa sekvensnummer kan mottagaren sedan meddela vilka paket som kommit fram och vilka som behöver sändas om.

Ibland behöver vi inte all denna funktionalitet. Då använder vi istället UDP (User Datagram Protocol). UDP är förbindelselöst och använder inga sekvens- eller kvittensnummer. UDP:s främsta uppgift blir bara att lägga till portnummer för att adressera rätt applikation. Ordet ”User” bär ingen speciell betydelse, protokollet lär från början ha haft namnet ”Unreliable” (otillförlitligt) men när TCP/IP började kommersialiseras tyckte man inte att den gamla betydelsen dög.

Lokala nätverk

Idag är Ethernet helt ohotat som teknik i lokala nätverk, det finns i princip inget alternativ förutom möjligen trådlösa nät – WLAN. Och även WLAN har en hel del gemensamt med Ethernet. ”Nätverkskort” har blivit ett vanligt namn på Ethernetkort. Ethernets framgångssaga beror delvis på att Ethernet är en enkel teknik som det har varit lätt att bygga på, de brister som finns med Ethernet är dessutom väl kända. Ett annat skäl är Ethernets fantastiska utveckling. Från 10 Mbit/s i slutet av 1970-talet till 10 Gbit/s.



Logisk buss men fysisk stjärna.

Ethernetkretsar filtrerar bort paket som är avsedda för andra mottagare än vår MAC-adress.

Vi ritar fortfarande Ethernet som en logisk buss, alla kan prata med alla. Vi väljer vilken station vi vill prata med genom att i Ethernetheadern ange avsändar- och mottagardress. Tekniskt sett kommer alla stationer att ha möjlighet att läsa paketets innehåll. Men Ethernetkorten filtrerar sedan bort de paket som är avsedda för en annan mottagare, på så sätt behöver inte operativsystem eller programvara arbeta med onödigt data. Klassiskt Ethernet (IEEE 10Base-5 och 10Base-2) använde en koaxialkabel som fysiskt medium, så den fysiska bilden av Ethernet stämde bra med den logiska.

Idag byggs Ethernet som stjärnformade nät. I centrum sitter en Ethernetswitch eller en hubb. Logiskt sett kan man gärna ha bilden av en buss framför sig. Kabeltyperna som används är så gott som ute-

slutande partvinnad kabel eller fiber och hastigheten finns i flera varianter från 10 Mbit/s till 10 Gbit/s. Den vanligaste standarden är 100Base-T (100 Mbit/s över partvinnad kabel, T som i twisted pair). Byter vi sista bokstaven mot ett F så avser standarden fiber. Byter vi första siffran mot 10 eller 1 000 så avser standarden 10 respektive 1 000 Mbit/s.

100Base-T kallas ofta ”Fast Ethernet”, det finns egentligen fler varianter. Den vanligaste standarden betecknas egentligen 100Base-TX.

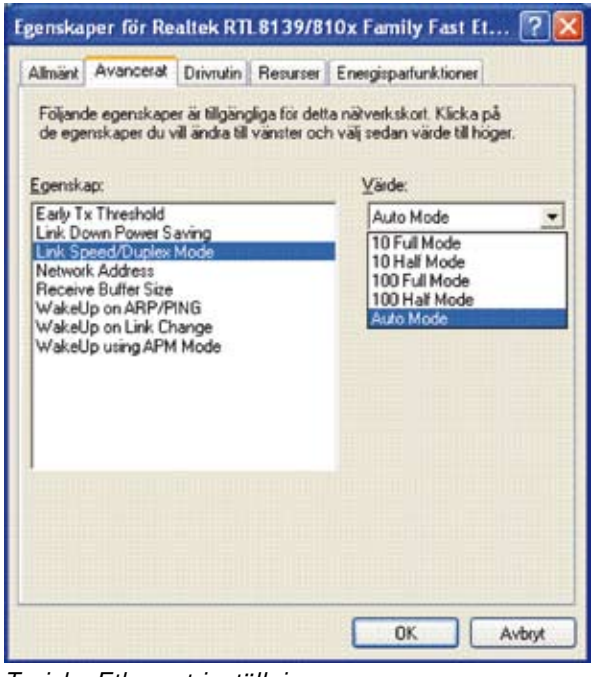
100Base-T kallas ofta Fast Ethernet och avser en standard om 100 Mbit/s över partvinnad kabel.

Att bygga ett Ethernetbaserat nätverk har blivit lätt. Standarder har medfört billig och driftsäker utrustning. Ethernet har gått från svårhanterligt rörmokeri med koxialkablage till ”plug and play” med färdigt partvinnat kablage. Att Ethernet är lätt att bygga innebär dock inte att det inte kan krångla.

1. Ethernet består av flera standarder. Mycket av detta löser sig genom autokonfigurering men det kan bli problem när all utrustning ska lyssna och ställa in sig. Ibland kan man vara tvungen att ställa in hastigheten på ett lokalt nätverk manuellt.

2. När partvinnat slog igenom fanns möjlighet till full duplex, ett kabelpar används för sändning och ett för mottagning. Samma princip som fall 1, utrustning ska klara av detta själv men vissa fabrikat fungerar dåligt ihop.

3. Partvinnat kablage fanns ursprungligen i två varianter: rakt kablage avsett att användas mellan en hubb och en dator och korsat kablage avsett för kommunikation dator till dator eller hubb till hubb. Många tillverkares utrustning kan idag ställa om automatiskt för korsat kablage men ibland fungerar det inte. Länklampan ska lysa i bägge ändar, annars kan man misstänka problem med kablaget.



Typiska Ethernet-inställningar.

4. Det är normalt med en viss paketförlust på Ethernet, men den bör bara ligga på någon eller några procent. På mer avancerad utrustning skiljer man på tidiga och sena (late eller out-of-windows collisions) kollisioner. För långt kablage eller för många kopplingar mellan switchar och hubbar leder till sena kollisioner, dessa ska inte förekomma på ett väl fungerande Ethernet.
5. Prispressen på Ethernetutrustning har lett till att utrustningen har minimala buffertar på varje port. Om två paket kommer in samtidigt måste utrustningen buffra paketet tills bussen blir ledig. Lågprisutrustning kan dessutom ha svårigheter med att hantera flera MAC-adresser på samma port. Detta gör att paket försvinner, och applikationer kan beskyllas för att fungera dåligt när felet egentligen ligger i det lokala nätet.

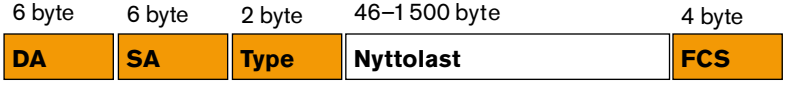
För att förstå varför paketförluster uppstår på Ethernet behöver vi titta lite noggrannare på den accessmetod som används. Accessmetoden anger hur stationerna ska dela på den gemensamma kanalen. För Ethernet heter denna metod CSMA/CD (Carrier Sense Multiple Access with Collision Detection). Tekniken är i princip enkel och påminner om hur (väluppfostrade) gäster delar på möjligheten att samtala under en middag. Alla har tillgång till en och samma kanal (Multiple Access). Lyssna först, om det är ledigt (Carrier Sense) så sänd. När du sänder så lyssna samtidigt, om avsänt meddelande skiljer sig från det du avlyssnar (Collision Detection) så beror det på en krock, någon annan sänder samtidigt. Avbryt i så fall genast, vänta en slumptid och försök på nytt.

Styrkan i den här algoritmen är dess enkelhet. Den kan lätt göras tusen gånger snabbare bara elektroniken hänger med. Den bygger heller inte på att någon station i nätet tar kontrollen utan alla stationer är jämlikar. Men den har en del inbyggda problem. Tekniskt sett kan vi aldrig garantera hur lång tid det tar innan ett meddelande kommer fram, det beror på vad de andra stationerna vill sända. Även om stationerna följer regelverket till punkt och pricka så kan två stationer börja sända i stort sett samtidigt innan de hinner uppfatta den andra stationens sändning. Därför uppstår kollisioner på Ether-

net. Den huvudsakliga lösningen på detta problem är att inte bygga för stora nät, tekniskt sett säger man att man minskar kollisionsdomänen, samt att se till att ha gott om bandbredd. Ju mer bandbredd desto snabbare går det att skicka meddelandet och desto mindre risk att någon annan vill sända samtidigt.

Ethernets paketformat

Det finns flera olika standarder för hur Ethernetpaket kan se ut, detta var ett komplext problem under 1980-talet med olika Ethernet-inkapslingar. Idag används främst Ethernet typ II som är enkel att förstå.



DA	Destination Address, mottagaradress
SA	Source Address, avsändaradress
Type	Detta fält anger vilken typ av nyttolast som bärs
FCS	Frame Check Sequence. Checksumma

Fälttyper inom Ethernet typ II.

Inom Ethernet typ II har vi en minimal header bestående av 14 byte uppdelade på destinationsadressen (DA), avsändaradressen (SA) samt typfältet. Datafältet består av 46–1 500 byte och kan innehålla viss utfyllnad (padding på engelska, det innebär att man fyller ut med det antal nollor som behövs). På slutet lägger vi till en 32-bitars checksumma och kommer totalt upp i en paketlängd om 64 till 1 518 byte. Ethernetpaket har alltså både en minimal och maximal längd.

Typfältet används för att deklarera vilken typ av datainnehåll ramen innehåller. Fältet består av två byte. Till exempel betyder 0x0080 att innehållet är IP.

Både avsändar- och mottagaradresserna består av 6 byte eller 48

bitar. Den här typen av adresser kallas MAC-adresser där MAC står för Media Access Control, även begreppet fysisk adress används. En MAC-adress kan till exempel vara 00-50-DA-69-D7-35.

En tillverkare av Ethernetkort registrerar sig hos organisationen IEEE och tilldelas en egen nummerserie (de tre första byten). Sedan ansvarar tillverkaren för att varje kort eller chip får ett unikt löpnummer, de sista tre byten. Varje MAC-adress är alltså unik och kan användas för att adressera just ett unikt Ethernetkort, ett WLAN-kort, ett Token Ring-kort eller en Bluetooth-enhet.

Varje Ethernetkort har en unik MAC-adress om 48 bitar som till exempel 00-50-DA-69-D7-35. Då de första tre byten är specifika för tillverkaren (3COM i detta fall) skrivs de ibland på formatet 3COM-69-D7-35.

Ethernet typ II har som sagt en väldigt enkel header. Det finns tillägg till standarden för att hantera olika logiska nät (VLAN) och prioriteringsfunktioner. Standarderna för dessa kallas 802.1Q respektive 802.1p.

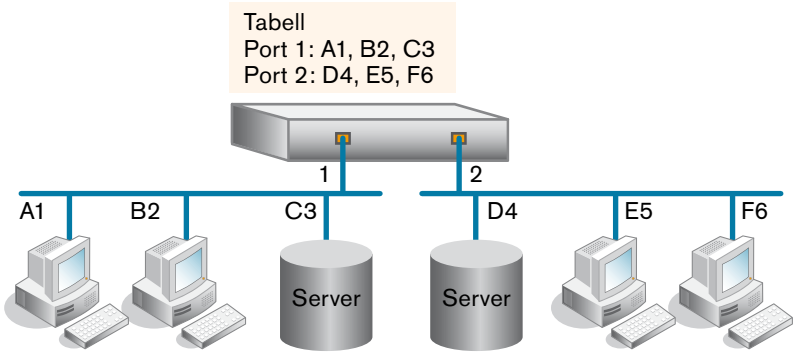
Ethernethubbar och Ethernetswitchar

En nackdel med äldre Ethernetarkitekturer som de koaxialbaserade 10Base-2 och 10Base-5 blev uppenbar när näten blev mer spridda och mer omfattande. Ett dåligt nätverkskort eller en glappande kontakt kunde drabba ett stort antal användare eftersom signalen inte kom vidare. Nätet blev aldrig bättre än dess svagaste länk.

En annan typ av lokalt nätverk, Token Ring, hade också problem med driftsäkerhet och där hade man löst det genom att bygga en centralt placerad enhet mitt i nätet: ett nav (hub). Nätverkskort eller kablage som fungerar dåligt kopplas bort av hubben. Konceptet spreds även till Ethernet. Hubben kan dessutom fungera som förstärkare (repeater) innan signalen skickas ut på varje enskild nätverksport.

Ethernet blev snabbt populärt och näten blev större och större. Ju större nät desto mer sannolikt att två noder vill sända samtidigt och risken för kollisioner ökar. En lösning på detta blev så kallade Ethernetbryggor. En brygga bygger upp tabeller genom att titta på avsändaradresser, på så sätt lär den sig vilka noder som sitter på respektive port.

Fördelen med en brygga är att A1, B2 och B3 kan kommunicera



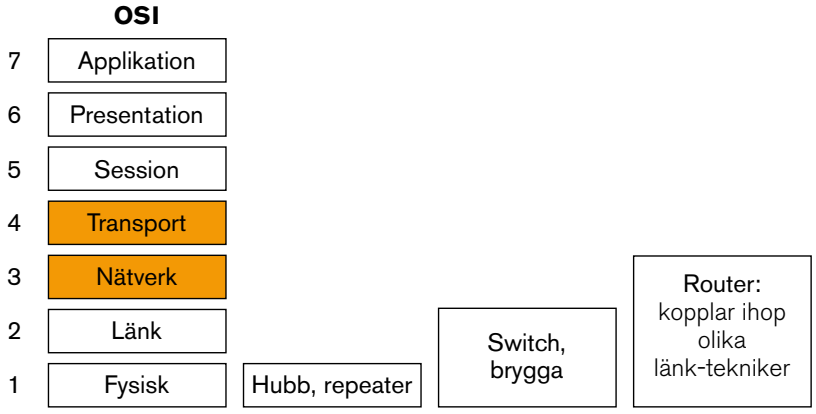
En brygga bygger själv upp tabeller med vilka adresser som sitter på respektive port.

för fullt utan att det stör det högra nätet. Vi får två separata så kallade kollisionsdomäner. Om däremot A1 vill skicka data till E5 så ser bryggan att detta paket måste vidareförmedlas. Men för noderna är det helt transparent vilket segment noderna sitter på.

I början av 1990-talet fanns multiportsbryggor men de var dyra. Tillverkaren Kalpana började marknadsföra sina multiportsbryggor som ”Ethernetswitchar” med stor framgång då priset per port var förhållandevis lågt. Switcharna lärde sig MAC-adresser på samma sätt som bryggor men man införde också ett snabbare sätt att skicka Ethernetpaket vidare. Bryggor läste in hela paketet och kontrollerade checksumman innan det skickas ut, så kallad store-and-forward. Men tekniskt sett kan vi skicka ut paketet så fort vi läst av Ethernetheadern eller till och med bara mottagaradressen. En vanlig teknik kallas för fragment free, då läser switchen in 64 byte innan det skickas ut. Vid 10 Mbit/s skulle det ta cirka 1,2 millisekunder att läsa in 1 518 byte, att läsa in 64 byte tar istället cirka 50 mikrosekunder. En avsevärd förbättring. Observera att en modern switch också måste kunna hantera store-and-forward för att klara olika hastigheter på respektive port.

Idag är Ethernetswitchar mycket billiga. De har i princip slagit ut hubbar från marknaden. Även utrustning som säljs som hubbar

är i teknisk mening ofta switchar. Ur säkerhetssynpunkt är switchar att föredra då de lär sig vilken eller vilka MAC-adresser som sitter inkopplade på respektive port och bara skickar information till denna adress (samt multi- och broadcast), detta gör det svårare att avlyssna trafik till och från andra stationer. I labbsituationer kan dock hubbar vara att föredra då all trafik skickas ut på alla portar, det gör det lättare att analysera trafik.



Genom OSI-modellen kan man tydliggöra skillnaden mellan hubbar, switchar och routrar.

Man kan tydliggöra skillnaden mellan hubbar, switchar och routrar genom OSI-modellen. En hubb jobbar på nivå ett genom att göra en ny kopia på paketet, switchen arbetar på nivå två genom att använda MAC-adresser. Routern jobbar på nätverksnivå med IP-adresser.

IP och samverkan med länknivån

För att adressering ska fungera ska varje nod i nätverket ha en unik adress. På länknivå är MAC-adresser vanligast. Ger du din Windows-dator kommandot ”ipconfig” så visar den vilken eller vilka MAC-

adresser som används. MAC-adresser kallas ibland även fysiska adresser.

```
Ethernet-kort Local Area Connection:

        Anslutningsspecifika DNS-suffix .
        Beskrivning . . . . . . . . . . . : Realtek RTL8139
        Fysisk adress . . . . . . . . . . : 00-0B-5D-C2-2F-68
        DHCP aktiverat  . . . . . . . . . : Nej
        Autokonfiguration aktiverat  . . . : Ja
        IP-adress . . . . . . . . . . . . : 192.168.3.245
        Nätmask . . . . . . . . . . . . . : 255.255.255.0
        Standard-gateway  . . . . . . . . : 192.168.3.1
        DHCP-server . . . . . . . . . . . : 192.168.3.1
        DNS-servrar . . . . . . . . . . . : 192.168.3.2
```

Ett exempel på resultat från kommandot "ipconfig /all". Vi kan se både IP- och MAC-adress. MAC-adresser kallas även fysiska adresser.

Vi skulle kunna använda MAC-adresser för att bygga ett globalt nätverk men en fördel med IP-adresser är att de har en hierarkisk uppbyggnad, den första delen av adressen används för att peka på vilket nätverk adressen hör till.

Jämför IP-adresser med postadresser. Vi skulle kunna använda personnummer för att hitta rätt mottagare, men det blir ett hårt arbete för postverket. Istället använder vi gatuadresser, jämför med IP-adresser. Det är bara den sista instansen som har kontroll på exakt var ”Kungsgatan 12 i Stockholm” ligger, de som ligger tidigare i kedjan tittar mer på Stockholm och Kungsgatan. Detta gör att adresseringstabellernas storlek kan hållas nere.

Det finns tre principiellt olika sätt att adressera en eller flera mottagare. Begreppen används lite olika i olika tekniker men principerna är de samma. Dels kan vi adressera en enstaka nod, detta kallas för unicast, och är den vanligaste adresseringsmetoden. Sedan kan vi adressera alla noder i ett nätverk, detta kallas broadcast.

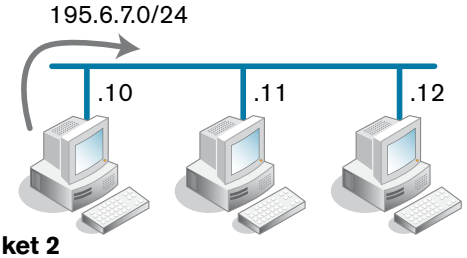
Broadcast används mycket i lokala nätverk, en nod kan fråga efter alla servertjänster av en viss typ. En nod kan till exempel fråga efter alla skrivarservers, alla noder på nätet kontrollerar om de ska svara. De som svarar skickar tillbaka information innehållande namn och typ av tjänst. Detta visas som en lista i klienten, och det är sedan slutanvändaren som väljer vilka server som ska användas. Kommunikationen övergår sedan till vanlig unicast.

Broadcast fungerar inte på Internetnivå, det skulle bli svårhanterligt om en nod frågar efter ”alla skrivare”. Därför begränsar alla routrar broadcastfrågor så de inte vidarebefordras.

Ett mellanting mellan dessa två tekniker är multicast. I multicast adresserar vi flera noder, alla noder som tillhör en viss multicastgrupp. I multicasting finns också flera klasser med olika räckvidd (scope) som exempelvis lokal länk, organisation eller global.

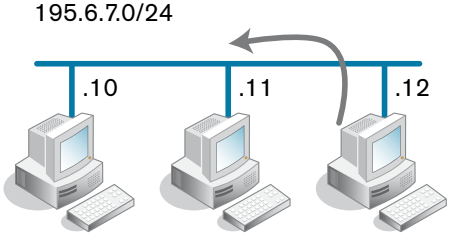
Paket 1

S-IP : 195.6.7.10	D-IP : 195.6.7.12
SA : 3COM-11-22-33	DA : FF-FF-FF-FF-FF-FF



Paket 2

S-IP : 195.6.7.12	D-IP : 195.6.7.10
SA : Netgear-22-33-44	DA : 3COM-11-22-33



ARP-handskakning. Begreppen i denna bild används genomgående: SA och DA står för Source och Destination Adress (MAC-nivå). S-IP och D-IP står för Source respektive Destination IP. Handskakning visas ovan på nivå två och tre.

ARP

Om en dator ska skicka IP-paket till en annan behöver den ta reda på mottagarens MAC-adress. Annars skulle paketet filtreras bort av mottagarens länknivå. Protokollet för att lösa detta problem kallas Address Resolution Protocol, ARP och används bland annat på Ethernet, Token Ring och WLAN. En motsvarande mekanism finns även inom Frame Relay för att finna så kallade DLCI-värden, där den kallas Inversed ARP. Inom nästa version av IP, IP version 6, har man löst detta på ett annat sätt.

Noden med IP-adress 195.6.7.10 behöver ta reda på MAC-adressen för nod 195.6.7.12. Den skickar då en så kallad ARP Request med en mottagaradress på Ethernetnivå där alla bitar satts till 1 (FF-FF-FF-FF-FF-FF). Alla Ethernet-

kort tar då emot paketet och granskar innehållet, en ARP Request. Den mottagare som har rätt IP-adress besvarar med en ARP Response och skickar då tillbaka sin MAC-adress till avsändaren.

Resultaten cachas av bägge noderna. När nod 195.6.7.12 får ARP-frågan så mellanlagrar den MAC-adressen tillhörande 195.6.7.10. På motsvarande sätt kommer nod 195.6.7.10 att mellanlagra resultatet, i detta fall MAC-adressen tillhörande 195.6.7.12. Bägge noderna tar för givet att ARP-handskakningen efterföljs av kommunikation mellan noderna, så cachen sparas i ett par minuter.

```
C:\Documents and Settings>arp -a
Interface: 192.168.111.2 --- 0x3
  Internet Address      Physical Address      Type
  195.6.7.12            00-90-7f-22-33-44     dynamic
```

Du kan se innehållet i arp-cachen genom kommandot ”arp -a”.

Med kommandot ”arp -a” visas innehållet i arp-cachen. Du kan även testa vilka växlar som finns till kommandot. Till exempel finns möjlighet att lägga in fasta mappningar mellan IP- och MAC-adresser.

På nästa sida visas resultatet av en LAN-analys från en ARP-fråga och motsvarande svar. Analysen har gjorts med programmet Wireshark. I denna guide används beteckningen ”0x” för att beteckna hexadecimala tal. Denna beteckning används även av Wireshark som framgår av exemplet på nästa sida (se till exempel fältet Type). Lägg märke till att Ethernet version II används och att typfältet anger att innehållet är ARP. Lägg också märke till hur Wireshark gör en översättning från de tre första byten i en MAC-adress till vilken tillverkare som registrerat denna serie.

Paket 1
Ethernet II,
Destination: Broadcast (ff:ff:ff:ff:ff:ff)
Source: **Fujitsu_c3:2f:98 (00:0b:5d:c3:2f:98)**
Type: ARP (0x0806)
Address Resolution Protocol (request)
Hardware type: Ethernet (0x0001)
Protocol type: IP (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: request (0x0001)
Sender MAC address: Fujitsu_c3:2f:98
Sender IP address: 195.6.7.10
Target MAC address: 00:00:00:00:00:00
Target IP address: 195.6.7.12

Paket 2
Ethernet II
Destination: Fujitsu_c3:2f:98 (00:0b:5d:
c3:2f:98)
Source: 3comEuro_9f:4a:fa (00:0f:cb:9f:4a:fa)
Type: ARP (0x0806)
Address Resolution Protocol (reply)
Hardware type: Ethernet (0x0001)
Protocol type: IP (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: reply (0x0002)
Sender MAC address: 3comEuro_9f:4a:fa
Sender IP address: 195.6.7.12
Target MAC address: Fujitsu_c3:2f:98
Target IP address: 195.6.7.10

Exempel på en ARP-fråga och motsvarande svar. Trafiken har spelats in med en LAN-analysator.

ARP och säkerhet

ARP är en grundläggande mekanism i moderna nätverk och på ett stort nätverk förekommer flera ARP-frågor i sekunden. Mekanismen är stabil och mycket transparent, vi behöver sällan fundera på den. Ändå finns det principiella säkerhetsproblem med funktionen.

En ARP-fråga skickas ut till alla noderna på ett nätverk. En nod som innehåller skadlig kod eller ett hackarverktyg kan välja att svara på ARP-frågor och ange sin egen MAC-adress som den eftersökta. Avsändaren kommer då att skicka alla paket till denna nod, förutsatt att den hann svara innan rätt nod skickade sitt svar. Noden kan nu agera mellanhand och analysera och kopiera alla paket till och från avsändaren.

IP version 6 använder Neighbor Discovery

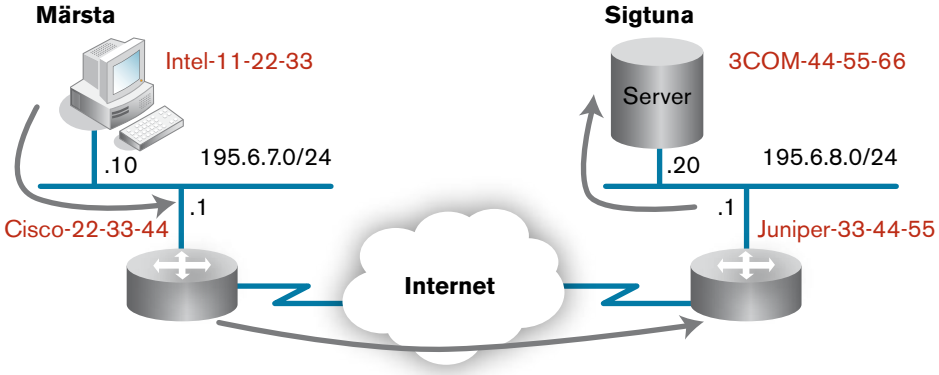
IP version 6 (ofta förkortat till IPv6) har ersatt ARP med en funktion som heter Neighbor Discovery. Principiellt bygger den på liknande mekanismer, noden söker MAC-adressen. Noden får börja med att fråga och den erhåller sedan ett svar. Funktionen skiljer sig lite åt beroende på om man letar efter en vanlig grannod eller en router. Inom IPv6 annonserar routrar att de finns genom ett paket som kallas Router Advertisement. Detta är ett paket på IP-nivå men som option skickar routern även med sin aktuella MAC-adress. På det sättet kan mottagare lagra även den MAC-adress som ska användas.

ARP bygger på broadcast och inom IPv6 använder man bara multicastteknik. Multicast är en teknik som kan göras mer flexibelt. Man kan ställa omfattningen (*scope*) på IPv6-nivå till att vara den lokala länken (*link local*), vilket då liknar broadcast. Men omfattningen kan även vara hela organisationen eller aktuellt företagsnät.

En nod som söker efter MAC-adressen till IPv6-adressen 2001:1234:5678:9:204:75ff:fec2:c188 skickar ut en multicast och frågar efter en mer generell adress inom IPv6 som kallas för Solicited Node. Den adressen skapas genom att man tar de sista sex hexadecimala siffrorna (24 bitar) av IPv6-adressen. Före det sätter man sedan strängen ”FF02:0:0:0:0:1:FF”. De första fyra siffrorna FF02 anger just att detta är en multicast med omfattningen satt till ”link local”. Även på Ethernetnivå är detta inte en broadcast utan en multicast (de två första byten sätts då till 33), hela mottagaradressen sätts i just detta fall till 33-33-FF-C2-C1-88. Det första paketet som går iväg från den frågande enheten kallas för Neighbor Solicitation. På samma sätt som routern gör vid Router Advertisement så skickas den egna MAC-adressen med som en option. Den nod som har rätt MAC-adress kommer att granska paketet och sedan undersöka frågan och se att en annan nod frågar efter dess MAC-adress. Den svarar då med en så kallad Neighbor Advertisement till rätt MAC-adress (och även till rätt IP-adress). Och den egna MAC-adressen skickas med som en option.

Samverkan mellan TCP, IP och Ethernet

Vi ska titta på hur nivå två till fyra samverkar vad det gäller adressering. Om det verkar krångligt så var lugn, vi kommer tillbaka till samtliga funktioner lite senare. I Märsta vill en klient skicka över kommandot ”GET / HTTP/1.0” till en webbserver i Sigtuna (detta skulle få webbservern att skicka HTML-text från sin webbsida). Webbservern lyssnar på port 80, och klienten skickar från port 1234. Det är mekanismen portnummer på transportnivån som gör att en nod kan hantera en mängd dataströmmar mot andra noder simultant, den kan se vilket program som ska ha datainnehållet.



S-port: 1234	D-port: 80	**TCP**	S-port: 1234	D-port: 80
S-IP: 195.6.7.10	D-IP: 195.6.8.20	**IP**	S-IP: 195.6.7.10	D-IP: 195.6.8.20
SA: Intel-11-22-33	DA: Cisco-22-33-44	**MAC**	SA: Juniper-33-44-55	DA: 3COM-44-55-66

Adressering på Ethernet-nivå och IP-nivå frigör sig från varandra. MAC-adresserna används bara lokalt.

Lägg märke till att adresseringen på länknivå (MAC) och nätverksnivå (IP) frigör sig från varandra. IP-adresserna är globala och de är de samma över alla typer av länkar. MAC-adresserna används bara lokalt, trots att vi ska till Sigtuna så är det routern som finns på det lokala nätverket i Märsta som vi adresserar. MAC-adresser är alltså inte intressanta för ett brandväggsskydd medan IP-adresser och portar kan användas.

Från paketförmedling till Internet

- Det finns flera källor på Internet om hur Arpanet och Internet uppstod. Det här kapitlet beskriver främst hur behovet av TCP/IP uppstod, och de designprinciper som ligger bakom.
- Utan Arpanet inget TCP/IP och utan TCP/IP inget Internet. Därför behandlas Arpanet uppkomst kortfattat. Även standardiseringsprocessen i form av RFC:er tas upp. Här tas även upp även hur den svenska grenen av Internet kom igång på ett tidigt skede.
- Sista delen av kapitlet behandlar kortfattat framtiden för TCP/IP och Internetprotokollen. Kapitlet kan läsas fristående.

Plötsligt hade vi fått något som hette Internet. En del sa att det var en fluga som snart skulle vara förbi. Andra sa att det skulle förändra världen.

En sak är säker. Utan TCP/IP hade vi inte haft Internet. Vi kanske skulle ha något annat protokoll som haft liknande funktioner men risken finns att vi skulle haft en massa öar av datorer och nät som inte var ihopkopplade. Varför blev det då TCP/IP som fick utgöra det klister som kopplade ihop nätverk, datortyper och program? Och hur länge till håller ett protokoll som har drygt 25 år på nacken? Är det inte dags att ersätta TCP/IP med något modernare?

För att göra det hela tydligare bör vi skilja på protokollfamiljen TCP/IP och Internetprotokollen (Internet Protcol Suite). Innan Arpanet konverterade till TCP/IP fanns ett nät av nät och en mängd program som kunde prata med varandra. Program som FTP och Telnet är äldre än själva TCP/IP. Självklart är det praktiskt om det finns en standard för hur program ska fungera för att överföra filer, skicka e-post, remote login, testa förbindelser med mera. Internetprotokollen finns standardiserade och dessutom finns det ofta källkod och open-source exempel att hämta. Gillar man inte open-source av någon anledning finns även färdiga program och kommersiella distributioner. Kopplingen mellan TCP/IP och en mängd standard-

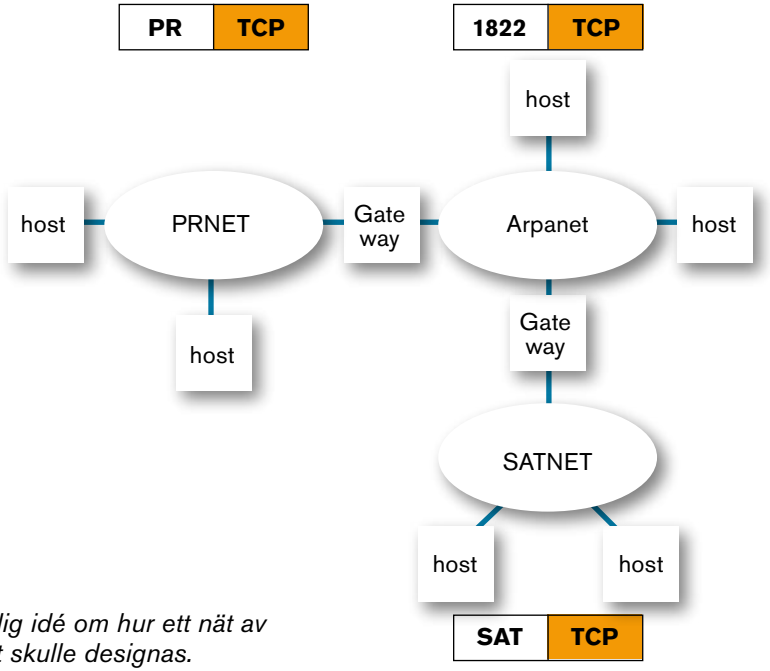
program hänger kvar än idag. Om vi säger att en dator har TCP/IP förväntar vi oss också att den har ett antal program. Men kopplingen är inte självklar och kommer säkert att förändras i framtiden.

Standardapplikationerna ovanpå TCP/IP är viktiga och har varit drivande för införandet av TCP/IP. Införandet av e-post har varit ett skäl att installera TCP/IP, det finns andra e-postsystem men med SMTP (Simple Mail Transfer Protocol) började e-posten bli global och kunna koppla ihop privatpersoner, universitet och företag. En standardiserad metod för att övervaka nätverk och noder har varit ett annat skäl. Då använder vi protokollet SNMP (Simple Network Management Protocol) ovanpå UDP ovanpå IP. I början av 1990-talet kom webben och vi fick ytterligare ett skäl att införa TCP/IP.

När TCP/IP utvecklades hade inte persondatorerna slagit igenom, de kom i början på 1980-talet. TCP/IP var främst avsett för användning på Arpanet, PRNET och SATNET. Vint Cerf lär själv ha sagt att om han hade vetat hur stort Internet skulle ha blivit hade han gärna velat utveckla en ny version av TCP/IP.

Huvudarkitekterna bakom TCP/IP var två personer: Vint Cerf och Bob Kahn. De började arbetet under 1970-talet. Ungefär samtidigt utvecklades de så kallade OSI-protokollen och IBM hade gått i bräschen för idén med skiktade arkitekturer genom sin standard Systems Network Architecture, SNA. Världen hade helt enkelt behov av att koppla ihop datorer och program på standardiserade sätt. Alternativet var att bygga nya så kallade konverterare varje gång två datormärken eller -modeller skulle kopplas samman. Cerf och Kahn arbetade flera år med att sätta samman en blandning av dåtidens senaste landvinningar inom datametoder och beprövad teknik.

En tanke med TCP, som protokollet hette från början, var att koppla ihop tre paketförmedlade nät. PRNET var ett nätverk för paketförmedling över kortvågsradio. SATNET var ett nätverk för paketförmedlad trafik över satellit. Och på Arpanet satt datorer som det var intressant att komma åt. Dessa tre nät har olika egenskaper. Över radio har vi ont om bandbredd, över satellit har vi lång fördröjning men ganska mycket bandbredd och på Arpanet hade man gott om bandbredd och kort fördröjning. Ett protokoll som hanterar alla dessa nät blir en utmaning och självklart även en kompromiss. Har vi ont om bandbredd vill vi ha korta paketlängder och en minimal header, men avancerade funktioner kräver en header för att upptäcka borttappade paket etc. Lång fördröjning gör att vi inte kan kvittera varje paket för sig. (En vanlig webbsida idag kan vara hundratals paket. Skulle vi kvittera dem var för sig från en sajt via



Tidig idé om hur ett nät av nät skulle designas.

satellit skulle det ta säg 400 × 500 ms = 200 sekunder.) Lösningen på det här och liknande problem var att göra TCP adaptivt. Man mäter hur lång fördröjning man har och anpassar trafiken därefter, på motsvarande sätt anpassas trafikmönstret efter hur ofta man tappar paket.

En annan princip var att inte störa de protokoll och adresseringsprinciper som fanns på varje nät. De var optimerade för sitt syfte. För att koppla ihop nätverken skulle en enhet som kallades gateway användas. Varje gateway skulle uppträda som en vanlig nod i respektive nätverk, men paketet som den skickade skulle sedan ha en ny header: TCP. Principen med en ny adresseringsstruktur finns kvar i TCP/IP, med IP får vi en ny global adresseringsmekanism oberoende av om protokollet under bara kan hantera några hundra noder.

TCP byggde också på antagandet att de underliggande näten är otillförlitliga. De noder som är slutgiltiga avsändare och mottagare

håller reda på vilka paket som kommit fram och vilka som behöver sändas om. Enstaka gateways och länkar kan tappa paket. De datorer som använder TCP är jämlikar, det vill säga det finns inget ”master/slave-” eller ”client/server”-förhållande. Alltså kan inte en dator sända ett paket till en annan och vara säker på att den mottagande datorn vill ta emot det. På engelska använder vi begreppet peer-to-peer.

Cerf och Kahn hade en ganska ödmjuk inställning till vad protokollet skulle användas till. Det fick andra avgöra. TCP såg till att dataströmmen som kom fram i ena ändan motsvarade vad som skickats i den andra. Protokollet är inte utvecklat för någon speciell applikation eller länk. De lade in funktioner för en avancerad flödeshantering med sekvensnummer och kvittenser, kryddat med en funktion som kallas sliding windows. En tanke var också att man skulle kunna prioritera mellan olika trafiktyper så TCP/IP när det rullades ut hade möjlighet att märka upp paket med ett fält kallat Type of Service (ToS).

Idén om skiktade arkitekturer hade spridit sig under mitten av 1970-talet och runt 1978 valde man att dela upp funktionera i TCP. De funktioner som behövdes för att välja väg och skicka ett paket vidare plockades ut och fick utgöra IP, Internet Protocol. Detta medförde att gateways (routrar) bara behövde kunna IP, medan TCP byggdes in i avsändande och mottagande nod.

Runt 1981 fick företaget Bolt Beranek and Newman pengar från ARPA (Advanced Research Projects Agency) för att skriva en ny implementation av TCP/IP. Programmeraren Bill Joy skriver koden för BSD Unix och introducerar begreppet socket, som kommer att bli en de facto-standard för gränssnittet mellan applikation och transportnivå. Joy börjar sedan på Sun, och Suns arbetsstationer får med sig TCP/IP i sitt operativsystem baserat på BSD Unix. Samtidigt som Unix-baserade arbetsstationer börjar komma ut på marknaden börjar persondatorer se dagens ljus och med dem kommer behovet av snabba nätverk. Ethernet blir det dominerande snabba lokala nätverket. Och till slut vill vi koppla ihop alla lokala nätverk. Internetworking blir ett begrepp.

TCP/IP har en motig start. Internationella Standardiserings Organisationen ISO jobbar samtidigt med sitt förslag på en skiktad

arkitektur: OSI. Trots att deras arbete inte är klart får de med sig många stora företag som Digital, HP och IBM. Vid offentliga upphandlingar är det OSI som gäller i Europa och USA. OSI är mer akademiskt uppbyggt och syftar även till att standardisera ett antal applikationer. Arbetet med OSI tar tid och TCP/IP visar sig fungera bra. I upphandlingar börjar man så småningom styra leverantörer mot att ”stödja OSI men tills vidare kan TCP/IP användas”.

Arpanet

Datorer användes från början mest för beräkningar och till att skapa tabeller (så vi kunde räkna vidare för hand). På svenska kallade vi det dator men på flera språk kom namnet att färgas av just möjligheten att beräkna. Men redan i början av 1960-talet fanns det folk som såg att de skulle kunna göra något annat. J.C.R. Licklider var en av dem. Han var en tvärvetenskapligt intresserad psykolog som kommit i kontakt med datorer tidigt. Han var en av de tidiga cheferna på ARPA. Licklider såg nya möjligheter med datorer och skrev ett antal artiklar och publikationer om nätverk och datorer. Han skrev till exempel att den politiska processen skulle kunna bli en enda stor telekonferens. Och han förutsåg hur betydelsefull kommunikation skulle kunna vara för forskning och utveckling. Detta skrev han under en tid då det knappast fanns datorer och än mindre några nätverk.

I skuggan av det kalla kriget satsade USA mycket pengar på grundforskning. Man hade flera datorer och ett visst behov av att koppla ihop dem. En matematiker vid namn Leonard Kleinrock hade utvecklat flera modeller för hur en ny slags meddelandeöverföring skulle kunna byggas. Tekniken fick senare namnet paketförmedling, men Kleinrock visade på möjligheten att dela upp ett meddelande i mindre block och hur matematisk köteori gjorde det möjligt att förutsäga nätverkets beteende. Paul Roberts var väl insatt i Klenrocks teorier och bestämde under sitt arbete på ARPA att det nya nät som skulle koppla ihop ARPA:s datorer skulle bygga på paketförmedling. En teknisk specifikation skrevs och den efterföljande

upphandlingen vanns av ett företag som hette Bolt Beranek and Newman. Tillsammans lyckades man bygga världens första paketförmedlade nätverk. Det visade sig att paketförmedling inte bara var en teori som saknade praktiskt betydelse utan en praktiskt användbar teknik. Arpanet var klart och fungerade 1969. Men vad skulle vi ha det till? Det dröjde ett par år innan e-post och filöverföring kom på plats ordentligt.

Ett helt annat spår som ledde till Arpanets födelse var en polskfödd amerikan vid namn Paul Baran. Baran skrev under tidigt 1960-tal ett antal artiklar om kommunikationssystem där han pekade på sårbarheten i dåtidens telekommunikationer. Ett fåtal missiler skulle kunna slå ut ett par av de centraliserade knutpunkter som fanns, de övriga skulle bli överlastade och hela systemet fungera dåligt. Om nätet istället kunde digitaliseras skulle man kunna göra flera hopp i nätet (analog förstärkning fungerar inte lika bra, vi förstärker även bruset). Barans idéer var inspirerade av hur hjärnan fungerar, antalet kopplingar är minst lika viktigt som antalet datorer (eller hjärnceller). Genom att använda många kopplingar med lägre kapacitet kan vi skapa ett nätverk som fungerar bättre som helhet. För att detta ska fungera behövs någon slags adaptiv vägvalsalgoritm (det vi idag kallar routing). Barans idéer möttes med stor oförståelse från den amerikanska telekombranschen, men till slut lyckades han övertyga Pentagon och flygvapnet att de kunde vara värda att prova.

Arpanet byggdes ut i flera omgångar under 1970- och 80-talet med kraftfullare teleförbindelser och fler och fler anslutna företag, organisationer och universitet. Det fanns även andra större nät byggda på andra protokoll som började ansluta sig och TCP/IP blev den gemensamma faktorn. Militären tyckte nätet blev för stort och satte sig på en egen gren kallad MILNET. För forskningsvärlden byggdes NSFNET. Då TCP/IP var den gemensamma faktorn för dessa nät började man någon gång på 1980-talet att referera till nätet som ”Internet”.

TCP/IP:s första år på Internet

En fundamental mekanism bakom framgången för TCP/IP är att protokollen utvecklas. TCP har förbättrats i flera omgångar avseende omsändningar och flödeshantering. Routingen av IP-paket har utvecklats från ganska enkla mekanismer i det första protokollet för programmet ”routed” (som sedan blev RIP) till dagens OSPF. De flesta förändringar har man kunnat lägga till och således blir tekniken bakåtkompatibel. Två TCP-implementationer kan handskaka och via optioner se om de kan använda mer avancerad flödeshantering.

När TCP/IP började användas i större nät visade det sig att näten var svåra att få igång efter driftstopp. När infrastrukturen kommit igång satte flera noder igång med att lasta nätverket, och nätverksutrustning behöver lite tid för att lära sig hur nätet ser ut och vilka MAC-adresser som används. En lösning blev slow start föreslagen av W.R. Stevens (se kapitlet TCP- och UDP-nivån). Tillsammans med andra mekanismer för effektiv flödeshantering har TCP blivit 2 till 10 gånger mer effektivt än vad det var från början.

Hur stora headers ska vi tillåta? På ett snabbt lokalt nät spelar det mindre roll men för en långsam uppringd förbindelse är det viktigt. van Jacobson föreslog en metod för komprimering av headers som fortfarande är populär (RFC 1144), men idag finns fler andra varianter.

Men vi har inte kunnat utveckla oss från alla problem. IP utvecklades innan persondatorer, lokala nätverk, Unix arbetsstationer och mobiltelefoner hade slagit igenom. Det var framsynt att välja en adressrymd om fyra miljarder adresser (32 bitar). Men det var synd att man inte valde en större adressrymd. Dessutom delade man initialt in adresserna i vad som kallades klass A-, B- och C-nät.

	Första siffran börjar på	Antal möjliga nätverk	Antal noder per nätverk	Subnätmask (standard)
A	0–127	128	16 777 216	255.0.0.0
B	128–191	16 384	65 536	255.255.0.0
C	192–223	2 097 152	256	255.255.255.0

Adressklasser.

Förr delades IP-adresser in i vad som kallades A-, B- och C-nät. Idag används istället klasslösa nät.

Det finns två ytterligare klasser. Klass D-nät börjar på 224–239 och används för multicast. Klass E-nät börjar på 240–255 och har reserverats för experimentellt bruk.

Det här innebar att 50 procent av adressutrymmet reserverats för mycket stora nät. Därför har man idag övergett denna historiska uppdelning även om man ofta refererar till exempelvis ett klass C-nät. (Ofta används det felaktigt, ett nät 82.3.4.0 med mask 255.255.255.0 betecknas C-nät. Det är egentligen ett subnät ur ett A-nät med en mask som ett klass C-nät.) I denna guide används i stället beteckningar ur klasslös indelning: /24 för masken 255.255.255.0.

På senare år har man börjat dela ut subnät ur det som historiskt sett var reserverat för klass A-nät. Detta har gjort att det historiskt har rått delade meningar om eventuell brist på adresser på Internet och ifall vi behöver IPv6 eller inte, men på sikt räcker inte heller denna metod. Men som Internet är uppbyggt idag har till exempel vissa amerikanska universitet vart och ett fler publika IP-adresser än hela Kina.

I en källare på KTH

Det svenska universitetsnätet införde TCP/IP 1988, i och med det fick svenska och nordiska universitet och högskolor tillgång till Internet. Televerket var inte berett att delta från början, e-post mellan företag fanns redan via Memo och man tjänade pengar på sina X.25-nät. Dessutom skulle ju OSI komma. Så en viktig Internetknutpunkt byggdes upp i källaren på KTH.

Många svenskar var inblandade i det tidiga byggandet av Internet-Sverige. Björn Eriksen var den som fick igång e-post mot EUnet i Amsterdam 1983. Björn Eriksen var även den som i stort sett själv kom att hantera den svenska toppdomänen ”.se” de närmsta femton åren. Under denna tid växte antalet ansökningar kraftigt. Alla dessa ansökningar hanterades av en person. Björn Eriksen var även inblandad i tilldelning och utdelning av IP-adresser. En annan tidig Internetpionjär i Sverige är Peter Löthberg. Historierna om hans hemmabyggda atomur för att erhålla korrekt tidsstämpling

och hur han lyckades sammankoppla i stort sett allt elektroniskt med VAX:ar och Cisco-routrar under 1990-talet är många.

Intressant ur svenskt perspektiv är att vi var tidigt ute på Internet-området. Detta var inte politiskt drivet, politikerna vaknade först i mitten av 1990-talet när det mesta var på plats (PTS, Post- och Telestyrelsen, drev i och för sig under 1990-talet en lyckad förändring så att knutpunkterna blev flera och flyttades till driftsäkra bergrum). Inom EU hade man främst en inriktning mot OSI-protokollen, något som hämmade utveckling av Internet i flera år för ett flertal europeiska länder. Universiteten i Sverige var tidigt ute med TCP/IP, flera av de verksamma hade varit i USA och sett att Internet fanns på plats och fungerade. Ett spår av Sveriges tidiga uppkoppling mot Internet är att Netnod AB driver en av de tretton rotservrarna för DNS.

Under tidigt 1990-tal började Internet öppnas för kommersiella intressenter. Detta nät erbjöds sedan Tele2 och Televerket (Telia). Televerket avböjde men Tele2 såg potentialen till vad som sedan fick namnet Swipnet. Swipnet var den första kommersiella Internettjänsten utanför USA. Till en början var Swipnet inriktade på företag. Webben hade börjat se dagens ljus men Internettrafiken i Sverige i början av 1990-talet var främst NNTP (news) och e-post. I IT-branschen hade det börjat bli trendigt att skriva ut e-postadresser i annonser (även om det ofta inte var någon som svarade när man skickade frågor till info@firma.se eller support@firma.se). 1994 plockade Swipnet ihop ett erbjudande om modemaccess med en bok. Det var tänkt att bli årets julklapp. Samma år hade Carl Bildt skickat e-post till Bill Clinton. En handfull företag hade förstått potentialen i vad webbsidor skulle kunna användas till. Hjulet hade börjat rulla och det skulle gå snabbt.

Request For Comments

Arpanet, och kanske även paketförmedling, hade i början karaktären av forskningsprojekt. Men i takt med att nätet byggdes ut och fungerade började det driftsättas. För att formalisera de överens-

kommelser som gjorts började man dokumentera. Från början handlade det om anteckningar från möten och diverse debattinlägg. Gradvis gick dessa dokument sedan över till mer formella standarder för protokoll och program. IP, ICMP och TCP har RFC-nummer 791–793 så processen fanns på plats före TCP/IP.

I april 1969 skrev Steve Crocker det första dokumentet i denna serie, följaktligen nummer ett. Dokumentet beskrev mötesanteckningar angående ett protokoll mellan två noder ”Host Software” på Arpanet. Enligt historien knappade han in dokumentet i ett badrum för att inte störa sina kamrater som han delade lägenhet med. Han valde att sätta namnet till Request For Comment, RFC. Ett namn som inbjuder till kommentarer och en öppenhet kring hur design och protokoll stegvis kan förbättras.

Ganska snart uppstod ett behov av samordning inom RFC:er. Man var tvungen att kontrollera så att en RFC inte stred mot en tidigare. Eller rättare sagt om det sker så måste man tala om vilken av varianterna som gäller för kommunikation på Internet just nu. Det står var och en fritt att komma med förslag på hur förbättringar ska införas. Den första person som tog på sig arbetet med att utföra denna uppgift var John Postel. John Postel innehöll denna position från år 1969 till sin bortgång 1998. Han utförde den med stor talang både vad det gäller kunskap och integritet. I takt med att Internet växte och kommersialiserades blev det ännu viktigare vilka RFC:er som fick status ”Standards track” respektive ”Best Current Practice” eller non-Standard.

Arbetet med RFC:er skiljer sig lite från traditionellt standardiseringsarbete. Man var tidigt inne på att få fram fungerande metoder och protokoll. Alltså antog man inte en standard om det inte fanns implementerad på någon plattform och senare tillkom kravet på att man dessutom ska ha skrivit en implementation till på en annan plattform med RFC:n som utgångspunkt. Dessa två ska fungera ihop (interoperabilitet) för att säkerställa kvaliteten på en RFC. Om dessa villkor uppfylls så kan RFC:n få status ”full standard”.

En översikt över vilka RFC:er som har status ”full standard” kan man få i RFC 5000. Idag år 2009 rör det sig om ett hundratal.

Det finns dock ett flertal RFC:er som inte alls har ambitionen att

bli standard. De kanske bara innehåller goda råd eller sammanställer gamla RFC:er på ett nytt sätt. Därför finns flera typer av RFC:er:

Standards track (med olika status)
- draft (utkast)
- proposed
- full standard
- obsolete

Best Current Practice BCP
(utvecklas i stort sett som standard track)

Non-standards track
- informational
- experimental
- historic

Under flera år innehöll var hundrade RFC, de som slutar på 00, referensinformation om vilken status och typ olika RFC:er hade. RFC 3700 innehöll sådan information, den ersattes år 2008 av RFC 5000. Idag är det säkrast att kontrollera statusen på www.rfc-editor.org.

Den officiella platsen för RFC:er finns på www.rfc-editor.org.

Framtidens IP-nät

Hur ska Internetprotokollen utvecklas i framtiden? Man kan titta på IETF:s (Internet Engineering Task Force) utvecklingsområden och arbetsgrupper för att få en uppfattning om vad som är huvudsakliga utvecklingsområden.

Område	Antal arbetsgrupper
Applications Area	12
General Area	1
Internet Area	29
Operations and Management Area	18
Real-time Applications and Infrastructure Area	14
Routing Area	16
Security Area	17
Transport Area	15

IETF:s (Internet Engineering Task Force) utvecklingsområden och arbetsgrupper ger en bild av hur de anser att Internetprotokollen ska utvecklas i framtiden.

Det exakta antalet varierar med tiden. Tanken är att en arbetsgrupp ska ha begränsad livslängd för att lösa en specifik uppgift. Antalet arbetsgrupper har varit mellan 100 och 150 i flera år.

Vill man titta på lite längre sikt kan man undersöka vad som är på gång inom IRTF, Internet Research Task Force (www.irtf.org). Man arbetar inom ett tiotal områden med siktet inställt på lite längre avstånd i både tid och rum…

Användningen av Internet har ändrats flera gånger sedan TCP/IP blev standard. E-post var drivande i början men det var i början av 1990-talet news-grupper som stod för huvuddelen av all trafik. När sedan webben slog så förändrades trafikmönstret. Från 1990 till 2000 räknar man med att antalet nätverk anslutna till Internet gick från storleksordningen 1 000 till 100 000, antalet användare ökade även de med en faktor 100. När sedan fildelning och peer-to-peer-nät återigen förändrade hur Internet används har vi alltså ett nätverk som är mångdubbelt större än när protokoll och principer fastställdes.

De principer som används på Internet när design och protokoll tas fram ska vara skalbara. Samma teknik ska fungera för 100 såväl som 10 000 000 användare.

När det här skrivs har uttrycket ”webben 2.0” använts i ett par år. Webben skulle ha kommit till en ny nivå där främst bloggar, podcasting, chat, wiki och virtuella världar är de drivande krafterna. Detta är möjligt tack vare den generella arkitekturen i TCP/IP och

på Internet. Alla mekanismer för detta ligger bakåt i tiden: möjligheten för vem som helst att sätta upp en server och/eller vara publicist. Meddelandetjänster typ IRC har funnits länge. Det är främst tidsperspektivet (med wiki kan vem som helst ändra innehållet på webbsidor snabbt) och den stora mängden publicister som är nytt. Och detta hade inte varit möjligt om inte tekniken varit lättanvänd.

Internet, TCP/IP och RFC:er är inte bara arkitektur och standard. Det är även ett arbetssätt med öppenhet och med ett arbetssätt som tillåter flera parallella utvecklingsspår. TCP/IP står inför en mängd utmaningar om ett och samma protokoll ska bli både bättre på att fungera i realtid, bättre på säkerhet, kunna hantera mångdubbelt större nät och samtidigt bli bättre på trådlös överföring. Men arbetssättet inom Internetgemenskapen gör att vi kan prova flera parallella utvecklingsspår. Och den här guiden handlar till stor del om hur vi ska utveckla TCP/IP mot bättre funktioner för trådlöshet, realtid och säkerhet. Det är en komplex fråga.

Om jag skulle göra några gissningar utöver detta skulle jag tro att vi måste adressera hur lagar ska appliceras på Internet. Internet befinner sig i en slags medeltid. Ute i de stora skogarna finns det gott om stråtrövare. Vi skyddar oss med att bygga stadsmurar och fästningar (brandväggar). Men är detta rätt lösning? Det är inte säkert att vi kommer att få vara anonyma när vi använder Internet om ett par decennier.

Man kan tycka vad man vill om de direktiv som EU och svenska staten vill använda för att tvinga operatörer att logga varje uppkoppling, men en regel som man har nytta av är att Internetoperatörer tvingas filtrera trafik så att avsändaradresserna inte manipuleras. Som Internet fungerar idag kan en avsändare på vissa delar av Internet enkelt manipulera sin avsändaradress på IP-nivå. Detta är ett skäl till att det är enkelt att uppträda anonymt på Internet. Operatörerna skulle enkelt kunna kontrollera att avsändaradress alltid stämmer med de IP-adresser som anslutningen tilldelats och jag tror att vi kommer hamna där. (Intressant nog visade det sig under 2007 att svenska myndigheter och ambassader själva använde anonymiseringstjänster för att minska spårbarheten. Detta blev upp-

märksammat då en anonymiseringstjänst delvis blev avlyssnad och en mängd inloggningsuppgifter blev utlagda på Internet.)

Vi vet heller inte om alla operatörer kommer fortsätta att byta trafik med varandra så oreglerat som det sker idag (det tekniska begreppet mellan operatörer är peering). Idag utbyter i stort sett alla operatörer trafik med varandra i ett världsomspännande nät. Det här är fundamentalt inom Internet. Men det är från ett fåtal nät i världen som huvuddelen av all spam kommer. Begreppet peering kommer från att vi ska mötas som jämlikar men det fungerar inte då ett fåtal operatörer inte får ordning på hur deras nätverk användas. Alltså finns risk att vi i framtiden sätter upp stora ”tullhus” eller karantäner mot dessa nätverk. Eller också får de inte vara med alls och Internet fragmenteras.

Ur teknisk synvinkel tror jag att trafiken kommer att omfatta mer och mer realtidsöverföring av audio och video, med andra ord IP-telefoni och IPTV (ingen ovanlig gissning…). IP är tänkt att kunna hantera multicast men det är först på senare år som vi kunnat hantera det i större skala. Och i framtidens Internet måste vi använda multicast om vi ska använda nätets resurser effektivt.

Nästa version av IP, IPv6, har funnits i flera år men införandet har gått långsamt. På senare år har det blivit allt mer uppenbart att adressbristen inom IPv4 kommer att begränsa tillväxten för Internet. För närvarande delas det ut ungefär 80 miljoner IPv4-adresser per år i hela världen. De IP-adresser som inte delats ut ännu räcker bara i ett par år till. Under 2008 skickade IANA, och ett flertal andra organisationer som arbetar med Internets infrastruktur, ut ett meddelande där man uppmanar organisationer att planera för och börja använda IPv6. När IPv4-adresserna blir en bristvara får man göra något drastiskt. Flera möjligheter har diskuterats som att börja ta betalt för adresser, att börja ta tillbaka adresser som inte annonseras på Internet eller att lappa och laga i IPv4. Man har även diskuterat att börja använda adressöversättning i ännu större grad eller att vara ännu mer restriktiv i utdelningen. Inget av dessa alternativ är problemfritt. Det kanske enklaste är att börja ta tillbaka adresser. När organisationen ISOC under 2009 frågade sina medlemmar visade det sig dock att cirka 75 procent tyckte att någon annan

skulle lämna tillbaka adresserna, bara 25 procent kunde lämna tillbaka delar av det egna adressutrymmet. En övergång till IPv6 är inte problemfri men vi har egentligen inga problemfria alternativ.

Referenser

Where Wizards Stay Up Late Katie Hafner, Matthew Lyon 1996

De byggde Internet i Sverige Inga Hamngren, Jan Odhnoff 2003

RFC 1958 Architectural Principles of the Internet

RFC 3700 med flera Internet Official Protocol Standards

IP-nivån

- Här beskrivs hur IP fungerar med statiska och dynamiska adresser (DHCP). Kapitlet behandlar grunderna för routing och hur IP-headern är uppbyggd. Subnätmaskens funktion, utseende och hur den används för att räkna ut nätnummer behandlas.
- Protokollet ICMP behandlas översiktligt.
- Kapitlet är tänkt att kunna läsas fristående.

För att din dator ska fungera på IP-nivån så behövs tre saker konfigureras:

- en IP-adress
- en standard gateway (default gateway)
- en subnätmask.

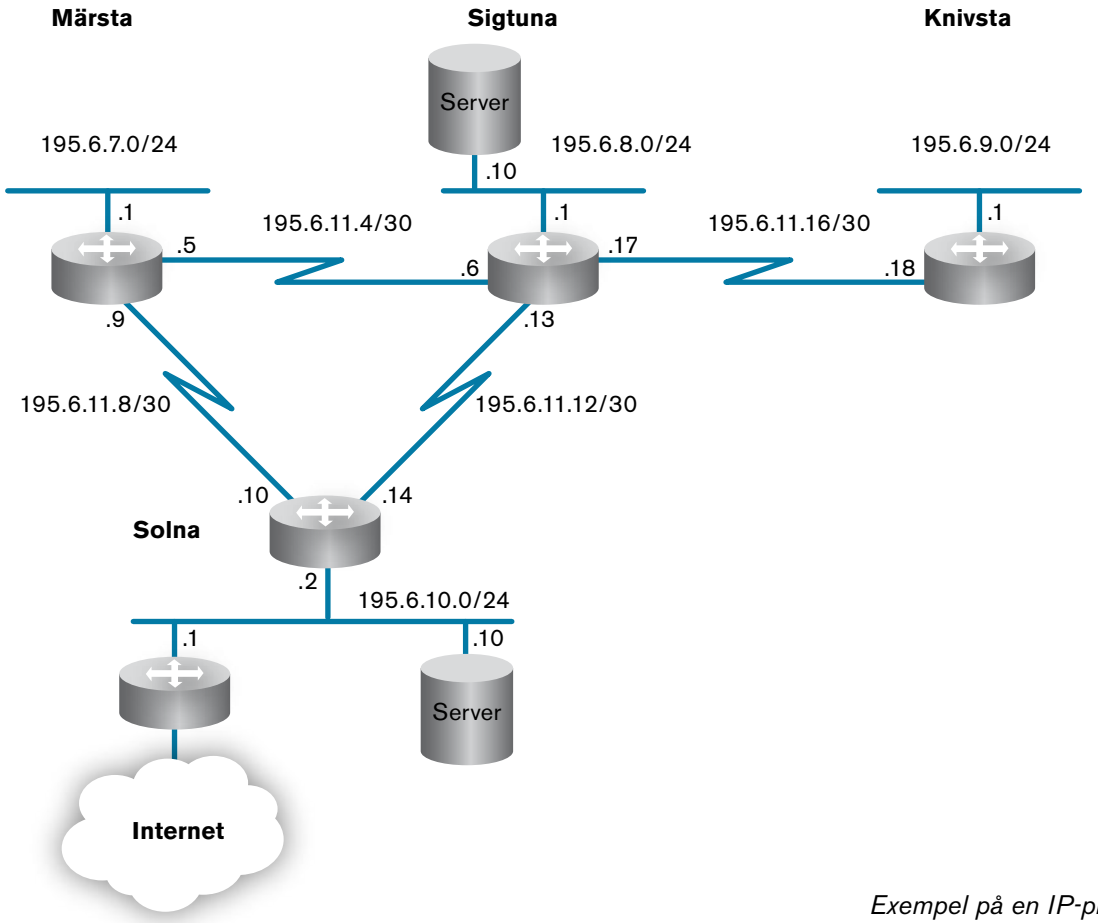
IP-adressen är datorns unika identitet på Internet, den som gör att andra hittar till din dator. Standard gateway talar om för IP-nivån vart den ska skicka paket den inte hittar till, tekniskt sett ”adresser som inte har en annan känd route”. Subnätmasken används för att ange hur stort nätet är. Men mer om allt detta nedan.

I Windows XP visas dessa tre inställningar ihop. För att du ska komma igång med del flesta applikationer krävs även IP-adressen till en namnserver (DNS står för Domain Name System), understa delen av inställningarna. IP-nivån fungerar dock utmärkt även utan namnserver.

Eftersom IP-adressen ska vara unik kan inte var och en hitta på sin egen IP-adress utan de måste delas ut centralt. Idag delas en del adresser,

IP-inställningar i Windows XP.

genom att så kallade privata adresser används tillsammans med adressöversättning. En privat adress fungerar dock inte på Internet utan måste översättas till en publik adress. Adressöversättning hanteras i ett eget kapitel. Du kan erhålla IP-konfigurationen från en central server via protokollet DHCP (Dynamic Host Configuration Protocol), eller manuellt från IT-avdelningen eller din operatör. På Internet och i stora organisationer finns det en funktion för detta som kallas LIR (Local Internet Registry). Använder organisationen publika



Exempel på en IP-plan.

adresser så har man förbundit sig till att hålla reda på hur de används. Det finns inga formkrav på dokumentationen men vanligast är en IP-plan.

I IP-planer skrivs oftast IP-adresser i kortform. Den router som i Solna sitter mot Internet har IP-adressen 195.6.10.1. Men eftersom subnätmasken och de tre första byten är gemensamma för hela nätet skrivs de på ett ställe.

Ur IP-planen ovan kan man läsa ut att en fungerande IP-adress i Knivsta skulle vara till exempel 195.6.9.25, subnätmask 255.255.255.0 och standard gateway 195.6.9.1. Standard gateway är IP-adressen till en router som leder ut till Internet. I Knivsta kommer vi bara ut på Internet via en router, som i sin tur går via Sigtuna och sedan Solna. Av historiska skäl kallas router i vissa sammanhang för gateway, detta är alltså den router man ska använda standardmässigt om man inte har en bättre idé om hur man ska nå den adress man försöker skicka data till. Även begreppet default route är vanligt.

Begreppet gateway har historiskt fått flera betydelser inom datakommunikation. I vissa sammanhang kan gateway och router användas som synonymer.

Att subnätmasken är 255.255.255.0 kan man utläsa ur beteckningen /24 (uttalas slash 24). Ofta skriver man inte enstaka noders adresser i en IP-plan utan man noterar nätnummer som 195.6.9.0. Vi ska titta på hur IP-adresser, subnätmasker och nätnummer hänger ihop lite längre fram.

Var kommer IP-adresserna ifrån?

IP-adresserna kan anges manuellt under /Inställningar/Nätverksanslutningar/LAN/Internet Protocol (TCP/IP) eller liknande. Du kan kontrollera inställningarna med kommandot ”ipconfig/all” i Windows. På Mac eller Linux skriver du ifconfig. Resultatet ska bli liknande:

I exemplet på nästa sida har IP-adresserna hämtas från en server, de rader som har med DHCP att göra har markerats med fetstil. För att IP ska fungera behöver IP-parametrarna anges korrekt och det finns mycket att vinna på att centralisera denna funktion i en DHCP-server. Dessutom flyttar många noder runt mellan olika nät-

```
Ethernet-kort Local Area Connection:

        Anslutningsspecifika DNS-suffix . . :
        Beskrivning . . . . . . . . . . . . : Intel(R) LAN 2435
        Fysisk adress . . . . . . . . . . . : 00-13-CE-26-F9-18
        DHCP aktiverat  . . . . . . . . . . : Ja
        Autokonfiguration aktiverat  . . . . : Ja
        IP-adress . . . . . . . . . . . . . : 192.168.1.2
        Nätmask . . . . . . . . . . . . . . : 255.255.255.0
        Standard-gateway  . . . . . . . . . : 192.168.1.1
        DHCP-server . . . . . . . . . . . . : 192.168.1.1
        DNS-servrar . . . . . . . . . . . . : 192.168.1.1
        Lånet erhölls . . . . . . . . . . . : den 2 maj 2007 16:41:41
        Lånet upphör  . . . . . . . . . . . : den 3 maj 2007 04:41:41
```

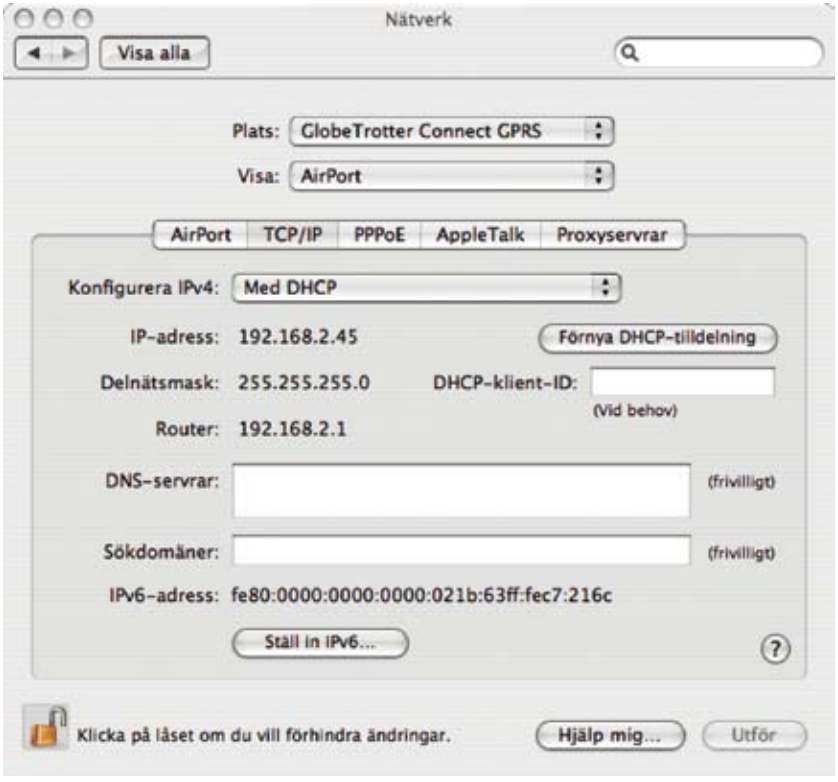
IP-adresser hämtade från en server.

verk. Med statiska IP-adresser får man gå in och ändra dem hela tiden, och för inte så många år sedan ledde ändrade IP-adresser alltid till en omstart. Med DHCP blir detta mycket lättare, noderna frågar efter IP-inställningar för varje nät. Det har också blivit vanligt att moderna operativsystem kontrollerar DHCP-lånet så fort länknivån ändras, det vill säga så fort man drar ut en nätverkskabel eller byter anslutning via WLAN.

Dynamisk utdelning av IP-adresser

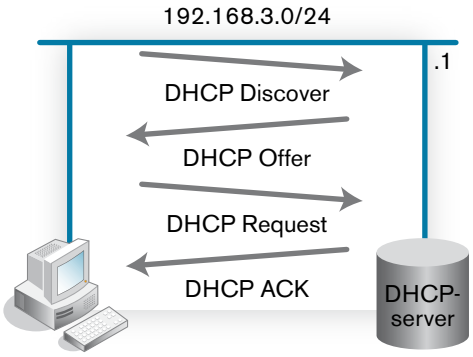
Den vanligaste metoden för dynamisk utdelning av adresser är DHCP (RFC 1541 och 2131 med flera). DHCP är en funktionell förbättring av det äldre protokollet Bootp. DHCP använder också samma portnummer som Bootp, se protokollanalysen på nästa sida. Bootp var ett enkelt protokoll där en server huvudsakligen höll reda

på vilken IP-adress som skulle delas ut till vilken MAC-adress. Sådana statiska mappningar kan även DHCP erbjuda men DHCP erbjuder flera förbättringar.



Inställningar för DHCP i MAC OS X.

Med DHCP kan flera noder dela på ett fåtal adresser. På servern lägger vi bara in hur stor adressrymd som ska delas ut. Servern håller sedan reda på vilken nod som fått dem. Då lånet av IP-adress är tidsbegränsat kan IP-adressen sedan erbjudas en ny nod. Vi kan använda flera DHCP-servers för att öka driftsäkerheten. Klienterna börjar med att fråga efter DHCP-servers (DHCP Discover) och får då ett erbjudande (DHCP Offer). Kommer det flera erbjudanden så



Utväxling av IP-information via DHCP.

gör det inget, klienten tar det första och bekräftar det till servern (DHCP request) varpå servern bekräftar (DHCP ack).

Man kan starta denna handskakning genom att ge kommandot ”ipconfig /renew” i Windows. En protokollanalys inspelad med Wireshark eller liknande program ger följande resultat (viss information utelämnad vid markeringar ”…”). Det första paketet skickas som en broadcast på både IP och Ethernet-nivå.

Paket 1
Ethernet II, Src: Fujitsu_c3:2f:98 Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol, Src: 0.0.0.0, Dst: 255.255.255.255
User Datagram Protocol, Src Port: bootpc (68), Dst Port: bootps (67)
Bootstrap Protocol
…
Bootp flags: 0x0000 (Unicast)
Client IP address: 0.0.0.0
Your (client) IP address: 0.0.0.0
Next server IP address: 0.0.0.0
Relay agent IP address: 0.0.0.0
Client MAC address: Fujitsu_c3:2f:98 (00:0b:5d:c3:2f:98)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) DHCP Message Type = DHCP Discover
Option: (t=116,l=1) DHCP Auto-Configuration
Option: (t=61,l=7) Client identifier
Option: (t=50,l=4) Requested IP Address = 10.36.21.159
Option: (t=12,l=14) Host Name = “lifebook-p7010”
Option: (t=60,l=8) Vendor class identifier = “MSFT 5.0”
Option: (t=55,l=11) Parameter Request List
End Option

Del ur en protokollanalys. Här visas Paket 1 – DHCP Discover.

Lägg märke till att klienten egentligen vill ha en annan adress, 10.36.21.159, än vad den erbjuds senare. Klienten skickar över sin MAC-adress till servern, som kan kontrollera eventuellt befintligt lån.

```
Paket 2
Ethernet II, Src: 3comEuro_9f:4a:fa , Dst: Fujitsu_c3:2f:98
Internet Protocol, Src: 192.168.3.1, Dst: 192.168.3.245
User Datagram Protocol, Src Port: bootps (67), Dst Port: bootpc (68)
Bootstrap Protocol
   Message type: Boot Reply (2)
   …
   Client IP address: 0.0.0.0
   Your (client) IP address: 192.168.3.245
   Next server IP address: 0.0.0.0
   Relay agent IP address: 0.0.0.0
   Client MAC address: Fujitsu_c3:2f:98 (00:0b:5d:c3:2f:98)
   Server host name: \377
   Boot file name: \377
   Magic cookie: (OK)
   Option: (t=53,l=1) DHCP Message Type = DHCP Offer
   Option: (t=54,l=4) Server Identifier = 192.168.3.1
   Option: (t=1,l=4) Subnet Mask = 255.255.255.0
   Option: (t=51,l=4) IP Address Lease Time = 12 hours
   Option: (t=52,l=1) Option Overload = Boot file and server host names
       hold options
   Option: (t=3,l=4) Router = 192.168.3.1
   Option: (t=6,l=4) Domain Name Server = 192.168.3.1
   End Option
   Padding
```

Från servern har en ny IP-adress erbjudits. Klienten får behålla den i 12 timmar.

I erbjudandet från servern kommer nu en ny IP-adress tillsammans med övrig konfiguration (nätmask, standard gateway och DNS). Det framgår även att klienten får låna IP-adressen i tolv timmar.

Klienten skickar en begäran om denna IP-adress till severn. Detta är fortfarande en broadcast och lägg märke till att klienten fortfarande skickar från IP-adress 0.0.0.0. Paket fyra är sedan servern kvittens där all IP-konfiguration överförs.

```
Paket 3
Ethernet II, Src: Fujitsu_c3:2f:98, Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol, Src: 0.0.0.0, Dst: 255.255.255.255
User Datagram Protocol, Src Port: bootpc (68), Dst Port: bootps (67)
Bootstrap Protocol
   …
   Client IP address: 0.0.0.0
   Your (client) IP address: 0.0.0.0
   Next server IP address: 0.0.0.0
   Relay agent IP address: 0.0.0.0
   Client MAC address: Fujitsu_c3:2f:98 (00:0b:5d:c3:2f:98)
   Server host name not given
   Boot file name not given
   Magic cookie: (OK)
   Option: (t=53,l=1) DHCP Message Type = DHCP Request
   Option: (t=61,l=7) Client identifier
   Option: (t=50,l=4) Requested IP Address = 192.168.3.245
   Option: (t=54,l=4) Server Identifier = 192.168.3.1
   Option: (t=12,l=14) Host Name = “lifebook-p7010”
   Option: (t=81,l=18) Client Fully Qualified Domain Name
   Option: (t=60,l=8) Vendor class identifier = “MSFT 5.0”
   Option: (t=55,l=11) Parameter Request List
   End Option
```

Paket 3 – Klienten meddelar att den accepterar erbjudandet via DHCP Request.

```
Paket 4
Ethernet II, Src: 3comEuro_9f:4a:fa, Dst: Fujitsu_c3:2f:98
Internet Protocol, Src: 192.168.3.1, Dst: 192.168.3.245
User Datagram Protocol, Src Port: bootps (67), Dst Port: bootpc (68)
Bootstrap Protocol
   Message type: Boot Reply (2)
   …
   Client IP address: 0.0.0.0
   Your (client) IP address: 192.168.3.245
   Next server IP address: 0.0.0.0
   Relay agent IP address: 0.0.0.0
   Client MAC address: Fujitsu_c3:2f:98 (00:0b:5d:c3:2f:98)
   Server host name: \377
   Boot file name: \377
   Magic cookie: (OK)
   Option: (t=53,l=1) DHCP Message Type = DHCP ACK
   Option: (t=54,l=4) Server Identifier = 192.168.3.1
   Option: (t=1,l=4) Subnet Mask = 255.255.255.0
   Option: (t=51,l=4) IP Address Lease Time = 12 hours
   Option: (t=52,l=1) Option Overload = Boot file and server host names
       hold options
   Option: (t=3,l=4) Router = 192.168.3.1
   Option: (t=6,l=4) Domain Name Server = 192.168.3.2
   End Option
   Padding
```

Paket 4 – Serverns kvittens att IP-konfigurationen överförts.

Hur länge klienten erbjuds att låna IP-adressen är en viktig parameter. Efter halva den tiden kommer klienten försöka att förnya lånet. Detta för att hela tiden försäkra sig om att ha en fungerande IP-adress. Om detta inte lyckas kommer klienten att göra en helt ny förfrågan (DHCP Discover) efter 87 procent av tiden.

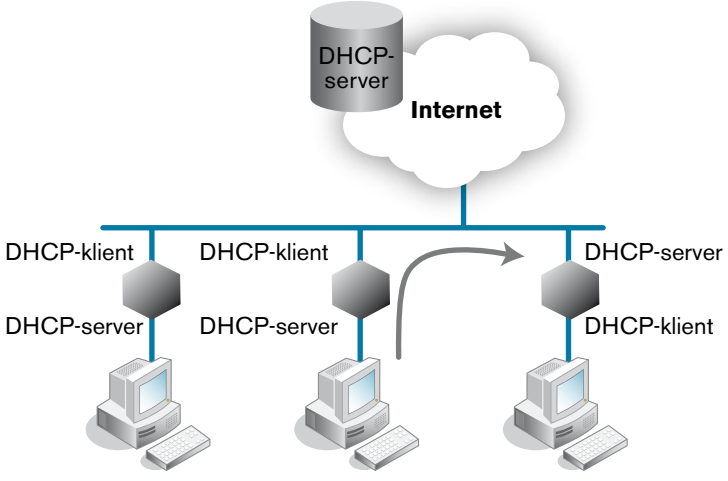
Hur långa lånetider man ska ha varierar. I ett företagsnät har man ofta lånetider på en vecka, antalet maskiner som flyttar sig är

inte så stort. På ett publikt trådlöst nätverk där användare kommer och går kan man ha lånetider på fem minuter. I ett operatörsnät låter man kunderna typiskt ha lånetider mellan 20 och 60 minuter. På mer avancerad nätverksutrustning är därför lånetiden alltid ställbar.

DHCP designades i början av 1990-talet och är idag en självklar del av modern nätverksteknik. Microsoft var snabba med använda tekniken, kanske lite för snabba men idag fungerar DHCP väldigt stabilt. Även i Unix/Linux finns fullgott stöd för DHCP både som server och klient. Själv har jag dock haft en del problem med distribution av DNS-information via DHCP till Linux-klienter.

DHCP gör IP-kommunikation lättare att komma igång med men också osäkrare.

Det är ofta svårt att kombinera säkerhet med användbarhet. Detta gäller även DHCP. DHCP är gjort för att inte krångla eller konstra. Varken klienten eller servern har i grundutförandet någon slags autentisering. Hur vet vi att vi pratar med rätt eller ens en godkänd server? En möjlig attack från en förövare är se till att svara på DHCP Discover från en nod och sedan ange sig själv som standard gateway. På så vis kan man göra kopior på all information till och från klienten. Det finns även andra möjligheter att ange fel subnätmask och på så



En användare kan felaktigt koppla in en DHCP-server till Internet.

sätt öppna nätet. Attacken bygger i princip på att förövaren sitter på samma lokala nätverk som den som attackeras. DHCP är tänkt att fungera lokalt för att minska riskerna, och man bör ha god kontroll ifall man ska låta det passera routrar.

En annan aspekt på problem med DHCP är välkänd hos operatörer. Användare kopplar in sig via små gateways som använder DHCP-klienter mot operatörer (WAN-anslutningen) och som sedan ska agera DHCP-server mot det egna nätet.

Någon användare vänder sin enhet fel och besvarar effektivt andra användares DHCP-frågor med sina egna normalt privata adresser. Det fungerar inte bara dåligt för den felvända enheten utan påverkar även andra. Utan onda avsikter får vi en effektiv Denial-Of-Service attack. Protokollet DHCP kan inte skydda mot detta utan det får operatören bygga funktioner för.

Fyra miljarder adresser

En IP-adress består av 32 bitar eller fyra byte. Det är en konvention att skriva dem på så kallad decimal punktnotation, där vi sätter in en punkt mellan varje byte. Detta gör att vi inte behöver skriva ut inledande nollor i varje byte.

195.006.007.010 kan skrivas som 195.6.7.10

Detta noteringssätt är helt förhärskande men det förekommer att IP-adresser skrivs i hexadecimalt format som C306070A.

32 bitar gör att vi har 2^{32} adresser till förfogande vilket motsvarar drygt fyra miljarder. I slutet av 1970-talet var det framsynt att ta till med ett så högt tal, på den tiden fanns inga persondatorer och lokala nätverk hade knappt sett dagens ljus. Idag kan vi beklaga att man inte tog till lite mer och att man inte begränsade utdelningen av IP-adresser initialt. Var man bara tidig var det lätt att få stora adresstilldelningar. Dessutom är IP-adresser som börjar med 224–255 reserverade för bland annat multicast, så i praktiken har vi under fyra miljarder IP-adresser som kan användas.

För att IP-adresserna ska vara unika behöver de administreras centralt. Detta görs av IANA (Internet Assigned Numbers Authority) som drivs av ICANN (Internet Corporation for Assigned Names and Numbers). Ansvaret är sedan delegerat till tre organisationer och i Europa är det RIPE som hanterar detta (Réseaux IP Européens se www.ripe.net). RIPE delar sedan ut block med IP-adresser till europeiska operatörer. På RIPE:s hemsida kan man söka upp vem som administrerar en europeisk IP-adress. Oftast sköter operatörer denna administration.

På www.ripe.net kan man kontrollera vem som registrerat en IP-adress. Databasen innehåller även kontaktinformation lite längre ner.

En publik adress är alltså spårbar. Detta behövs för att man ska kunna kontakta operatören eller organisationen ifall IP-adressen är in-

blandad i något som strider mot god ”netiquette” eller ifall det finns problem med till exempel e-post eller hantering av domäner. Netiquette, eller på svenska nätvett, är ett brett begrepp innefattande allt från goda råd hur man bör bete sig på nätet för att underlätta samvaron med de övriga användarna, till hur man ska undvika rena olagligheter.

RFC 1855 är en god start för god netiquette.

Om du tittar igenom avtalet med din Internetoperatör finns det ofta klausuler om att operatören förbehåller sig rätten att stänga av din anslutning ifall ditt användande strider mot god netiquette. Detta trots att begreppet är ytterst fritt för tolkningar. Men syftet men att använda netiquette är gott. På Internet ska det finnas plats för olika åsikter men vi behöver formulera hur vi ska dela med oss av dem och hur vi ska bemöta dem som har andra åsikter.

Väldigt många miljarders miljarder adresser med IPv6

När man började arbetet på 1990-talet med nästa version av IP, IP version 6, ville man rätta till ett par problem med IPv4. Det viktigaste var att utöka adressrymden. Adresserna i IPv6 är därför fyra gånger så långa, det vill säga 128 bitar. Det blir väldigt gott om adresser, man kan dela ut en miljard adresser i sekunden till alla människor på jorden i miljarder år. Det har funnits fler fördelar med IPv6, men ett par av dem har även flyttats över till IPv4 som exempelvis kryptering och autentisering (finns med IPsec i IP version 4). Andra har lösts med DHCP, som gör att den autokonfiguration som används i IPv6 inte är lika banbrytande. Det finns andra fördelar med IPv6 än det ökade adressutrymmet. Till exempel att alla noder ska kunna hantera IPsec, det blir inte ett tillägg som i IPv4. Det gör att vi kan börja använda kryptering och autentisering i större utsträckning.

IPv6-adresserna blir fyra gånger så långa som i IP version 4. För att korta ner adresserna i löpnade text skrivs de på hexadecimal form i grupper om 16 bitar, vilket motsvarar fyra hexadecimala siffror per grupp. Grupperna skiljs åt med kolon. För att korta ner noteringssättet ytterligare kan man använda dubbla kolon ”::”,

vilket visar att en eller flera grupper bara består av nollor. Några exempel på giltiga IPv6-adresser är:

2001:234:567:1::1 (som även skulle kunna skrivas ut som 2001:234:567:1:0:0:0:1)
2001:234:5678:999:204:76ff:fec2:c193
fe80::204:76ff:fec2:c193

Enkel routing

Varje IP-nod har en enkel routingtabell. Du kan få fram den genom kommandot netstat -rn eller ”route print” i Windows. I Unix, Linux och Mac OS X används kommandot netstat -r.

```
===========================================================================
Aktiva vägar:
   Nätverksadress          Nätmask   Gateway-adress      Gränssnitt    Mått
          0.0.0.0          0.0.0.0      192.168.3.1   192.168.3.174       25
        127.0.0.0        255.0.0.0        127.0.0.1       127.0.0.1        1
      192.168.3.0    255.255.255.0    192.168.3.174   192.168.3.174       25
    192.168.3.174  255.255.255.255        127.0.0.1       127.0.0.1       25
    192.168.3.255  255.255.255.255    192.168.3.174   192.168.3.174       25
        224.0.0.0        240.0.0.0    192.168.3.174   192.168.3.174       25
  255.255.255.255  255.255.255.255    192.168.3.174   192.168.3.174        1
Standard-gateway:      192.168.3.1
===========================================================================
```

Routingtabell. Först kontrolleras om mottagaren sitter på samma nät som vi, i så fall behöver vi inte blanda in routrar och operatörer. Sedan kontrolleras om vi vet något om nätet eller just denna adress. I sista hand skickas paketet till en router som förhoppningsvis vet mer.

För att förstå tabellen behöver vi förstå subnätmaskens roll. Och vi behöver titta på hur grundfunktionen för routing fungerar:

1. Är mottagaren direkt adresserbar?

2. Är mottagaren en känd nod?
3. Tillhör mottagaren ett känt nät?
4. Använd ”default route”

Routingprocessen består av fyra huvudsakliga steg.

Routing-processen testar i denna ordning tills den hittar ett giltigt villkor. Och om den inte hittar något villkor som är applicerbart så kastas paketet och protokollet ICMP får se till så att ett felmeddelande skickas. Ordningen är i sig rätt självklar. Först kontrollerar vi om mottagaren sitter på samma nät som vi, i så fall behöver vi inte blanda in routrar och operatörer. Sedan kontrollerar vi om vi vet något om nätet eller just denna adress. I sista hand får vi skicka paketet till en router som förhoppningsvis vet mer.

Routrarna arbetar på samma sätt. Antingen sitter de på samma nät som mottagaren och då handlar det om LAN-teknik hur paketet ska vidarebefordras. Annars undersöker routern om den vet något om just denna nod eller detta nät. Annars skickas paketet till routerns standardlösning: default route. På så sätt routas paketen högre upp i hierarkin. Till slut kommer paketet hamna på en knutpunkt där anslutna routrar känner till alla nätnummer på Internet. Paketet leds då till en väg där routrarna känner till detta specifika nät. De här Internetknutpunkterna kallas idag för IXP, Internet eXchange Points. I Sverige finns cirka tio stycken.

Netnod (www.netnod.se) sköter driften på fem stycken IXP:s i Sverige. På www.ep.net finns en lista över aktuella IXP:s.

För att kunna avgöra om en mottagare är direkt adresserbar behöver noder och routrar förstå hur stort nät de hanterar. Detta är subnätmaskens uppgift och vi kan uppfatta nätets storlek olika. Många routrar i världen arbetar med nätet 17.0.0.0/8. Detta är ett stort nät om drygt 16 miljoner noder. Men internt på Apple kan de arbeta med 17.0.0.0/24 som bara avser drygt 250 adresser. Det här är ett fundament inom IP. Adressrymder kan aggregeras till större nät. Detta gör routingtabellerna mindre och routingprocessen effektivare.

Första steget, direkt adresserbar, avgör om paketet behöver adresseras till ett lokalt nät. Med hjälp av subnätmasken /24 i detta fall så kan noden räkna ut att nätets storlek är 256 adresser.

Default route (default gateway är samma sak men en vanlig beteckning för en enstaka nod) talar som sagt om vart enstaka paket ska skickas. Default route betecknas med nätnumret 0.0.0.0 och mask 0.0.0.0.

I routingtabellen på föregående sida finns nätet 127.0.0.0. Detta nät med nodadressen 127.0.0.1 används bland annat för felsökning. 127.0.0.1 kallas för localhost och gör det bland annat enkelt att felsöka. Vill man undersöka om IP fungerar så startar man gärna med localhost. Vid felsökning kan man gärna starta med localhost. Adressen används även i accesslistor och brandväggar. Om man startar en servertjänst är det inte ovanligt att den bara svarar på localhost innan man uttryckligen konfigurerat programmet eller brandväggen att svara på andra adresser.

Vad gör subnätmasken?

Subnätmasken består av 32 bitar, som var och en kan ha värdet ett eller noll. På samma sätt som med IP-adresser skrivs varje byte decimalt med en särskiljande punkt. 255.255.255.0 motsvarar alltså 11111111 11111111 11111111 00000000. Ett annat sätt att notera subnätmasken är att notera antal ettor, i detta fall 24 stycken vilket noteras ”/24”. Denna kortform är vanlig i IP-planer. På samma sätt kan man lätt få fram att 255.255.0.0 kan noteras som ”/16”. Däremot kräver det räknande via binära tal för att se hur till exempel 255.255.255.248 kan skrivas som ”/29”. På nästa sida följer en tabell med bägge noteringssätten. Det finns andra möjliga värden på subnätmasken men detta är de absolut vanligaste.

Subnätmasken anger nätets storlek.

Subnätmasken anger nätets storlek. En nod som har subnätmasken 255.255.255.0 kan då räkna ut att den sitter på ett nät som har 254 noder direkt adresserbara. Om den istället har nätmasken 255.255.255.248 så är 6 noder direkt adresserbara.

Med hjälp av subnätmasken räknar noden ut vilket nätnummer den sitter på. Beräkningen går till så att IP-adressen multipliceras med subnätmasken bit för bit (programmeringsmässigt används en logisk AND-operation: 0 AND 0 = 0, 0 AND 1 = 0, 1 AND 0 = 0,

1 AND 1 = 1). Att använda en AND-operation är ett vanligt sätt i programmering för att ta bort värden vi inte vill behandla. Vi kallar detta för att maska bort bitar och därav kommer namnet subnätmask.

En nod med IP-adressen 195.6.7.25 och subnätmask 255.255.255.0 utför följande beräkning:

IP-adress	195.006.007.025
Nätmask	255.255.255.000
Nätnummer	195.006.007.000

En nod med IP-adressen 195.6.7.25 och subnätmask 255.255.255.240 utför följande beräkning, nu med binär notering:

IP-adress	11000011 00000110 00000111 00011001
Nätmask	11111111 11111111 11111111 11110000
Nätnummer	11000011 00000110 00000111 00010000

Om vi omvandlar denna siffra till decimal punktnotation får vi 195.6.7.16. Noden kan alltså räkna ut nätnumret och dessutom räkna ut att nätets storlek är 14 noder, Första tillåtna värde blir 195.6.7.17 och det sista 195.6.7.30. Även IP-adressen 195.6.7.31 tillhör detta nät men detta värde används för IP-broadcast, det vill säga när alla noder ska adresseras. Totalt ingår IP-adresserna 195.6.7.16–195.6.7.31 i adressrymden men det lägsta numret används som nätnummer och nätnumret behövs i routingtabeller. Det högsta numret används för broadcast. Det är därför vi inte kan adressera 16 noder utan 14.

Subnätmasken används alltså för att räkna ut nätets storlek och det nätnummer som noden sitter på. Noderna 195.6.7.25 och 195.6.7.30 kommer att räkna ut att de sitter på samma nät. Och om vi tittar på hur routing fungerar är detta första villkoret,

Subnätmask	/n	Antal noder
255.0.0.0	/8	16 777 214
255.255.255.0	/16	65 534
255.255.128.0	/17	32 766
255.255.192.0	/18	16 382
255.255.224.0	/19	8 190
255.255.240.0	/20	4 094
255.255.248.0	/21	2 046
255.255.252.0	/22	1 022
255.255.254.0	/23	510
255.255.255.0	/24	254
255.255.255.128	/25	126
255.255.255.192	/26	62
255.255.255.224	/27	30
255.255.255.240	/28	14
255.255.255.248	/29	6
255.255.255.252	/30	2

På varje nät är den högsta och lägsta adressen reserverade. Den lägsta används för att adressera nätet och den högsta används för IP-broadcast.

kontrollera om noden är direkt adresserbar, i så fall kan noderna utbyta data direkt. Om vi återvänder till IP-planen i början på detta kapitel kan alltså noderna i Sigtuna pratat direkt med sin server medan noderna i Märsta och Knivsta räknar ut att de måste gå via sin router.

En felaktig subnätmask gör att noden inte kan skicka paketen vidare på rätt sätt.

Felaktig subnätmask får till konsekvens att noderna misstolkar både nätnummer och nätstorlek och då fungerar inte routing. De kan då råka ut för att de försöker skicka paket till en dator som inte sitter på samma nät eller att de inte når sin default gateway. Alla tre IP-parametrarna IP-adress, mask och default gateway måste alltså vara rätt.

I tidig IP-teknik hade subnätmasken bara värdena /8, /16 och /24. Men detta var en väldigt grov indelning som snabbt ledde till brist på IP-adresser. Själv arbetade jag under tidigt 1990-tal på ett företag med cirka 30 noder, men vi behövde på den tiden dela upp nätet. Därför erhöll vi två stycken publika nät med nätmask /24 vilket gav oss adresseringsmöjlighet upp till drygt 500 noder. Andra som var tidigt ute och lite större än oss fick ett helt /16-nät. När variabel subnätmask infördes runt mitten av 1990-talet så blev både användning och utdelning av publika adresser mycket effektivare. Det enda pris vi fått betala är att det blivit lite krångligare att räkna ut nätstorlek och nätnummer.

IP-headern

Om vi använder en nätverksanalysator för att spela in en IP-header så skulle vi normalt se 20 byte, till exempel sekvensen:

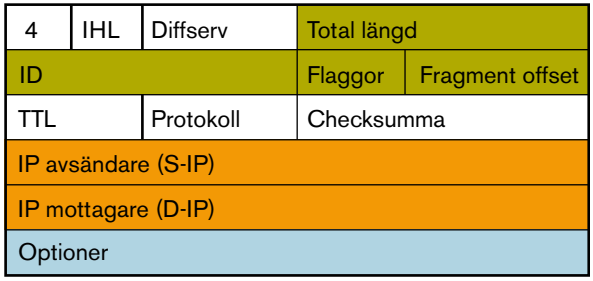
```
45000056b1ae40003506586b58831e68c0a8 03f5
```

Vi lägger in mellanslag för att öka tydligheten:

```
4500 0056  b1ae 4000  3506 586b  5883 1e68  c0a8 03f5
```

För att kunna tolka detta brukar man granska fyra byte i taget, och läsa dem som en rad. (Kom ihåg att en byte motsvarar åtta bitar eller 2 hexadecimala siffror.) Vi får då fem rader enligt figuten:

- Första siffran anger att detta är IP version 4.
- Nästa siffra (dvs 4 bitar) IHL står för IP Header Length och mäts i 32-bitars ord. Värdet 5 är vanligast och innebär att headern är 20 byte lång.
- Nästa byte är ofta just ”00”. Detta fält har historiskt tolkats som Type of Service, förkortat ToS. I modern IP tolkas detta fält enligt en standard som kallas Diffserv, en kort form av differentiated services. Fältet används av noder för att kunna skilja olika trafiktyper åt.



Figuren visar schematiskt hur man läser/analyserar en IP-header.

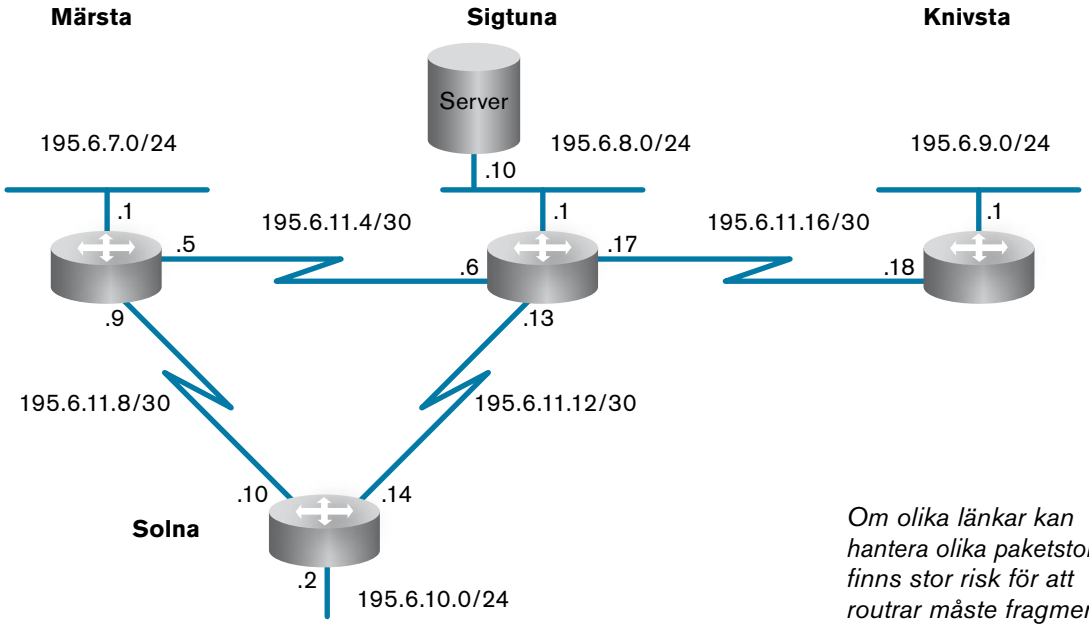
- Total längd består av 2 byte och anger paketets storlek inklusive header. Mäts i byte, 0x0056 motsvarar decimalt 86 byte. (Beteckningen 0x innebär att efterföljande format är hexadecimalt.)
- Varje paket förses med en unik identitet bestående av två byte. I detta fall 0xB1AE. IP-identiteten används bland annat vid fragmentering.
- Tre bitar är reserverade för flaggor. Den första används för närvarande inte. Den andra biten avser ”Don’t Fragment” den tredje avser ”More Fragments”. Dessa tre flaggor tillsammans med offsetvärdet bildar två byte. Värdet 4 (binärt 0100) innebär att flaggan Don’t Fragment är satt till ett. En router får alltså inte fragmentera detta paket.
- Fragment offset består av 13 bitar. Värdet är ofta noll som i exemplet ovan. Funktionen förklaras i nästa avsnitt.
- TTL, Time To Live motsvarar antal routerhopp paketet ska kunna passera. I detta fall 0x35=53 hopp innan paketet kastas. Totalt kan IP inte hantera ett Internet som skulle ha mer än 255 routerhopp, fältet är åtta bitar stort. På Internet idag ligger vi inte i närheten, ett typiskt värde idag är snarare 10–30 hopp innan paketet når sin mottagare. Av historiska skäl lever beteckningen med time kvar, i modern IP borde parametern kallas ”hops to live”.

- Protokollfältet med i detta fall innehåll 06 tolkas som att IP bär protokollet TCP.
- Checksumma 586b används för att kunna detektera överförings-fel eller fel i program. Checksumman i IP är av en enkel typ och detekterar bara fel i IP-headern, inte i nyttolasten. IP-nivån förutsätter inte att nivå två kontrollerar innehållet, därför kan överföringsfel uppstå och IP-nivån skulle då fatta routingbeslut på felaktiga paket. Behovet av denna funktion har minskat kraftigt sedan IP togs fram, moderna nivå två protokoll hanterar överföringsfel effektivare än IP.
- Funktionen för fältet IP-avsändare framgår av namnet. I den här guiden används ofta kortformen S-IP, IP-Source, men även beteckningen ”Src” förekommer. Det hexadecimala värdet 5883 1e68 motsvarar 88.131.30.104.
- IP mottagare, i den här guiden ofta betecknat som D-IP, D för destination. c0a8 03f5 motsvarar 192.168.3.245.

Fragmentering

I bästa fall kan vi helt undvika fragmentering. TCP ska dela upp dataflödet i lagom stora paket. Men en avsändare kan inte vara säker på hur paket hanteras av alla länkar som kan vara inblandade i en överföring mellan två noder. Dessutom kan ju förhållandena ändras. Därför kan en router få ett inkommande paket med en storlek som nästkommande länk inte kan hantera. Alltså behöver paketet delas upp – fragmenteras.
Varje länk har ett maximal värde för hur stora paket den hanterar, värdet kallas för MTU Maximum Transmission Unit. För ett vanligt Ethernet är värdet 1 500 byte, men värden ned till 576 byte är tillåtna i IPv4.

Om vi tänker oss att servern i Sigtuna ska skicka paket till en nod på Märsta-nätet så kommer den försöka med paketlängden 1 500 byte. Den seriella länken mellan Sigtuna och Märsta har en

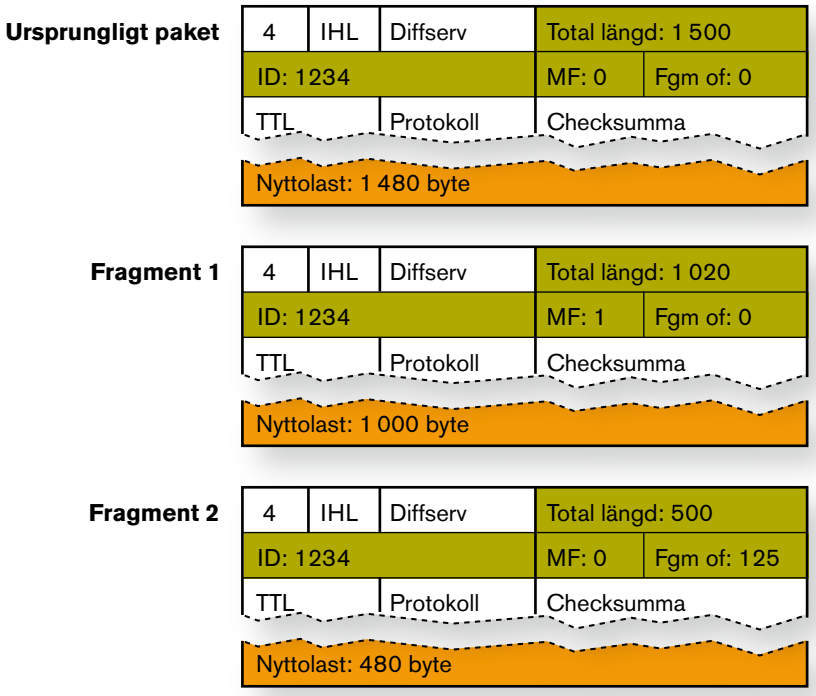


Om olika länkar kan hantera olika paketstorlekar finns stor risk för att routrar måste fragmentera.

MTU om 1 024. Routern har därför inget annat val än att dela upp paketet.

När routern fragmenterar får den göra det i sektioner om åtta byte, och längden på ett fragment mäts i ”enheter om åtta byte”. Ett giltigt värde skulle därför vara 1 000 byte nyttolast plus 20 byte header. Totallängden blir då 1 020 vilket inte överskrider MTU för länken. Routern behåller samma identitet som det ursprungliga paketet. Första fragmentet skickas iväg med flaggan More Fragments = 1 samt Fragment Offset satt till 0. Nästa fragment är det sista så flaggan More Fragments sätts till 0. Fragment offset sätts till 1 000/8 = 125. Totallängden för varje paket sätts till nyttolast plus 20 byte header.

För mottagarens IP-stack, det vill säga slutstationen kommer sedan fragment tas emot med flaggan More Fragments satt till ett. Det vill säga mottagaren förstår att detta inte är den sista delen av ett paket så den får mellanlagra detta fragment. Sedan kommer



Ett paket kan delas upp av en router i fragment.

fragment två, mottagaren ser på ID-värdet att dessa hör ihop. Flaggan More Fragments satt till noll gör att mottagaren kan avgöra att detta var det sista fragmentet.

Offsetvärdet tillsammans med flaggorna gör att pakten kan komma fram i godtycklig ordning. Om det andra fragmentet kommer först så skulle flaggan More Fragments vara satt till noll. Men Fragment Offset är satt till 125, mottagaren kan alltså avgöra att innehållet i detta fragment ska föregås av $8 \times 125 = 1\,000$ byte data.

Att undvika fragmentering

Fragmentering är något vi vill undvika. Det belastar routrar som helst ska ägna all kraft åt att vidareförmedla paket. Och vi har ju en gång delat upp dataflödet i paket, varför inte göra rätt från början?

Fragmentering kombinerat med paketförluster är riktigt dåligt. Om ett fragment saknas kan mottagaren inte återskapa det ursprungliga datagrammet (paketet). Efter ett tag kommer TCP upptäcka att ett paket saknas och skicka om det men nu med ett annat ID, samtidigt som det ofullständiga paketet ligger och tar plats hos mottagaren. Alltså måste mottagaren regelbundet städa bort misslyckade refragmenteringar.

En avsändare kan hindra fragmentering genom att sätta flaggan Don't fragment till ett. En router som tar emot ett paket som behöver fragmenteras kommer då att skicka tillbaka ett felmeddelande. Avsändaren har då möjlighet att skicka om paketet men med minskad paketlängd.

Ofta kan man i operativsystem eller register sätta vilken MTU man vill använda. Om man misstänker att någon slags IP-tunnling kan förekomma kan det vara en god idé att ställa ner MTU till 1 480 eller 1 460 för att ge plats åt extra headers. (IP-tunnling innebär att paket förses med en ny IP-header, till exempel för att kunna routas på Internet med en publik adress.)

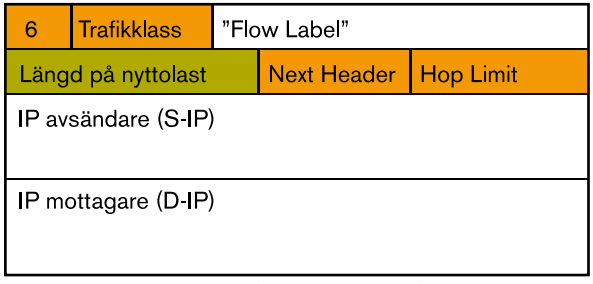
En sista möjlighet som vissa operativsystem använder är att använda det lokala nätets MTU då man adresserar noderna som sitter på samma nätverk medan man använder en mindre paketstorlek då paketen skickas via default gateway.

Kort om IPv6

På samma sätt som i IPv4 behövs tre saker på varje nod för att IPv6 ska fungera. En adress som alltså blivit fyra gånger så lång. En subnätmask, men nu har den bytt namn till ”längd på prefix” (samma noteringssätt som /n i IPv4). Och så slutligen en default gateway.

Det finns IPv6-adresser reserverade för multicast och adresser

avsedda att användas på testnät med mera, men de flesta IPv6-adresser kommer att vara publika och globala. De delas ut av operatörer på samma sätt som IPv4-adresser. Dessa adresser börjar med siffran 2001. Nästa två grupper är de som blir unika för varje organisation och anslutning. Grupp nummer fyra pekar på vilket nät inom organisationen som avses, varje företag får alltså drygt 65 000 nät, där varje nät kan innehålla miljarders miljarder noder. Denna grupp kallas på engelska för Subnet ID. De första fyra grupperna (eller /64) betecknar alltså nätnumret. De sista 64 bitarna är nodens unika nummer (*Interface ID*), ofta skapas den adressen från MAC-adressen. Man utgår i princip från MAC-adressen. Man utgår i princip från MAC-adressens 48 bitar och stoppar in FFFE i mitten. Detta kallas för en EUI-64 adress, den är lite mer komplicerad då även en bit inverteras i början (inverteringen gör att vissa adresser som :: blir enklare att komma ihåg). MAC-adressen 00-04-76-C2-C1-93 blir på detta sätt 204:76ff:fec2:c193.



Headern i IPv6 är en förenkling jämfört med IPv4.

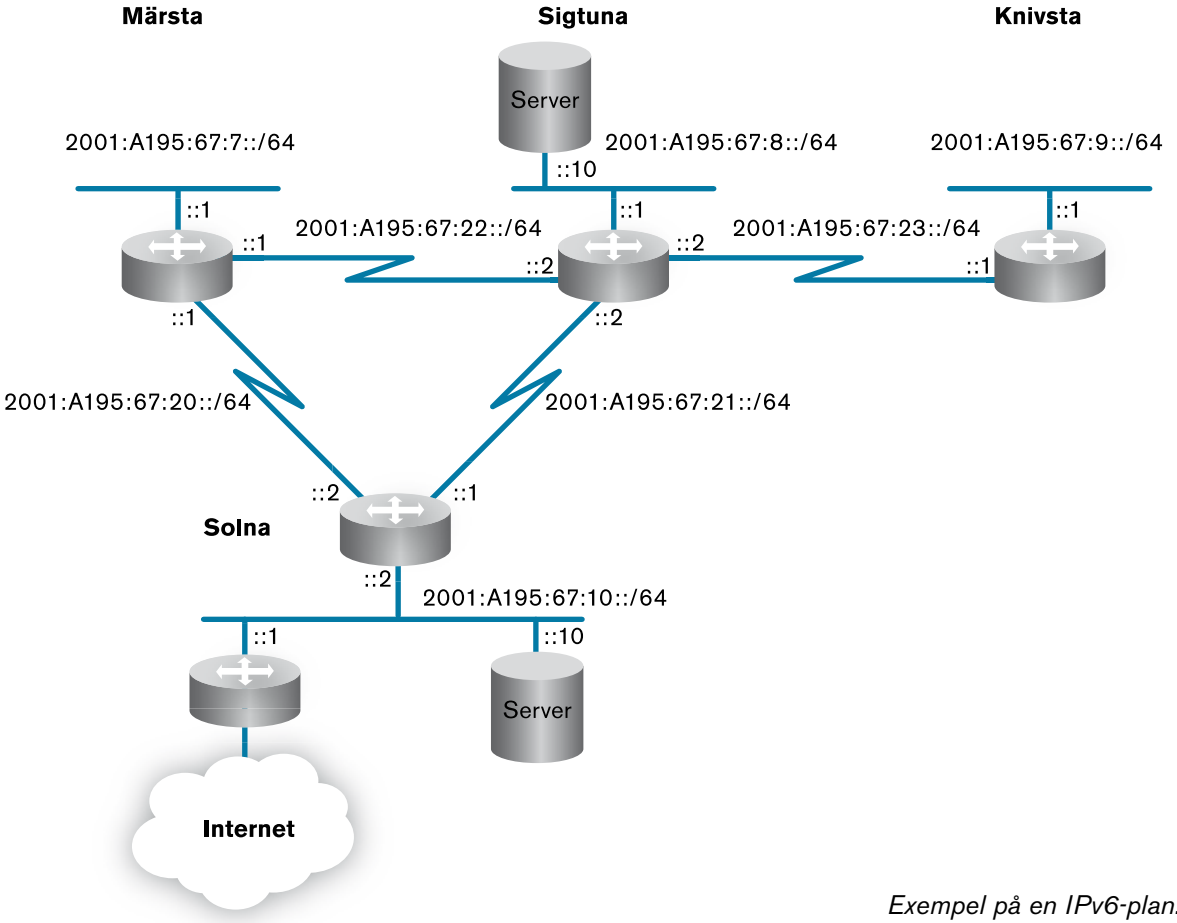
Headern i IPv6 är en förenkling jämfört med version 4. Funktioner som inte behövs så ofta har tagits bort eller flyttats. Trots att adressfälten är fyra gånger så långa ökar den totala längden bara från 20 till 40 byte. Flera av fälten från version 4 är kvar men några av dem har fått nya namn.

Versionsfältet är nu satt till 6. Fältet trafikklass består av en byte och tolkas på samma sätt som Diffserv i IPv4. Fältet "Flow Label" saknar motsvarighet i IPv4, med hjälp av IP-adresser och denna etikett kan flöden mellan noder hanteras på olika sätt. Fältet används ganska sparsamt, men en metod för hur fältet kan användas finns beskriven i RFC 3697.

Fältet "Längd på nyttolast" (*Payload Length*) kan innehålla värden upp till 65 536 och omfattar data i paketet samt de headers som kommer efter IPv6-headern. Det finns en möjlighet att hantera ännu större datalast, med hjälp av en annan header. Next header talar om hur nästa header ska tolkas. Nästa header kan vara nivå fyra som TCP eller UDP men det kan även vara en ny header för att hantera exempelvis IPsec eller routing. Next Header motsvarar proto-

kollfältet i version 4. Hop Limit ersätter TTL i version 4. Varje gång en router passeras minskas värdet med ett. Sist i headern kommer adressfälten som nu alltså är fyra gånger så långa.

Inga optioner används. Utan optioner får IPv6-headern en fast storlek vilket är en förenkling. Istället för optioner finns ett antal nya headers definierade som kan läggas till som en kedja innan nivå fyra-headern och nyttolasten. Behöver vi lägga till en option för



Exempel på en IPv6-plan.

routrar så blir det en routing header. Behöver vi autentisera så blir det en header för det, och på motsvarande sätt med eventuell fragmentering. Varje header innehåller fältet ”Next Header” så att mottagaren vet hur inkommande data ska tolkas. Den sista headern i kedjan pekar ut nästa header som TCP, UDP eller ICMP.

När arbetet med IPv6 startade på 1990-talet var autokonfigurering inte så vanligt inom IP, en användare knappade oftast in IP-adresser för hand. Idén med att routrar annonserar nätnumret och att en nod, för att skapa en unik adress, lägger till motsvarande en MAC-adress var beprövad inom andra tekniker än IP. Min personliga åsikt är att man överskattat denna betydelse och att de som designat IPv6 underskattat betydelsen av att ibland behöva arbeta med IP-adresser och inte DNS. Automatiska IPv6-adresser via EUI-64 adresser blir svåra att memorera. Jag tror därför att vi ofta kommer att använda DHCP för att dela ut IPv6-adresser och att de kommer ha ett enklare format som ::10, ::11 etc. På det sättet blir IPv6 inte krångligare än IPv4.

Adresserna startar med gruppen 2001 om de är publika. De två följande grupperna är unika per organisation. Nästa grupp är organisationens interna nätnumrering. Nodnumret består sedan av de sista grupperna. En IPv6-plan skulle alltså kunna ha utseendet enligt bilden på föregående sida (jämför med tidigare IPv4-plan i detta kapitel).

Kortfattat om ICMP

Ibland fungerar inte routing eller förmedling av paket på IP-nivån. Paket kommer inte fram eller en router hinner inte hantera paketet i tid. I sig är IP både förbindelselöst och baserat på den praktiska modellen går-det-så-går-det (best effort). Förbindelselös kommunikation innebär att paketet skickas utan att vi kontrollerat om mottagaren är redo att ta emot paket, eller att mottagaren över huvud taget existerar. IP behöver i princip inte ta hänsyn till de problem som kan uppstå. Problem som uppstår ska istället hanteras av protokollet ICMP, Internet Control Message Protocol.

I praktiken ska vi se ICMP som en del av IP. IP beskrivs i RFC

791 och ICMP i nummer 792. De har utvecklats för att vara beroende av varandra. När problem uppstår med att skicka ett paket vidare på IP-nivå så skickas normalt ett ICMP-paket tillbaka till den nod som står som avsändare på det paket som förorsakade problemet. Problemen kan till exempel vara:

- Vi hittar ingen bra väg till mottagaren, varken som nod eller nät (”no route to host” eller ”destination unavailable”).
- Paket har passerat för många routrar, vanligtvis beror detta på fel i routingtabeller så att paketet skickas runt i en loop. (Felet som uppstår kallas time exceed.)
- Parameter problem.

Det är även mekanismer i ICMP som används med felsöknings-kommandona traceroute och ping.

Referenser

RFC 791 Internet Protocol
RFC 792 Internet Control Message Protocol

I äldre dokument används bara klasser och begreppet subnät förekommer inte. Begreppet introduceras i:

RFC 917 Internet Subents
RFC 940 Toward an Internet Standard Scheme for Subnetting

RFC 1541 DHCP
RFC 2131 DHCP (en uppdatering av RFC 1541)

Protokollet DHCP använder samma portnummer som BootP och hanteras av routrar som BootP. För att läsa om hur routrar hanterar DHCP (en viktig fråga angående till exempel hur DHCP Discover ska skickas vidare) måste man alltså läsa äldre dokument om hur BootP hanteras.

RFC 1519 Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy

TCP- och UDP-nivån

- I det här kapitlet går vi igenom hur UDP och TCP använder portnummer. TCP:s trevägs handskakning förklaras. Headers för TCP och UDP gås igenom. Flödeshanteringen i TCP inklusive sliding windows förklaras kortfattat.
- Kapitlet är tänkt att kunna läsas fristående.

TCP-nivån på din dator kan du inte göra så mycket åt. Det finns inte så mycket att ställa in, och programmen väljer själva om de ska använda TCP eller UDP. Men det betyder inte att du inte har nytta av att känna till hur TCP och UDP fungerar. Det är TCP och UDP som ser till så att flera program kan använda en och samma IP-nivå. Det är flödeshanteringen i TCP som måste fungera effektivt så att kanalen mellan sändare och mottagare utnyttjas så effektivt som möjligt.

I kapitlet IP-grunder tittade vi på de funktioner som TCP lägger till:

- TCP gör kommunikationen förbindelseorienterad
- TCP lägger till portnummer
- TCP delar upp dataflödet i lagom stora paket
- TCP lägger till sekvensnummer på paketen så mottagaren kan kvittera vilka paket som kommit fram
- TCP hanterar omsändning av paket

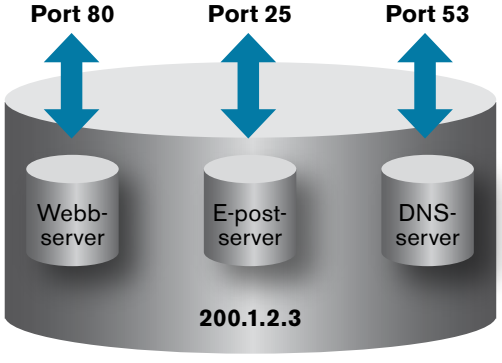
Hur paket ska sändas om och kvitteras är relativt komplicerat om det ska ske på ett effektivt sätt. Och vad som är effektivt beror på sammanhanget. Vi lägger in hela den här hanteringen under begreppet flödeshantering nedan.

Portnummer

Applikationer får inte kommunicera med IP direkt.

I arkitekturen kring TCP/IP finns inte så många fundament men ett av dem är att en applikation inte pratar direkt med IP. TCP eller UDP ska finnas i mellan och lägga till portnummer så att flera applikationer delar på IP:s funktioner. Man kan också se det som olika nivåer. Med hjälp av IP adresserar vi själva datorn, så på IP-nivå handlar det om kommunikation mellan datorer. Med hjälp av TCP och UDP adresserar vi applikationer (som vi också kan kalla program eller processer). TCP och UDP möjliggör kommunikation mellan applikationer.

Med hjälp av portnummer blir det bland annat möjligt att starta fler samtidiga filöverföringar och hämta filer från fler servrar på en gång eller att öppna flera fönster i en webbläsare och öppna fler sessioner på en gång. Med hjälp av portnummer kan en server vara webbserver och e-postserver på en gång.



En server kan lyssna på fler olika portnummer.

Det finns möjlighet att använda 65 536 olika portnummer (2^{16}). De 1 023 lägsta numren kallas för ”wellknown ports” och används av välkända kommunikationsprotokoll för filöverföring, e-post, namnuppslagningar, surfning och mycket mer. Vilket nummer ett protokoll använder finns bland annat fastslaget i RFC 1700 (Assigned Numbers). RFC 1700 är sakligt sett inte aktuell längre (den är inte ”standards track”) men i stort sett följs den fortfarande. En uppdaterad lista kan hämtas på:

http://www.iana.org/assignments/port-numbers

Portnummer	Protokoll	Anm
20	FTP	FTP Kontroll
21	FTP Data	
22	SSH	Secure Shell
23	Telnet	
25	SMTP	
53	DNS	
67	BOOTP	BOOTP och DHCP servers
68	BOOTP	
80	HTTP	
88	KERBEROS	
110	POP	
111	RPC	Sun Remote Procedure Call
123	NTP	Network Time Protocol
137	NetBIOS	
143	IMAP	För e-post
161 och 162	SNMP	Nätverksövervakning
443	HTTPS	Secure HTTP
1080	SOCKS	Proxy Server
3389	Remote Desktop	

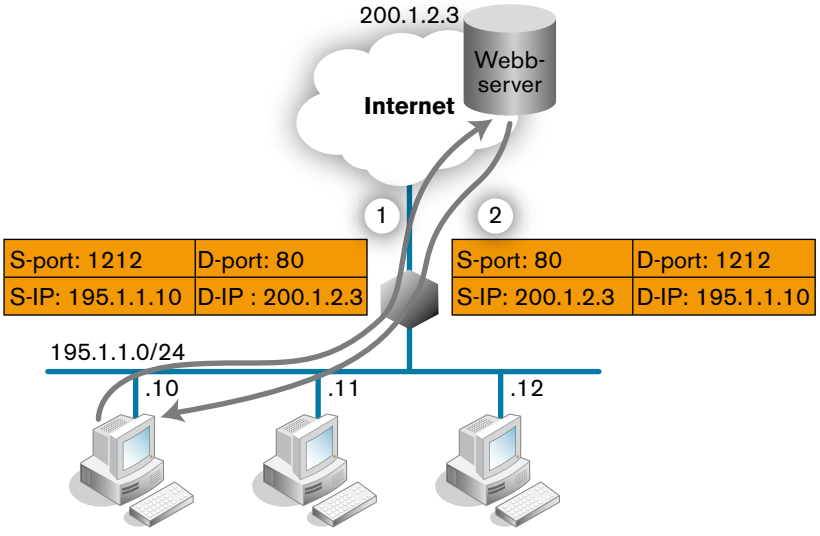
Några populära protokoll och de portnummer de använder.

Flera av dessa protokoll använder både TCP och UDP. Om möjligt behåller man i så fall samma portnummer. Ett exempel är DNS för namnuppslagningar. När enkla namnfrågor ska göras används UDP. Men när större överföringar ska göras mellan två namnservrar så använder de TCP. Allt detta med port 53 som serverport.

TCP/IP ger en möjlighet för alla slags program att kommunicera över alla slags WAN och LAN.

Servrar använder väldefinierade portnummer. Klienter däremot använder slumpartade portnummer med nummer över 1 023. Om vi startar en webbläsare och skriver in webbadressen till en webbsida så etableras en session från port 1 212 till mottagarport 80 i fallet nedan:

Webbklienten vet således att den ska etablera kontakt med port 80. När servern ska skicka tillbaka ett paket håller den bara reda på vem (IP-adress och portnummer) som skickat frågan. En session



Hur portnummer används.

identifieras inte bara av den egna IP-adressen och portnumret utan även av motsvarande information på andra sidan. Alltså kan servern returnera svaret till klientens portnummer.

Ska man vara noga så delas portnummer in i tre kategorier: Well Known Ports, Registered Ports och Dynamic Ports.

Du kan enkelt kontrollera vilka portar din egen maskin lyssnar på med hjälp av kommandot ”netstat -an”. Kommandot returnerar en tabell med de portar som är öppna för kommunikation (135, 455, 5225, 5226 och 8008 nedan). Adressen 0.0.0.0 innebär att datorn tar emot anrop från vilken annan adress om helst. Adress 127.0.0.1 innebär att programmet bara accepterar anrop från den egna maskinen (127.0.0.1 kallas även localhost). TCP/IP gör det alltså möjligt att bygga tjänster som bara kan användas av den egna maskinen eller från vilken maskin som helst på Internet.

Kommandot netstat visar även vilken status varje session har. I läget ”listening” agerar processen som server och ligger och väntar på anrop. I läge ”established” har den etablerat en session med andra änden och överföring av data är möjlig. I RFC 793, som beskriver TCP, finns de tillstånd som TCP kan ha beskrivna i en så kallad

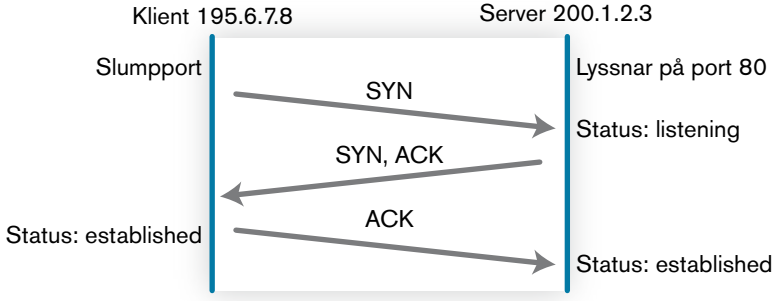
```
C:\>netstat -an

Aktiva anslutningar

  Prot.  Lokal adress             Extern adress           Status
  TCP    0.0.0.0:135              0.0.0.0:0               LISTENING
  TCP    0.0.0.0:445              0.0.0.0:0               LISTENING
  TCP    0.0.0.0:5225             0.0.0.0:0               LISTENING
  TCP    0.0.0.0:5226             0.0.0.0:0               LISTENING
  TCP    0.0.0.0:8008             0.0.0.0:0               LISTENING
  TCP    127.0.0.1:1025           0.0.0.0:0               LISTENING
  TCP    127.0.0.1:1031           127.0.0.1:5225          CLOSE_WAIT
  TCP    127.0.0.1:1032           127.0.0.1:5225          CLOSE_WAIT
  TCP    127.0.0.1:1033           127.0.0.1:5225          CLOSE_WAIT
  TCP    127.0.0.1:1034           127.0.0.1:5225          CLOSE_WAIT
  TCP    127.0.0.1:1035           127.0.0.1:5226          ESTABLISHED
  TCP    127.0.0.1:1039           127.0.0.1:5225          CLOSE_WAIT
  TCP    127.0.0.1:1043           127.0.0.1:5225          CLOSE_WAIT
  TCP    127.0.0.1:1654           127.0.0.1:5225          TIME_WAIT
  TCP    127.0.0.1:1655           127.0.0.1:5225          TIME_WAIT
  TCP    127.0.0.1:5226           127.0.0.1:1035          ESTABLISHED
  TCP    127.0.0.1:5679           0.0.0.0:0               LISTENING
  TCP    127.0.0.1:8005           0.0.0.0:0               LISTENING
  TCP    127.0.0.1:10110          0.0.0.0:0               LISTENING
  UDP    0.0.0.0:445              *:*
  UDP    0.0.0.0:500              *:*
  UDP    0.0.0.0:1054             *:*
  UDP    0.0.0.0:4500             *:*
  UDP    127.0.0.1:123            *:*
  UDP    127.0.0.1:1900           *:*
```

Tabellen, som du får fram via kommandot ”netstat -an”, visar de portar som är öppna för kommunikation.

tillståndsgraf. I den bästa av världar skulle established och listening vara vanligast, men eftersom TCP inte kan ta för givet att alla paket kommer fram blir nedkoppling komplicerat och ett flertal tillstånd kan inträffa. Några exempel finns med ovan i form av TIME_WAIT och CLOSE_WAIT.

Handskakning

I TCP etableras en session med hjälp av handskakning i tre steg. Eftersom det är två jämlikar (peers) som ska etablera en session och kanalen under inte är tillförlitlig blir det lite omständligare än ett enkelt open/accept förfarande. De tre stegen motsvarar helt enkelt en begäran om att starta en session, en kvittens av detta följt av ett meddelande om att även andra sidan är intresserad och slutligen en kvittens.



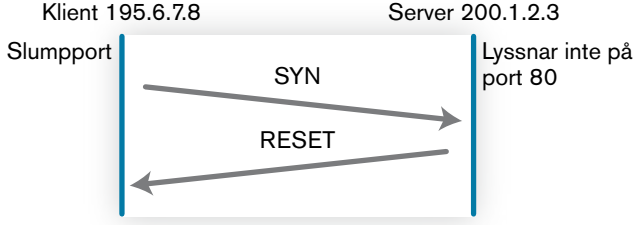
Handskakning i TCP.

TCP är förbindelseorienterat (connection-oriented är den engelska benämningen till skillnad från connectionless). Man handskakar och etablerar sessionen mellan två punkter innan överföring av information startar. När kommunikationen är etablerad kan den hanteras som en så kallad socket som identifieras av IP-adresser och portnummer i bägge ändar.

På motsvarande sätt avslutas en session med tre paket: FIN, FIN-

ACK samt ACK. FIN står givetvis för finish, och denna begäran kan komma från server eller klient. Vid avslutning av en session kan det dröja mellan de två FIN-paketen. Ena sidan kanske vill avsluta men den andra sidan anser sig ha mer data att sända. Först när bägge sidor skickat FIN och det sista paketet kvitterats är sessionen avslutad. Nedkoppling i TCP är komplicerad då TCP baseras på en otillförlitlig underliggande nivå. Avsändaren kan inte vara säker på att FIN-paketet kommer fram. Jämför med ett telefonsamtal över en mycket brusig förbindelse. Hur kan man vara säker på att få fram sitt ”klart slut” och är det mitt meddelande eller andra sidans kvittens som inte kommer fram? Man kan faktiskt bevisa att det inte går att göra tillförlitlig nedkoppling över en otillförlitlig kanal. TCP:s lösning är att starta olika timers så man tillslut betraktar förbindelsen som nedkopplad.

För att påskynda nedkoppling så finns även en möjlighet att använda RESET. Kommunikation bryts då omedelbart, inga kvittenser skickas och buffrar etc kastas. RESET används av TCP i vissa fall då ett oväntat paket mottages. RESET används även om en klient försöker anropa en port som inte används på en server som ett sätt att meddela att ingen process svarar på den porten eller att servern inte kan hantera fler inkommande sessioner.



Om en server inte lyssnar på en port kan den besvara uppkopplingsbegäran (SYN) med RESET. Klienten ska då förstå att denna port inte är aktiv. Metoden används bland annat av hackers vid en så kallad portskanning. Angriparen skickar SYN-paket till portar som kan vara öppna. Normalt får man inget svar eller RESET, om man istället erhåller SYN+ACK så vet man att porten är i lyssnande läge.

Dela upp dataflödet i paket

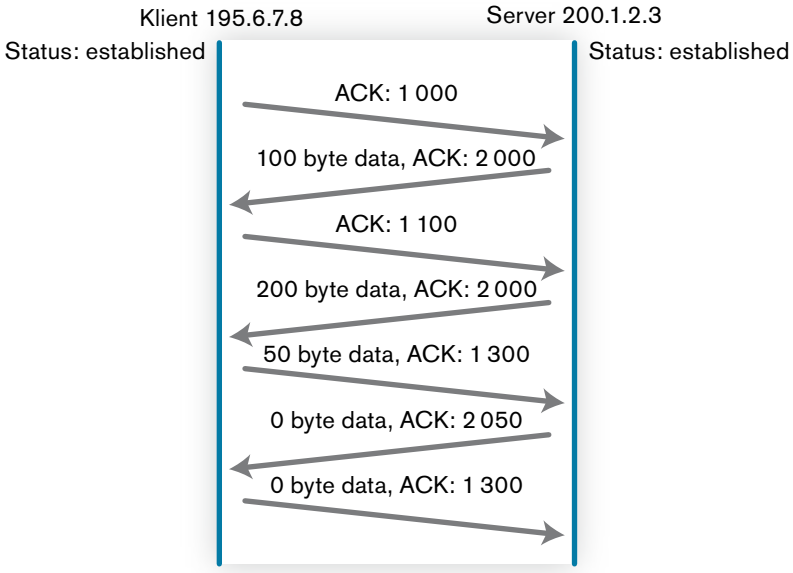
En av TCP:s viktigaste uppgifter är att upp mot applikationen leverera tjänsten ”ett kontinuerligt dataflöde” samtidigt som underliggande IP-nivå förväntar sig paket av en viss maximal storlek. TCP tar därför en liten genväg och kontrollerar vad länknivån har för MTU (Maximum Transmision Unit) och sedan delas dataflödet upp i paket. Paketen numreras och kvitteras på ett sofistikerat sätt.

Ett enkelt protokoll är att man numrerar varje paket med start på ett och sedan ökar med ett. TCP är mer komplicerat och hämtar första värdet från en räknare som går hela tiden. Om vi tänker oss att en användare eller en process skulle starta fler sessioner inom en kort tid så hade det varit en nackdel om alla startar på ett. Sannolikheten ökar då att man felaktigt kvitterar ett paket. Och det är just sekvensnummer man synkroniserar med SYN-paketet, det första sekvensnummer som används i sessionen (SYN står för synchronize). Att sekvensnumren startar på ett nummer som är unikt och jobbar med höga siffror gör det dessutom svårare för en angripare att ta över eller koppla ner en session. (En angripare kan ju enkelt krångla till det för en användare genom att skicka nedkopplingspaket (RES-flaggan) till den server som användaren kommunicerar med.)

För det andra så kvitteras inte varje enskilt paket. Istället kvitterar mottagaren hur många byte som tagits emot i form av nästa förväntade sekvensnummer. Avsändaren räknar på motsvarande sätt ut vad mottagaren borde kunna kvittera (senast kvitterade nummer ökat med antal skickade byte). Fördelen med detta är lätt att förstå:

Paket nummer tre, det med ACK = 1 100, behöver inte skickas i sekvensen utan servern kan från paket fem se att jämfört med paket ett har mottagaren tagit emot 1 300–1 000 = 300 byte vilket stämmer med antal skickade byte.

Att inte varje paket behöver kvitteras har stor betydelse för prestanda. Om TCP ska fungera över kontinenter så skulle kvittens av varje paket dra ner prestanda då ”roundtrip delay” kan vara över hundra millisekunder. Istället kan TCP kvittera fler paket på en gång. I Windows XP skickas kvittenser tidsstyrt var 500 millisekund. Man utnyttjar i TCP även möjligheten att skicka kvittenser samtidigt



Klienten behöver egentligen inte sända paket nummer tre (se text).

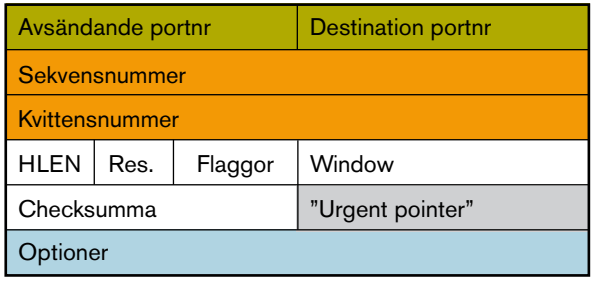
som man skickar data (det är ju bara att sätta giltiga värden i ACK-fältet). Tekniken att om möjligt skicka kvittenser och annan kontroll-information samtidigt som man skickar data brukar kallas piggy-backing.

För att dessutom öka prestanda och tillförlitlighet använd en teknik som kallas sliding windows, men mer om den senare.

TCP-Headern

De funktioner som TCP behöver hantera lägger vi in i headern. I bilden på nästa sida visas headern med fyra byte per rad. Totalt är headern 20 byte lång plus eventuella optioner. Maximal längd på headern är 60 byte.

De fyra första byten anger avsändande respektive mottagande portnummer. Fyra byte används för sekvensnummer, vilket gör att



TCP-header.

sekvensnumret kan räknas upp till drygt fyra miljarder, vilket normalt är tillräckligt högt för att inte sekvensnummer ska upprepas under en session. Kvittensnumret används för att kvittera hur många byte som kommit fram.

HLEN anger längden på TCP-headern och räknas precis som med IP i 32-bitars ord. Typiskt värde är alltså fem motsvarande 20 byte. HLEN består av fyra bitar och därför blir maximal längd på headern 60 byte. Sedan följer fyra till sex bitar som är reserverade för framtida användning.

För att hantera handskakning med mera används ett antal flaggor:

URG Urgent. Används ej i modern TCP
ACK Anger att kvittensfältet har ett giltigt värde
PSH Push
RST Reset. Kan användas vid nedkoppling av en session
SYN Synchronize. Används vid etablering av en session
FIN Finish. Användas vid normal nedkoppling av en session.

ACK-flaggan har i stort sett alltid värdet ett eftersom man passar på att kvittera med hur mycket data som mottagits. RST, SYN och FIN används vid upp- och nedkoppling av en förbindelse.

PSH-flaggan används mycket och hänger samman med en funktion inom TCP som heter push. Det är TCP som avgör när data ska sändas, därför kan TCP samla ihop data innan det skickas iväg. En ovanpåliggande applikation kan dock skicka kommandot ”push” vilket meddelar TCP att detta data ska skickas och inte får buffras. Dessutom sätts PSH-flaggan i motsvarande paket. Mottagande TCP-stack ska på motsvarande sätt inte buffra inkommande data utan skicka upp det till mottagande applikation omgående.

Dessa är de klassiska flaggorna från RFC793, men tillägg finns i RFC2481, A Proposal to add Explicit Congestion Notification (ECN) to IP. Före Urgent-flaggan finns då två flaggor: Congestion Window Reduced (CWR) och ECN-Echo. Dessa används för att förbättra flödeshanteringen i TCP.

Fältet Window används för att hela tiden meddela den andra

sidan om hur stor buffert man har. Det vore ju ett slöseri att skicka en massa data som sedan inte mottagaren har möjlighet att bearbeta.

Checksumman används för att kontrollera riktigheten i headern samt lite mer. Se avsnittet om UDP.

Fältet ”Urgent Pointer” har som man kan misstänka att göra med flaggan Urgent. Varken flaggan eller fältet används i modern TCP.

TCP-optioner

Eftersom flödeshantering och felhantering är ett område som utvecklats och fortfarande utvecklas används optioner till TCP relativt ofta. När en session etableras kan sändare och mottagare handskaka om vilket stöd för flödeshantering de använder.

En typisk option kan ha värdet 0x0204 05b4 0101 0402. Detta är en option om åtta byte som Windows XP kan annonsera.

- 02 anger typ av option, här ”Maximum Segment Size”
- 04 anger optionens längd i byte
- 05b4 är det hexadecimala talet 1460, dvs maxlängden när man tar emot paket. Annat relativt vanligt värde är 1 380 decimalt.
- 0101 är NOOP (no operation)
- 0402 är flaggor som meddelar att man har stöd för SACK enligt RFC 2018.

Hur stora paket som ska användas under sessionen är en svår fråga. Det optimala om man ska skicka mycket data är så stora som möjligt men utan risk för att fragmentering inträffar. Detta är svårare än det låter eftersom IP-nivåns routrar kan ändra den väg som paketen routas, så MTU kan variera över tiden. Ett annat problem är att IP kan välja att lägga till optioner eller tunnla IP i IP eller i IPsec, detta påverkar hur stor last vi kan skicka med. Det här därför blivit allt vanligare att man inte använder 1 500 byte (typiskt värde för Ethernet v II) utan går ner lite till 1 460 för att ha marginal.

Windows Scaling Option är en annan relativt vanlig option. I

grunden har TCP stöd för en buffert om 64 Kbyte. Om kanalen har hög bandbredd men lång fördröjning är detta en begränsning, detta kan gälla en kanal över mobila nät, via satellit eller mellan kontinenter.

Om vi tänker oss att vi har en förbindelse om 8 Mbit/s och en fördröjning (RTT Round Trip Time, det vill säga fram och tillbaka) om 100 ms så behöver man en buffert om 8 000 000 × 0,1 = 800 kbit vilket motsvarar 100 kbyte. (Avsändaren bör kunna hålla hela kanalen full med data tills paketen har kvitterats. Dubblar vi fördröjningen behövs en dubbel så stor buffert.) 64 kbyte är ett gammalt värde. Om vi tänker oss en satellitkanal om 500 ms fördröjning (RTT) så skulle detta ge en kapacitet om 64 × 8 × 1 024/0,5 vilket ger cirka 1 Mbit/s. Snabbare kanaler än så skulle inte ge någon prestandaförbättring på grund av TCP:s begränsningar. Idag med Gigabithastigheter uppträder problemet redan vid 0,5 mikrosekunder. För att kunna använda större buffrar finns optionen Windows Scaling Option. Man meddelar vilken faktor fönsterstorleken ska multipliceras med under handskakningen vilket gör att mångdubbelt större buffert kan användas.

Windows Scaling är relativt grundläggande kommunikationsteori men tyvärr relativt okänd bland systemarkitekter och designers. Organisationer köper snabba internationella förbindelser men kombinerar dessa med gamla TCP/IP-implementationer från 1990-talet som ska göra backup över nätet och det kan vara en dålig kombination. En snabb analys kan visa om Windows Scaling används, annars måste man byta TCP/IP-stack. Problemet finns även med enkla handdatorer och långa fördröjningar över mobila tjänster.

Brandväggar kan ha svårt att hantera Windows Scaling. I Windows XP är det därför förinställt som en avslagen funktion. I Vista däremot är det påslaget som standard, men går att slå av via kommandoläget om man misstänker att det orsakar problem.

Ytterligare en option är tidsstämpling vilket kan användas för att beräkna kanalens fördröjning vilket bland annat kan användas för att beräkna när en kvittens borde tagits emot.

Flödeshantering

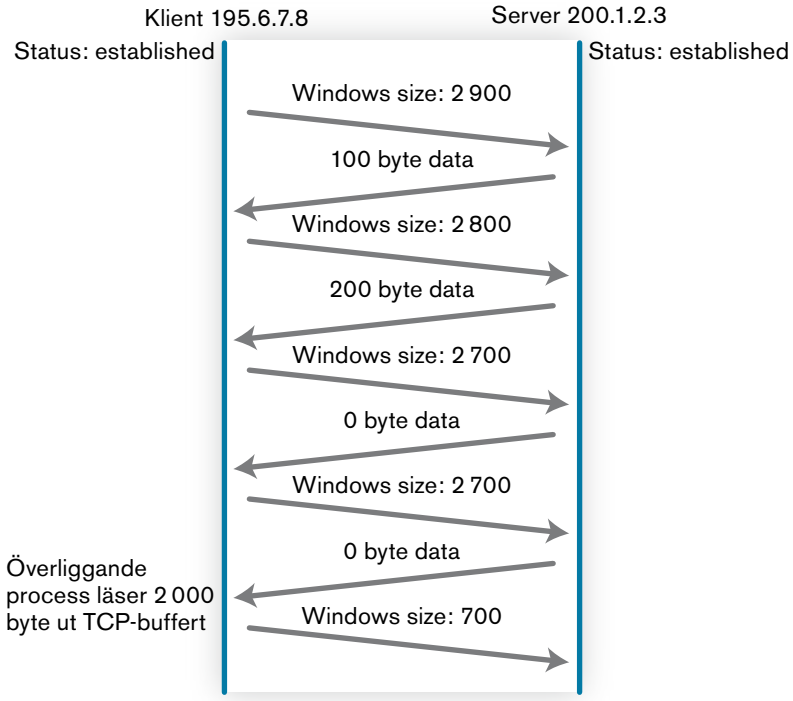
TCP är ett exempel på hur Internet designats. Jämfört med telefonnätet så ligger många funktioner i TCP hos avsändare och mottagare och inte i själva nätet. Routrar skickar paket vidare så snabbt de kan, det är i TCP som funktioner för att hitta borttappade och duplicerade paket ligger. På engelska kallas denna design för edge centric. I själva verket är det ett komplext samspel mellan hur TCP försöker utnyttja nätet och hur routrar hanterar dataströmmar.

För att uppskatta hur effektiv och tillförlitlig flödeshantering TCP använder så bör man ha använt ett tidigare protokoll som Kermit eller X-modem. Det kändes ofta som att det räckte med att någon stängde en dörr ovarsamt i närheten så avbröts kommunikationen och man fick börja från början. Äldre protokoll var känsliga för störningar och utnyttjade inte kanalen så effektivt. Det är lätt att se hur effektivt TCP är. Om man sätter sig på en långsam förbindelse på exempelvis 256 kbit/s och ska överföra en fil om 8 Mbyte så borde det minst ta $8 \times 8 \times 10^6 / 256 \times 10^3 = 0{,}25 \times 10^3 = 250$ sekunder. Ofta når man 70–90 procent av detta teoretiska värde. Använder man en applikation som visar momentana värden kan man också se hur överföringshastigheten ökar successivt för att hitta kanalens maximum. Man kan också prova med att dra ut sladden några sekunder och sätta tillbaka den. TCP kommer oftast att klara störningar. Vi ska titta översiktligt på de metoder TCP använder.

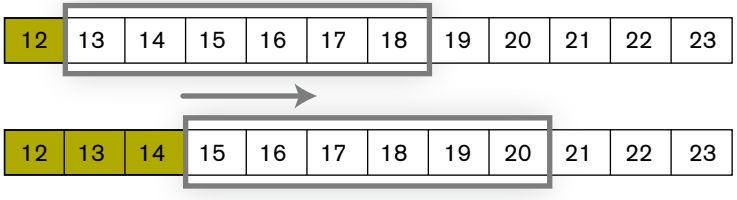
Testa själv

För att inte skicka över data som inte kan tas emot meddelar varje nod hela tiden hur stor mottagande buffert den har. Det är till detta som fältet ”Windows” används. I början av en session skickas denna information men buffertens storlek varierar med tiden. TCP skickar ju upp informationen till mottagande process men det varierar i vilken hastighet applikationen tar emot data.

För att kunna skicka data effektivt använder TCP en metod som kallas sliding windows. Sliding windows används även av andra program som Z-modem. Sliding windows gör processen mer komplex men prestanda, speciellt vid borttappade paket, blir avsevärt bättre. Vi tittar på en lite förenklad modell jämfört med vad som används i



Under sessionen meddelar TCP den andra ändpunkten hur stor buffert den har för mottagning.



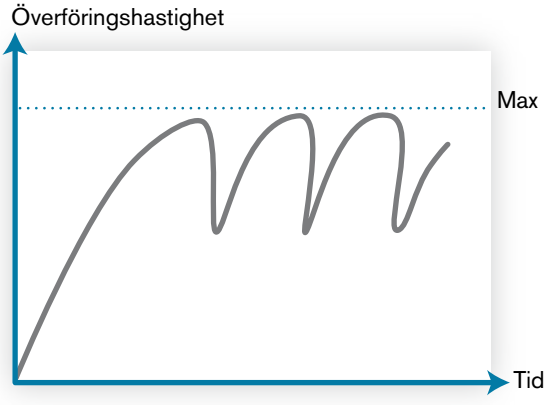
Sliding windows.

TCP. TCP använder som bekant antal byte i sin sekvensräknare och för att kvittera.

Om vi har en fönsterstorlek om sex byte ovan och tänker oss att byte 12 är kvitterad så kan byte 13–18 sändas iväg. Om sedan byte 13–14 kvitteras kan fönstret glida fram och skicka byte 19 och 20. Om något händer och till exempel 13 inte kvitteras men däremot 14–17 så får vi sända om byte 13. När sedan 13 kvitterats kan fönstret glida fram till 18 och skicka fem nya byte.

Sliding windows ökar prestanda betydligt jämfört med om varje paket ska kvitteras för sig.

Hur snabbt TCP skickar ut paket får en avgörande betydelse på prestanda. Modern TCP använder flera tekniker. Ett grundläggande antagande är att kanalen mellan två ändpunkter har en maximal kapacitet och när paket försvinner beror det på trafikstockning (congestion). Mottagaren eller en switch eller router på vägen har inte hunnit hantera paketet riktigt. Om paket inte blir kvitterat så minskar TCP överföringshastigheten genom att öka tiden innan den sänder nästa paket. På detta sätt undviker TCP att skicka ut en massa paket på ett nätverk som inte hinner med. Om det går bra så minskar den tiden mellan paketen igen. Vi får alltså en kurva som i princip ser ut som här bredvid.



Schematisk bild av hur TCP utnyttjar en kanal.

När TCP startar en ny session tar det tid innan kanalen utnyttjas till max, detta kallas för slow start och har visat sig vara en bra metod. Flödeshantering och omsändning i TCP är komplext och vi kommer att återkomma till det i ett senare kapitel. Men redan med denna enkla modell ser vi en fördel med TCP. Om vi startar en ny överföring som konkurrerar om samma kanal så kommer TCP att backa och den första sessionen ger plats även för den andra. TCP har alltså en slags automatik som gör att flera sessioner delar på befintlig bandbredd och TCP undviker att belasta nätverk i onödan.

TCP har funktioner som gör att flera sessioner delar på befintlig bandbredd och TCP undviker att belasta nätverk i onödan.

Hur fungerar UDP?

User Datagram Protocol, UDP, innehåller minsta möjliga funktion, i princip lägger vi bara till portnummer för avsändare och mottagare. UDP-headern består av åtta byte.

Avsändande portnr	Destination portnr
Längd	Checksumma

UDP-header.

Längdfältet anger hur långt UDP-paketet är inklusive last (mäts i byte). Vidare finns ett fält för att beräkna checksumma för att verifiera headern och lite mer. Avsändaren har en möjlighet att bestämma om checksumman ska användas eller inte. Oftast används funktionen, den kontrollerar riktigheten i överfört datainnehåll (lasten). Eftersom IP inte beräknar checksumma inklusive last så gör funktionen nytta på UDP-nivån. Avsändaren får också sätta fältet till noll vilket betyder att checksumma inte har beräknats. Ett protokoll kan ju utvecklas för att bara fungera lokalt över ett tillförlitligt LAN. Ifall beräkningen av checksumman skulle ge värdet noll, så betecknar man istället detta med värdet oxFF.

Både TCP och UDP använder samma princip för att beräkna checksumma. I beräkningen ingår TCP/UDP-headern, en pseudo-header samt lasten.

Både TCP och UDP använder samma princip för att beräkna checksumma. Checksumman beräknas på:

1. Själva TCP/UDP-headern (där man före beräkningen sätter fältet för checksumman till noll).
2. En pseudo-header vars innehåll främst kommer från IP-nivån. Som innehåll använder man avsändarens och mottagarens IP-adress, protokollfältet samt längdfältet (från TCP respektive UDP).
3. Nyttolasten.

Anmärkning:
Pseudo-headern har främst haft historisk betydelse eftersom varken TCP eller IP kunnat lita på att data från nivå två är korrekt. I modern datakommunikation används bättre checksumme-kontroller som CRC-32 på nivå två. Pseudo-headern påverkas även vid adressöversättning (NAT).

Poängen men en pseudo-header är att vi tillsammans med TCP/UDP-headern kan kontrollera att informationen har nått rätt mottagare avseende både adresser, protokoll och portnummer. Är inte denna information rätt behöver vi inte behandla paketet.

UDP ger alltså i stort sett bara multiplexeringsfunktion så att flera processer/program kan dela på IP-nivån. Inga kvittenser görs, så vi har ingen kontroll på att om paketen kom fram. På samma sätt som i IP får vi ett nät som använder ”best effort”. Kommunikationen är förbindelselös, vi har alltså ingen kontroll på om mot-

parten vill eller kan ta emot paket. Eller om mottagaren ens existerar.

I sin enkelhet används UDP flitigt:

- protokollet är enkelt och kräver lite resurser
- ibland har vi inte tid att vänta på omsändning
- vi kan få överföringsfel (lokalt nät)
- vi slipper etablera session, det är effektivt om vi ska göra något enkelt som att bara skicka en fråga och få ett svar
- när vi vill utveckla en egen flödeshantering
- vid tidsstyrda protokoll där man skickar om data regelbundet, exempelvis vissa routingprotokoll.

Hur påverkas TCP och UDP av IPv6?

I en skiktad arkitektur ska vi kunna byta protokoll på nivå tre utan att det påverkar nivå fyra, transportnivån. I praktiken kommer flera noder ha stöd för både IPv4 och IPv6 och då behöver vi en mekanism som styr vilket protokoll som ska användas. Problemet behandlas i ett eget avsnitt under kapitlet IP-baserade program.

Nivå fyra och framtiden

TCP och UDP har bägge förtjänster men i takt med att IP används även inom avancerade tjänster för telekom så ökar kraven. Ett protokoll som passar bättre för bland annat signalering har utvecklats inom IETF. Protokollet heter SCTP (Stream Control Transmission Protocol).

Även inom ramen för hur TCP fungerar går det att utveckla flödeshanteringen så att den anpassar sig bättre för exempelvis trådlösa förbindelser. Samverkan mellan TCP, UDP och hur IP tappar paket är komplext. Här görs fortfarande arbete och nya tekniker tas fram.

Referenser

RFC 793	Transmission Control Protocol
RFC 768	User Datagram Protocol
RFC 2001	Handlar om TCP Slow Start, fast retransmit med mera
RFC 2018	TCP Selective Acknowledgment Options
RFC 2481	A Proposal to add Explicit Congestion Notification (ECN) to IP

IP-baserade program

- Det här kapitlet behandlar några klassiska TCP/IP-baserade program. Främsta fokus är HTTP men även lite enklare applikationer som telnet och FTP behandlas.
- Kapitlet är tänkt att kunna läsas fristående men fungerar bäst om du vet hur TCP/IP fungerar.

Så har vi kommit fram till programmen. Själva vitsen med TCP/IP är ju att alla typer av program ska kunna kommunicera med varandra oavsett hur kanalen eller länken ser ut. Vi har tidigare konstaterat att vi förväntar oss att visst stöd och vissa program ska ingå i Internetprotokollen. Ofta har program och protokoll blivit synonymer.

Det finns ett par program som i princip inte dokumenterats. Hit hör traceroute och ping, hur de fungerar behandlas bland annat i kapitlet Felsökning i TCP/IP-miljö. Traceroute används av tusentals användare i hundratals länder. Trots det finns det väldigt lite skrivet om hur programmet kan och borde fungera, och de beskrivningar som finns tillkom efteråt. Programmen och kommandona är så gamla så det kommer från en tid då man i Unix hade den informella principen att ”all dokumentation finns i källkoden”. Ett program blev de facto standard och tillslut ett eget protokoll.

Telnet

Att kunna logga in på en annan dator via en fjärranslutning är en gammal idé, det var bland det första man ville göra med Arpanet. I en grafisk miljö brukar man prata om remote desktop, i textbaserade system om remote login. Telnet hanterar textbaserade system.

Principen kan verka enkel: det som skrivs på den ena datorns tangentbord ska utgöra input på andra sidan. Det som utgör output på en sida ska visas på en textbaserad konsol på andra sidan. Men

det är svårt och behandlas i en mängd RFC:er som 854, 859–861 och många fler (och tyvärr behandlar varje RFC bara delar av protokollet). Svårigheterna har med en mängd optioner att göra, att applikationen ligger nära operativsystemet och hårdvaran (en miljö kanske använder knappt 50 tangenter och en annan runt 75) men även med principen och att göra om enstaka tangentnedtryckningar till paket på ett effektivt sätt. Fortfarande är textbaserade system vanliga, de kan göras väldigt inmatningseffektiva, den datamodell som de arbetar med är väl beprövad och de är lätta att nå: i stort sett vilken dator som helst med TCP/IP kan vara en telnet-terminal. I Linux är tangent och konsol fortfarande standard input respektive output. I Windows finns stöd för telnet i konsolläge och i programmet Hyperterminal.

Liksom många andra applikationer bygger telnet och FTP på TCP.

Telnet bygger på TCP, vi vill ju att dataöverföringen ska vara tillförlitlig och visa samma data i bägge ändar. Telnet använder ofta push-funktionen i TCP för att ge en snabbare användarupplevelse.

För att förhandla fram optioner och inställningar som 7- eller 8-bitars ASCII, hur man ska hantera kombinationer av Carriage Return samt Line Feed etc så används ett begrepp vid namn NVT, Network Virtual Terminal. Både klient och server tar omvägen över NVT, det gör att man enklare kan anpassa ett telnet-program för varje maskins egenheter.

Till telnet finns en mängd optioner. Både klient- och serversidan kan meddela att de avser att använda en option och den andra ändan får bekräfta eller stoppa. Några exempel på optioner är ECHO (tillåta lokalt eko) och end-of-record. NVT ser till att bägge ändar kan kommunicera överhuvudtaget.

Telnet kan användas på andra portar än nummer 23. Detta är praktiskt vid felsökning med mera.

Testa själv

Telnet ställer inga krav på att det verkligen är ett tangentbord eller en konsol som skickar eller tar emot data. Detta gör att telnet kan användas för att testa förbindelser och protokoll. Det vi skickar från ena sidan kommer fram till den andra. Genom att använda telnet på andra portar än port 23 kan telnet användas vid felsökning. Vi kan till exempel telnetta till en webbserver på port 80 med kommandot ”telnet www.firma.se 80”. Sedan skickar vi en slumpartad textsträng och trycker Enter två gånger. Webbserver kommer då svara med att vårt HTTP-anrop var fel. Men då ser vi att webbservern svarar. Om vi kan syntaxen för HTTP kan vi dessutom skicka

korrekta kommandon till servern och se hur den svarar. Samma sak gäller bland annat protokollen SMTP, POP3 och FTP.

Telnet har dålig säkerhet. Kommunikationen sker i klartext vilket gör att användaridentitet och lösenord kan avlyssnas enkelt. Detta gör att användningen av telnet bör begränsas. En följd av detta är också at flera Unix-varianter idag inte tillåter att man loggar in sig med root-behörighet via telnet.

Att all kommunikation sker i klartext, även felhantering och intern förhandling mellan klient och server, är säkerhetsmässigt en nackdel. Men för felsökning är det en fördel. Man kan enkelt koppla in en analysator och se var problemen uppstår. Detta är en egenskap som flera TCP/IP-baserade program delar och ett skäl till att Internetprotokollen blivit populära. Felhanteringen är dessutom relativt utförlig, vid förhandlingar om optioner inom applikationer sker mycket i klartext och inte bit-orienterat (via flaggor). Om den ena sidan inte förstår skickar de ofta meddelanden med text ”the client sent a message the server can’t understand” eller något liknande.

Felmeddelanden och intern kommunikation i klartext gör felsökning enkel men är en säkerhetsrisk.

Secure Shell – SSH

Säkerheten i telnet är låg och inloggningssekvenser skickas i klartext. Eftersom det finns ett behov av fjärrterminaler har konceptet utvecklats vidare till bland annat SSH, Secure SHell. SSH har även blivit populärt genom en lättanvänd shareware vid namn PuTTY.

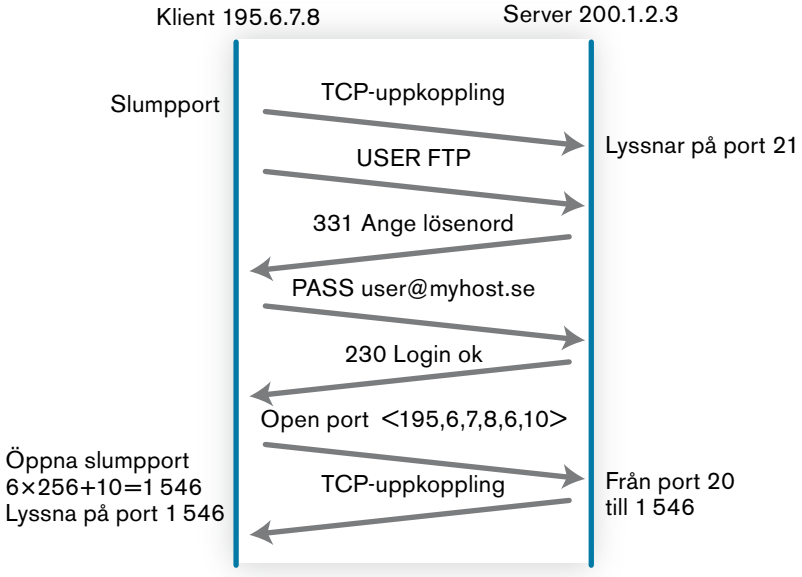
SSH ger möjlighet till autentisering via PKI-teknik. Med hjälp av serverns publika nyckel och ett fingeravtryck kan klienten autentisera servern. Den publika nyckeln kan sedan användas för att upprätta krypterad kommunikation mellan klient och server. På samma sätt som det går till i HTTPS förhandlar sedan klient och server fram ett symmetriskt krypto och en sessionsnyckel för att spara CPU-belastning.

En funktion till som gjort SSH populärt är en funktion för port forwarding. SSH kan konfigureras så att en godtycklig applikation och port kan använda SSH, och SSH kan således skapa en krypterad tunnel mellan två noder.

File Transfer Protocol – FTP

Program för att överföra filer tillhör också de äldsta tillämpningarna. De äldsta RFC:erna är från tiden innan TCP/IP. Klassisk FTP fungerar fortfarande på ett sätt som inte passar när brandväggar ska konfigureras, se nedan. Och precis som för Telnet är säkerheten svag, såväl kontrollinformation som inloggningar skickas i klartext.

FTP använder två portnummer. På port 21 skickas kontrollinformation för inloggning med mera. När sedan data ska överföras (vilket även omfattar en listning av vilka filer som finns att hämta) så startas dotterprocesser på servern och de använder port 20.



Klassisk FTP använder två portnummer och dataöverföring sker via en session som startas från servern.

Det som är lite udda i klassisk FTP är att dataöverföringen dels startar från servern sessionsmässigt (ett problem då brandväggar brukar acceptera inkommande SYN+ACK-paket men inte SYN-

paket), dels att det då är servern som har ett känt portnummer som initierar sessionen. Med FTP i passivt läge, även kallat PASV, så löser man detta problem och PASV är numer oftast förvalt flera klienter. Ett klassiskt undantag har varit FTP i MS Windows kommandoläge. FTP och speciellt passivt läge behandlas även i kapitlet om adressöversättning.

FTP använder en delmängd av protokollet telnet för att skicka kommandon och textsträngar. Även NVM används på ett sätt liknande telnet, men jämfört med telnet används bara ett fåtal optioner. På samma sätt som med telnet påverkar detta säkerheten negativt men det underlättar för felhantering.

Det finns varianter på filöverföring med högre säkerhet, där till exempel Kerberos används för inloggning eller SSL (Secure Socket Layer) används för att skydda såväl inloggning som överföring. En annan konsekvens av FTP:s svaga säkerhet är att FTP ofta används för anonym FTP. En användare loggar in med användarid ”anonymous” eller ”ftp” och anger sedan sin e-postadress som lösenord (av artighetsskäl så att servern kan logga hur servern används). Till stor del har webbservrar tagit över denna funktion men en del leverantörer väljer fortfarande att visa olika information på webb- respektive FTP-servrar.

En förenklad variant av FTP har utvecklats i form av Trivial FTP, TFTP. TFTP blev snabbt en succé och används flitigt då till exempel konfigurationsfiler ska överföras eller då system ska boota upp. TFTP utmärks av:

TFTP använder UDP och är ett väldigt enkelt protokoll för att överföra (huvudsakligen små) filer.

- ingen inloggning eller autentisering
- ingen möjlighet till kryptering
- UDP används som nivå 4-protokoll. Detta sparar både minne och kod. För att kontrollera att data kom fram används ett enkelt stop-and-wait protokoll där servern väntar på kvittenser innan nästa paket skickas. Både klient och servern har timers så att de kan sända om vid behov.
- Protokollet kan användas för att ta emot och sända filer (kallas READ respektive WRITE inom protokollet).

HyperText Transfer Protocol – HTTP

HTTP har utvecklats från att vara ett ganska enkelt protokoll för att överföra webbsidor (HTML och de objekt som hör till dem) till en generell och ganska komplex metod med flera varianter. HTTP används inte bara för att skicka webbsidor utan även i modern systemutveckling via regelverk som Web Services och SOAP. I detta kapitel går vi igenom dem översiktligt.

HTTP har flera egenskaper gemensamma med telnet och FTP. TCP används. Inloggning, kommunikation och felhantering sker i klartext. Detta gör felsökning enkel men det medför också säkerhetsproblem.

För att göra överföringen tillförlitlig använder HTTP protokollet TCP. För att avgöra vad som ska överföras används URL:er. Uniform Resource Locators är en specifik form av den mer generella formen URI, Uniform Resource Identifiers. Populärt kallar vi dem ofta ”länkar” och de har formen ”protokoll://domännamn [:portnummer] / sökväg [:parametrar] [?fråga]. Variabler inom hakparentes är optioner så en enklare form för endast HTTP blir endast ”http://domännamn/sökväg”. För att översätta domännamn används DNS. Sökvägen anger vilket dokument eller objekt som vi vill hämta. Normalt är det klienten (webbläsaren) som hämtar objekt från webbservern men metoden i sig är tvåvägs och servern kan hämta objekt från klienten.

I princip kan vi alltså få olika felmeddelanden beroende på om det är sökvägen som inte finns eller om det är namnuppslagningen som inte lyckas. Eller om namnuppslagningen lyckas men aktuell IP-adress inte svarar på port 80.

HTTP har ett par egenskaper som är värda att notera:

HTTP är tillståndslöst. Detta gör det enkelt att bygga servers, de behöver inte hålla reda på vad klienten gjort förr. Klienten frågar efter ett objekt (anger en sökväg) och får ett svar. I princip är överföringen sedan klar och sessionen stängs. Detta är HTTP:s grundfunktion, men ibland vill vi ha kontroll på vad som skett tidigare, det vill säga servern svarar olika beroende på vad klienten har gjort tidigare. Flera tillägg har utvecklats för detta. Ett sätt är att använda cookies (se sidan 79) ett annat sätt är att skicka status via själva URL:en. Följaktligen blir då URL:erna bärare av information och i princip för komplexa för människor. Följande är kanske inte en länk man tipsar sin kompis om: http://www.bolag.se/jtrac/flow?_flowExeKey=cC5728581-2A39-B914-6D50-8A7981D8CA5F-

k71AAD132-2501ABE5-D231-94460E3C4B6C&_eventId=back.

HTTP är tvåvägs. (En egenskap som delas med de flesta program.) Normalt efterfrågar klienten objekt eller egenskaper hos servern men motsatsen finns också. Dessutom finns kommandona PUT och POST för att klienten ska kunna överföra information till servern.

HTTP är byggt för caching och mellanhänder (proxies).

HTTP version 1.1 använder persistenta förbindelser. När klienten hämtat ett objekt (varav själva HTML-texten kan vara ett) så behålls TCP-förbindelsen öppen tills ena parten begär att den ska stängas. Detta ger flera fördelar för komplexa (moderna) webbsidor. En modern webbsida består av många objekt och flera av dessa kan överföras via samma session. HTTP version 1.0 öppnade en session per objekt. HTTP version 1.0 och dess teknik ökar handskakningen via TCP och dessutom tar det tid för TCP med slow start att uppnå bästa prestanda. HTTP version 1.1 ger en klar förbättring speciellt över länkar med lång fördröjning.

Testa själv

För att testa HTTP kan du enkelt använda telnet och fråga efter en webbsida.

```
C:> telnet www.twoviews.se 80
GET /index.html HTTP/1.1
```

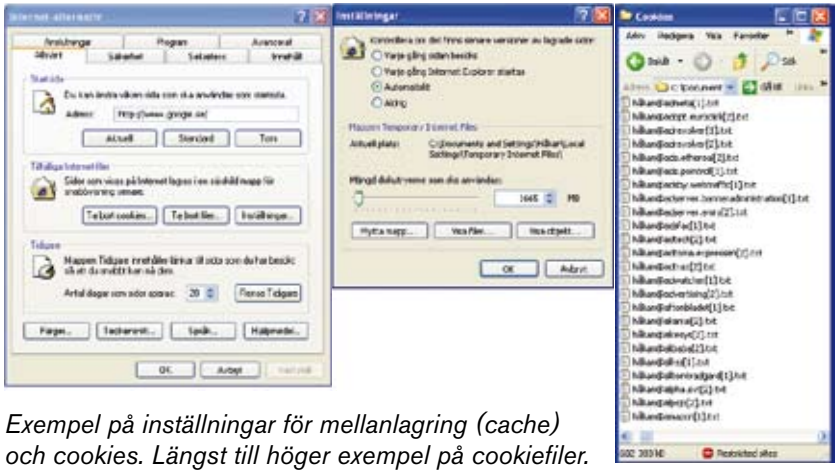
Slå sedan Enter två gånger och din terminal ska få en mängd HTML-kod. En webbläsare ser sedan vilka objekt som ingår och fortsätter fråga efter dem.

HTTP har utvecklats från att i grunden vara ett enkelt protokoll till en komplex samling tekniker och optioner. En webbsida kan bestå av 30–40 objekt och behöva hundratals paket. Det vi uppfattar som en webbsida kan egentligen bestå av flera webbsidor som hämtas på olika servrar. För att förbättra prestanda används dessutom mellanlagring på klienten men även på Internet.

Det är också värt att notera att många webbplatser lagras på många ställen. Om vi surfar till exempelvis microsoft.com, www.fbi.gov eller www.whitehouse.gov så hamnar vi oftast på en av många kopior av webbplatsen. Detta gör att prestanda förbättras eftersom kopian är lokaliserad nära oss, men det gör både namnuppslagning och HTTP mer komplicerad. Vidare kan klient och server förhandla om språkstöd etc så klienten hämtar olika objekt beroende på våra lokala inställningar.

Webben var från början tillståndslös, en sak som kan lösas med cookies. En cookie är i huvudsakligen en liten sträng som läggs på en bestämd plats på din dator. Varje webbplats skapar en egen sträng. En cookie är mer tillförlitlig än en IP-adress eftersom bland annat adressöversättning gör att fler användare kan uppträda som en IP-adress. Den här lilla strängen gör att en server kan komma ihåg att när du går till kassan så har du tidigare klickat på köp av två böcker. All denna logik bygger på vad servern gör och lagrar om din unika cookie.

På ett liknande sätt kan servern lagra vilka sökningar du brukar göra och sortera svaren efter vad du valt. Givetvis kan servern på detta sätt även styra vilken reklam den vill visa. Cookies är alltså



Exempel på inställningar för mellanlagring (cache) och cookies. Längst till höger exempel på cookiefiler.

inte en speciellt avancerad funktion och röjer inte så många hemligheter i sig. Men genom att granska vilka cookies en användare har på datorn kan man se vilka webbplatser han besökt. Webbläsare har historiskt också haft problem så att servrar har kunnat läsa varandras cookies, vilket inte är meningen och en oetisk användning. Cookies fick därför snabbt oförtjänt dåligt rykte. I alla moderna webbläsare kan man blockera cookies, men vissa webbplatser fungerar inte utan cookies så de får då användas tillfälligt.

Cookies används även för att räkna antal unika besökare på en webbplats. Flera webbplatser lever på annonser och man behöver skaffa sig en uppfattning om antal unika besökare.

Hur påverkas program av IPv6?

Att byta ut ett nätnivå-protokoll som IP har visat sig vara svårt. Men det ställer även till det för applikationerna. Vi kan se detta på hur det triviala programmet ping fungerar nu när IPv6-stöd är på väg att införas. I Windows Vista skriver du alltid ”ping nodnamn” oavsett om du ska pinga en IPv4 eller en IPv6-adress. Windows Vista ställer DNS-frågor efter både IPv4- och IPv6-adresser. Om den får träff på en IPv6-adress prioriterar den IPv6. (Du kan tvinga ping att använda version 4 eller 6 med växlar till kommandot.) Samma kommando i dagens Linuxvarianter fungerar ibland inte alls med IPv6, man måste skriva ”ping -6” eller använda programmet och kommandot ping6. Och även om man gjort det kanske det inte fungerar, då ligger skillnaden kanske i huruvida den DNS som innehåller rätt information pratar IPv4, IPv6 eller både och.

Ping är ett enkelt program. För mer komplexa program är läget inte bättre. Vissa program fungerar inte alls med IPv6, även om de principiellt bara borde få rätt information från DNS. Många program blir direkt motsträviga om man försöker få dem att fungera med IPv6 och förklarar glatt att adresser som 2001:234:5678:999:204:76ff:fec2:c193 inte är ett ”giltigt namn!” eller en giltig sökväg. Program som innehåller IPv4-adresser i kod eller konfigurationer får

självklart stora problem. För att underlätta övergången till IPv6 finns verktyg för utvecklare som går igenom kod och varnar för potentiella svårigheter och problem.

Säkerhet på applikationsnivå

De program vi gått igenom i det här kapitlet har alla i någon mening säkerhetsproblem. De flesta äldre program skickar även intern kommunikation och inloggningssekvenser i klartext. Detta gör debuggning och felsökning enkel men det gör trafiken känslig för avlyssning.

De flesta program finns i förbättrade varianter med bättre säkerhet inklusive autentisering och kryptering. För HTTP finns en förbättring kallad HTTPS, som bygger på SSL, Secure Socket Layer.

SSL, Secure Socket Layer

När du använder en Internetbaserad banktjänst eller köper något via ett kreditkort på Internet finns ingen delad hemlighet eller lösenord som du och banken eller kortinlösaren kommit överens om. Den pinkod som används (fyra till sex siffror) är alldeles för svag för att skydda informationen, en dator testar sig igenom sex siffror på bråkdelen av en sekund. Men ändå är allt som skickas mellan din dator och webbplatsen skyddat, och det är det lilla hänglåset som syns i webbläsaren som ger en fingervisning om hur det går till.

Secure Socket Layer, SSL, är ett bra exempel på en modern metod för att kryptera information som ska överföras. Den är beprövad, den är väldigt spridd, den finns i ett par varianter och den använder både symmetrisk och asymmetrisk kryptering. Och som flera andra metoder har den haft problem i de första implementeringarna som kom.

Idag använder vi SSL version 3 eller 3.1. Historiska problem med SSL är inte alls så illa som det låter utan kanske till och med en styrka. Moderna metoder för säkerhet bygger sällan på att någon enstaka person sitter inlåst på en kammare och uppfinner en metod.

Det är istället en grupp som utvecklar en metod, i fallet SSL var det företaget Netscape som utvecklade en metod för säker informationsöverföring mellan webbläsare och webbserver. Uppfinnarna lanserar sedan metoden och vips kommer ett antal andra arbetsgrupper att försöka hitta brister. Netscape fick kritik för brister i sina första releaser och kom i mitten av 1990-talet med tre releaser i tät följd (SSL version 1 till 3). Resultatet har blivit en metod som är testad och beprövad, och inte mycket har behövt ändras sedan dess. IETF tog några år senare fram en variant på SSL, där man redde ut ett par oklarheter, som kallas för TLS (Transport Layer Security).

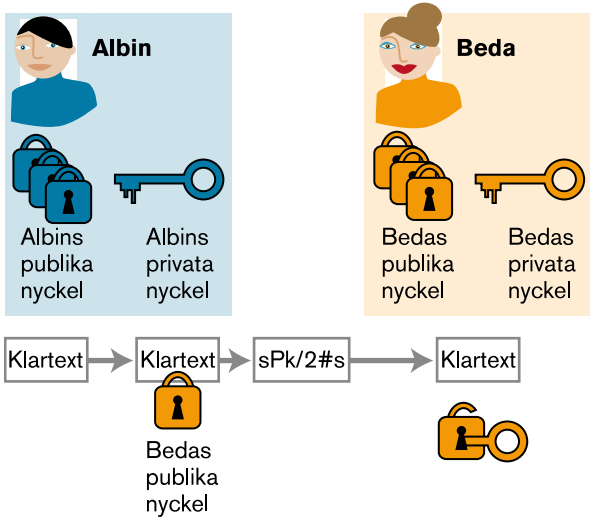
SSL bygger på att vi krypterar informationen mellan nivå fyra och fem enligt OSI:s referensmodell. Det gör att flera protokoll som ligger ovanpå TCP eller UDP ganska enkelt kan anpassas och då förekomma i en krypterad variant. HTTP finns i varianten HTTPS. NNTP (Network News Transfer Protocol) finns i varianten NNTPS. POP3, FTP och Telnet finns i varianter som bygger på SSL. Eftersom vi krypterar på en högre nivå än nivå tre kommer felmeddelanden (ICMP) inte krypteras, inte heller DNS- eller routinginformation. Men vi skyddar data som överförs mellan program och ofta är det just det vi är ute efter.

För att förstå SSL behöver vi kunna skilja på symmetrisk och asymmetrisk kryptering. Fram till 1970-talet fanns bara symmetrisk kryptering. Symmetrisk kryptering bygger på att sändare och mottagare använder samma nyckel vid kryptering och dekryptering. För att förstå principen kan vi tänka oss att nyckeln består av 40 bitar, det vill säga tio hexadecimala siffror. Ett exempel skulle kunna vara strängen 1846B3CF71. Om ett meddelande är digitaliserat kan vi uppfatta även det som siffror (eller ett väldigt stort tal) och nu bestämmer vi att vår metod för kryptering består av att vi multiplicerar vår klartext med nyckeln. Den metod som vår mottagare använder är att dividera med nyckeln.

Det finns ett par problem med den här modellen. Dels är metoden sårbar. Om sändare och mottagare kommer överens om en dålig nyckel som till exempel 1111111111 så kommer kryptot bli enkelt att knäcka och även det krypterade meddelandet kommer röja delar av klartexten. Dels kanske den som vill knäcka metoden kan få av-

sändaren att skicka ett meddelande bestående av klartexten 1 och då visas nyckeln rakt av. Men huvudproblemet med all symmetrisk kryptering handlar om hur avsändare och mottagare ska byta nycklar. Det sker i och för sig utanför modellen men är ändå ett stort problem. Speciellt med långa nycklar så skriver vi ner dem eller sparar dem på ett lagringmedium och då kan obehöriga komma åt dem. Trots denna brist är symmetriska krypton baserade på en delad hemlighet (*shared key*) väldigt vanliga. Några exempel på symmetriska krypton är DES, 3DES, IDEA, Blowfish, AES.

Asymmetriska metoder använder en nyckel för kryptering och en annan nyckel för dekryptering. De har den förvånansvärda egenskapen att vi kan börja med att kryptera med den ena nyckeln (den publika, kallas även offentlig nyckel) och sedan dekryptera med den andra (den privata) och på så sätt få tillbaka klartexten. Men vi kunde lika gärna börjat med att kryptera med den privata nyckeln och sedan dekrypterat med den publika. Det kan verka lite som magi och matematiken som ligger bakom är avancerad och kräver datorer. Metoderna uppfanns på 1970-talet, och den mest spridda kallas RSA efter uppfinnarnas initialer.



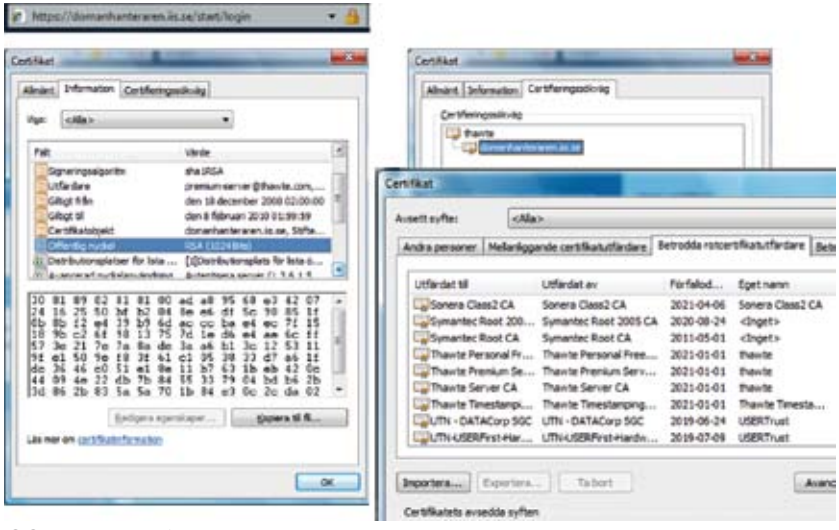
Albin och Beda med sina privata och publika nycklar. Den privata nyckeln håller man för sig själv. Den publika nyckeln kan man kopiera och sprida. På bilden visas den publika nyckeln symboliskt med ett hänglås. I verkligheten är både den privata och publika nyckeln långa teckensträngar.

Varje användare i ett system för asymmetrisk kryptering har alltså två nycklar, den privata som man ska hålla för sig själv och den publika som kan spridas fritt. Vill man kryptera ett meddelande till en mottagare så hämtar man mottagarens publika nyckel och krypterar med den, då blir det bara mottagaren som kan dekryptera meddelandet.

En avsändare kan även kryptera ett meddelande med sin egen privata nyckel innan det skickas iväg. Mottagaren kan bara dekryptera meddelandet genom att hämta avsändarens publika nycket och kan därmed vara säker på att det verkligen kommer från rätt avsändare. Vi får på det här sättet en metod för digitala signaturer.

Observera att Albin inte behöver träffa Beda

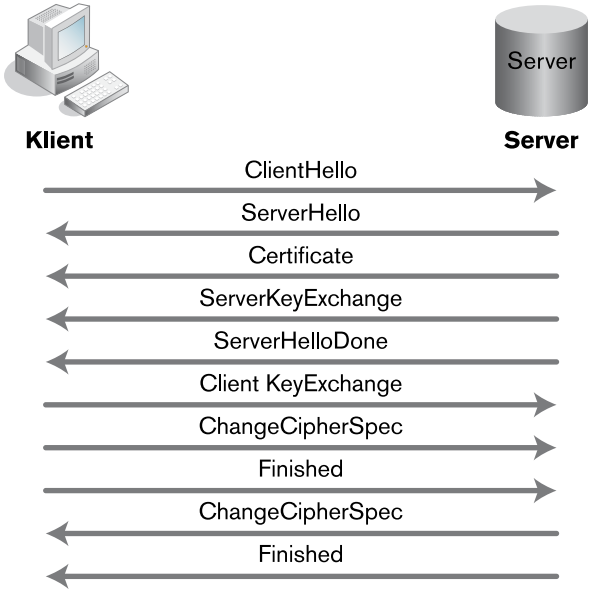
för att få tag på hennes publika nyckel. Albin kan lika gärna fråga exempelvis David om han har tillgång till Bedas publika nyckel. Och om Albin litar på David, och David är ärlig, så fungerar modellen. En annan möjlighet är att gå till en server som hanterar nycklar eller något som skulle kunna likna en myndighet som tillhandahåller medborgarnas publika nycklar. Men om det nu finns en illasinnad användare vid namn Crydolf så kan ju han ha bytt ut Bedas nycklar för att komma åt den information som bara är avsedd för Bedas ögon. För att systemet med publika nycklar ska fungera behövs därför en infrastruktur vi kan lita på, den beteckning som vanligen används är Public Key Infrastructure eller PKI. PKI bygger på att vi litar på en eller flera så kallade rot-CA (*Certificate Authority* eller certifikatutfärdare). Dessa har till uppgift att garantera att en publik nyckel hör ihop med en viss användaridentitet eller en viss webbplats, och att ingen Crydolf har bytt ut dem.



SSL och certifikat. Klickar du på hänglåset i webbläsaren så visas certifikatet och där finns serverns publika (offentliga) nyckel. Under certifieringssökväg ser man att thawte står som garant för att detta är den nyckel som tillhör domännamnet "domanhanteraren.iis.se". Internet Explorer har som förinställt värde att lita på Thawte bland många andra.

Ett certifikat är ett elektroniskt dokument som består av en publik nyckel, giltighetsdatum etc. samt en utfärdare. Litar vi på certifikatutfärdaren ska vi kunna låta denne signera att vi fått rätt information i vårt elektroniska dokument.

Dags att återvända till SSL. I likhet med många andra moderna säkerhetsmetoder kombinerar SSL det bästa av två världar. Vi använder symmetrisk kryptering under den största delen av sessionen. Symmetrisk kryptering går fort och belastar inte datorns processor så mycket. Men vi använder asymmetrisk kryptering och serverns publika nyckel för att kunna byta nyckel på ett säkert sätt.



Handskakningen i SLL mellan klient och server.

Klienten börjar handskakningen. I huvudsak meddelar den vilka versioner av SSL den har stöd för, vilka krypteringsmetoder den stöder samt ett slumptal som ska användas när krypteringen startar. Servern svarar med i princip samma information förutom slumptalet. Servern skickar sedan över sitt certifikat och övriga certifikat hela sökvägen upp till en rot-CA. Det är sedan klientens ansvar att verifiera certifikatets giltighet. I nästa skede överför servern sin publika nyckel och meddelar att den är klar. Klienten skickar sedan över den gemensamma nyckel som ska användas för symmetrisk kryptering under sessionen, och denna nyckel krypteras med serverns publika nyckel. Bara servern kan därmed dekryptera och komma åt sessionsnyckeln. Klienten skickar sedan över vilken metod som ska användas för den symmetriska krypteringen, till exempel DES och SHA-1. Servern avslutar med att kvittera metoden.

Genom SSL får vi kontroll på att vi pratar med rätt server (serverautentisering) och vi får ett sätt att skydda informationsöverföringen. Vill vi kontrollera klienten eller användaren kan det göras via ett certifikat det också (det finns ett tillägg till SSL med klientautentisering). Men det vanligaste är att använda

lösenord eller dosor för att generera engångslösenord. Till skillnad från HTTP där lösenord överförs helt oskyddat kan lösenord överföras på ett säkert sätt via HTTPS tack vare sessionsnyckeln.

Referenser

RFC 959	FTP
RFC 1630 och 1738	Behandlar URL
RFC 2854	HTML
RFC 1945	HTTP 1.0
RFC 2616	Behandlar HTTP version 1.1
www consortium	www.w3c.org
HTTP Essential	Stephen Thomas, 2001 (Enklare att läsa än RFC 2616)
TCP/IP Handboken	Gunnar Gunnarsson, 1999 Trots namnet handlar boken mest om applikationer.

Domain Name System – DNS

- Det här kapitlet behandlar DNS översiktligt. Det är ett utdrag ur ett mer omfattande kapitel i boken. Hur vi klarade oss innan DNS behandlas. Namnstrukturer och toppdomäner gås igenom. Hur DNS-frågor görs och hur svaren mellanlagras och behandlas.
- Kapitlet är tänkt att kunna läsas fristående men fungerar bäst om du vet hur TCP/IP fungerar.

DNS är ett protokoll för att i ett globalt nätverk av DNS-servers kunna ställa frågor och få svar. Om den aktuella servern inte känner till svaret så kan den svara med en hänvisning till en annan server istället. Program för DNS och protokollet har funnits i tjugo år och fungerar enligt samma principer. Två saker krånglar till det om du vill förstå DNS:

- användningen och begreppen har ändrats med tiden. Eftersom DNS fungerar väl globalt så finns det grupper som vill lägga in fler och fler funktioner i strukturen.
- I Windows används också begreppet domäner. Från början hade Windows-domäner ingenting med DNS att göra, men med Windows Server 2000 och senare 2003 används DNS även för detta. Tanken är att förenkla men eftersom de flesta organisationer inte vill visa sin interna struktur externt så blir det i alla fall komplext. (Tanken med DNS var från början att alla DNS:er ska kunna fråga varandra.)

I det här kapitlet utgår vi från dagens begrepp och hur grundfunktionerna i DNS används. DNS kan även användas för lastdelning och det kan användas för dynamiska uppdateringar då en nod byter IP-adress (Dynamic DNS enligt RFC 2136). Det finns även ett tillägg DNSSEC med vars hjälp man kan upptäcka förfalskade DNS-data. Dessa funktioner behandlas dock inte i detta kapitel.

Så här långt har guiden mest handlat om IP-adresser. Men de flesta användare använder inte IP-adresser utan kommer bara i

Nodnamn (host name) och domännamn (domain name) är två olika saker. FQDN är ett namn som fungerar i DNS-strukturen och kan närmast översättas med komplett domännamn.

kontakt med nodnamn. I den här guiden används benämningen nodnamn, på engelska används ofta begreppen ”host name”. I DNS används begreppet domännamn (*domain name*) vilket även omfattar namn som www.bolag.se. Ett mer exakt begrepp är det engelska FQDN, Fully Qualified Domain Name. Jag har använt begreppet komplett domännamn eller FQDN nedan för att särskilja det från nodnamn (det är inte ovanligt att en dator har FQDN som www.bolag.se medan nodnamnet speglar maskinens uppbyggnad som 2GXeonDuo).

Hosts-filen

Internet och TCP/IP har en gång fungerat utan DNS men i praktiken kan vi inte vara utan det idag. I stort sett alla TCP/IP-implementationer har kvar den tidiga metoden i form av en hosts-fil. I Windows XP nås den under C:\WINDOWS\system32\drivers\etc. I filen finns i princip två kolumner, den högra är nyckeln till det vi letar med. När vi får träff så kontrolleras vad som står till vänster. Om vi till exempel skriver in sekvensen ”x.acme.com” i en webbläsare så fungerar i stort sett alla program så att de använder proceduren ”gethostbyname”. Och från hosts-filen kan vi då erhålla resultatet 38.25.63.10.

```
102.54.94.97     rhino.acme.com          # source server
38.25.63.10      x.acme.com              # x client host

127.0.0.1        localhost
```

Testa själv

Du kan enkelt testa hur hosts-filen fungerar. I de flesta Linuxvarianter kan man välja om DNS ska användas i första hand eller om hosts-filen ska användas. I Windows används hosts-filen till att uppdatera den lokala DNS-cachen på klienten.

Ta reda på en IP-adress som används och styr ett komplett domännamn som www.kth.se till denna IP-adress via ett tillägg till hosts-

filen. Mata sedan in domännamnet i en webbläsare. Du kommer att styras enligt hosts-filen. Även efter att denna ändring tas bort ur hosts-filen ligger denna information kvar ett tag, detta beror på att man mellanlagrar DNS-information (alltså även i en vanlig arbetsstations resolver).

Namnstrukturer

DNS inför två stora förändringar jämfört med hosts-filer. För det första blir det svårt att genomföra globala förändringar om varje dator har en lokal fil. Under Internets tidiga år gjorde man så att det fanns en masterfil som användare kunde hämta regelbundet. Men den här modellen håller inte om vi kräver att ändringar ska slå igenom fort och kunna hantera miljontals användare. DNS hämtar istället bara den information som behövs, och den hämtas när den behövs. En nyckel till att hantera informationen effektivt är sedan att svar mellanlagras (cachas).

DNS hämtar bara den information som behövs och informationen mellanlagras.

DNS införde dessutom en namnstruktur. Tidigare hade varje nod ofta ett enkelt namn som rhino eller alma. Men om det nu fanns två servers med samma namn. Vem är mest alma om två universitet hade samma datornamn? Och hur ska vi hantera alla maskiner som har namn som anknyter till funktioner, typ mailserver, postman, webserver och nameserver?

Ett komplett domännamn innehåller normalt minst tre nivåer. Två exempel är www.firma.se och mail.bolag.com. Detta kallas som sagt ett FQDN och utläses i det första fallet som noden ”www” i domänen ”firma.se”. Ett komplett domännamn går hela vägen från den enstaka noden ända upp till roten (ibland understryker man detta genom att lägga till en extra punkt som refererar till roten, till exempel ”www.firma.se.”). Ett komplett domännamn går ofta att översätta till en IP-adress, men det är ofta intressant att ställa andra frågor än bara IP-adresser. Vi kan till exempel fråga vilken nod som är ansvarig för domänen, vi kan fråga om det finns en e-post server med mera.

Toppdomäner

Nationella toppdomäner betecknas ofta ccTLD's: Country Coded Top Level Domains.
De generiska betecknas gTLD's: Generic Top Level Domains.

Sedan länge har toppdomänerna funnits i två varianter. Nationella toppdomäner som se, no och de samt generiska som com och net. De generiska utökades under 2000 till 2002 och omfattar idag:

Toppdomän	Tillkom	Anm
.aero	2002	Flygfartsindustri
.arpa		För infrastruktur och fanns i princip innan DNS. Används bland annat för baklänges-uppslagning.
.biz	2001	Jämför med .com
.com	1985	Kommersiella bolag (den mest använda toppdomänen)
.coop	2002	
.edu	1985	Utbildningsväsende
.gov	1985	Amerikansk statsförvaltning (governmental)
.info	2001	
.int	1988. Stängdes 2000.	Internationella organisationer, bland annat drivet av önskemål från NATO.
.mil	1985	Amerikansk militär
.museum	2001	För museer
.name	2002	Individer
.net	1985	"Network support Centers". Tillsammans med "com" en av de toppdomäner som det varit lättast att registrera namn på.
.org	1985	Övriga organisationer. Tillsammans med "com" en av de toppdomäner som det varit lättast att registrera namn på.
.pro	2004	Professionals. Am begrepp som bland annat omfattar läkare och advokater.

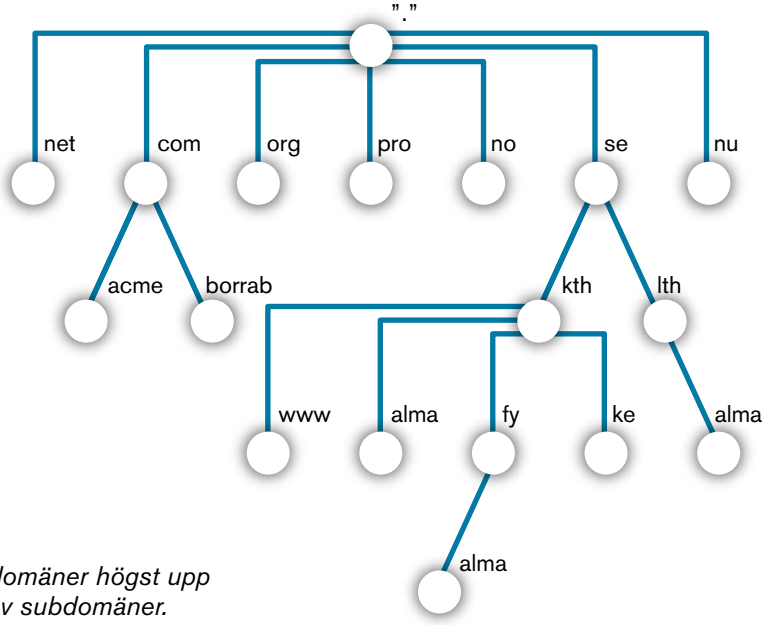
Toppdomännamn.

Under 2007 tillkom fler nya domäner som cat, jobs, mobi, travel. Andra förslag var mail och tel.

Generiska toppdomäner styrs av ICANN. Hur de används har styrts av riktlinjer för respektive domän. De nationella subdomänerna har fungerat olika i olika länder. I England har de till exempel ofta an-

vänt subdomäner som co och ac under sin toppdomän uk. Amerikanska bolag använde sällan domänen us utan de generiska. I Sverige var man under 1990-talet väldigt restriktiv men utdelning under ”.se” och krävde registrerade varumärken, föreningar eller aktiebolag. Enskilda firmor registrerades under länsbokstäver (detta harmonierar med hur bolagsnamnen hanteras). Från 2003 gäller dock principen först till kvarn under .se-domänen och konflikter löses i efterhand.

Eftersom det under 1990-talet var enklare att registrera com, net och org-domäner så blev de ofta populära och till och med svenska statliga utredningar dök upp under com-domänen. Under 1990-talet började även namnpirater att registrera organisatoriska domäner eller vanliga felslagningar av populära domäner för att sälja dem vidare. Den praktiska lösningen på detta blev att sökmotorer används oftare eftersom vi inte vet om vi ska söka på ”bolag.com” eller ”bolag.net” eller ”bolagab.se”. Och i princip kan den som administrerar en domän bara styra hur subdomäner till just den domänen delas ut.



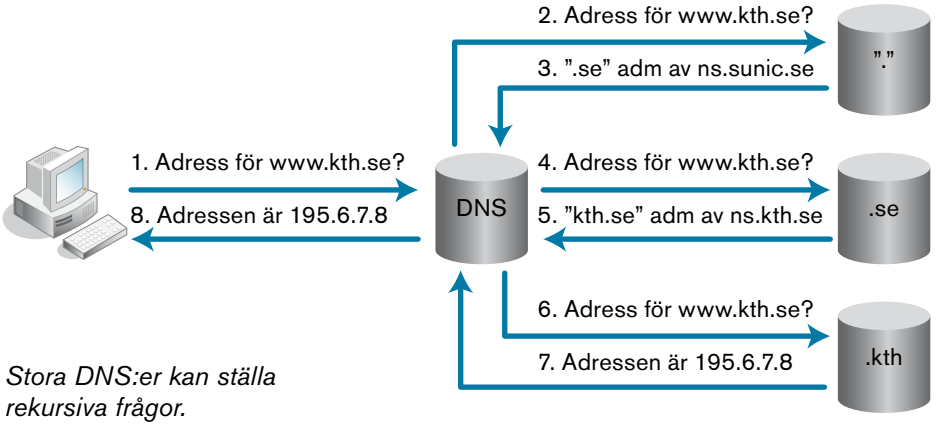
Toppdomäner högst upp följs av subdomäner.

Med toppdomäner och en namnstruktur får vi ordning på namnen så att de blir unika. Vi kan skilja på alma.kth.se och alma.lth.se. Högst upp (det vi kallar roten ologiskt nog) har vi alla namn. De så kallade rotservrarna känner sedan till de knappt 300 toppdomänerna och vilka namnservrar som har vidare information. Poängen är sedan att de hänvisar till dessa servrar för mer information. På samma sätt kan ett par maskiner vara ansvariga för se-domänen, men de innehåller främst information om varje subdomän som registreras och vilken namnserver som har informationen. Om vi granskar domänen kth.se så ser vi att det finns två noder vid namn www och alma. Däremot är fy och troligen även ke subdomäner som delegeras vidare. I DNS-strukturen hanteras enkla nodnamn och domäner på samma sätt. Från namnen i sig kan vi inte se att www är en nod och ke är en subdomän. (Det finns dock ett antal konventioner som till exempel www för en server som svarar på port 80 och ns för den nod som hanterar domäner.) Därför skulle vi få meddelandet ”Non Existing Domain” förkortat som NXDOMAIN om vi frågade efter det felaktiga ww.kth.se.

Resolvers

En vanlig arbetsstation eller PC innehåller en liten DNS-resolver. Applikationer skickar som sagt en uppkopplingsbegäran med kommandot gethostbyname. DNS-resolvern på din PC löser detta men den tar hjälp av en fullödig DNS. Denna skiljer sig från din PC:s enkla på så sätt att den kan ställa rekursiva frågor. Ur PC:ns synpunkt ställs frågan till en DNS (1) och strax efter kommer svaret (8).

En DNS har en tabell med IP-adresser till de 13 rotservrar som finns globalt. Frågan skickas till en av dessa. Rotservern svarar (3) med att den är just rotserver, men den här frågan avseende se-domänen är delegerad till en (eller fler) servrar. Till DNS-protokollet hör också möjligheten att lägga till extra information för att minska antalet frågor, så ett fält ”additional information” innehåller information om IP-adressen till ns.sunic.se. Vår DNS skickar nu



Stora DNS:er kan ställa rekursiva frågor.

frågan till ns.sunic.se (4) som svarar att information om kth.se har delegerats vidare. Eventuellt sker liknande delegering en eller två gånger till men tillslut hamnar vi hos en DNS som är ansvarig för denna domän. Denna svarar med IP-adressen för www.kth.se (7). Vår DNS kan nu leverera svaret till vår PC-klient (8). Hemligheten med DNS är alltså att servrarna hela tiden delegerar frågan till någon som vet. Gott ledarskap alltså.

Svaren från en DNS innehåller även tidsvärden (Time To Live). Detta gör att svaren kan mellanlagras och här ligger mycket av effektiviteten. Nästa gång vår DNS behöver fråga om något som avser hela se-domänen så vet den att den kan skicka frågan till ns.sunic.se. Rotservern behöver inte belastas igen. Om en annan klient skickar samma fråga till vår DNS så kan den direkt svara med IP-adressen. Hur länge svaren ska sparas kan variera. Typiska värden kan vara 8 till 24 timmar. Kortare och längre värden förekommer, till exempel brukar rotservrarna meddela att svar vad gäller toppdomäner kan mellanlagras i sju dygn.

Eftersom svar mellanlagras tar det tid innan ändringar slår igenom. Vill man byta IP-adress på en maskin bör man först ställa ner tidsvärdet så den lagras kort tid och sedan genomföra förändringen. Har man sin DNS utlagd till en operatör tar det även tid innan de genomför förändringen, speciellt om man ska byta vilken server som

är ansvarig för en domän, en så kallad domain transfer. Felaktiga ändringar får stort genomslag och säkerhetsmässigt vill operatörerna vara säkra på att den som begär flytt och ändringar är behörig. Ändringar kompliceras också ofta av att fler servrar blandas in.

Eftersom DNS är en viktig del av Internet behövs redundans. Om du försökt registrera en ny domän så har du säkert fått frågan om vilka maskiner som ansvarar för domänen. Det räcker inte med att ange en adress utan två. Förr användes begreppen primary och secondary DNS dessa har idag ersatts av master och slave. En master-DNS kan ha flera slave-DNS. Det är master-DNS som innehåller informationen i original och slave-DNS kollar regelbundet efter förändringar och hämtar vid behov uppdaterad information. Överliggande domän har normalt vetskap om två eller fler servers så DNS-frågorna kan besvaras utan att vara beroende av enstaka servrar.

Fråga och svar

Nedan visas en inspelning av ett paket till och från en DNS-server. UDP och port 53 används som väntat.

För att kunna matcha svar med fråga används en identitet (2 byte). Sedan följer 16 bitar som används för flaggor, här kan vi se att det är en standardfråga och DNS-servern förväntas använda en rekursiv sökning. I övrigt består en DNS-fråga av fyra delar: *question* (fråga), *answer* (svar), *authority* (behörighet) och *additional* (tillägg). Efter flaggorna följer information om huruvida dessa delar används och deras antal. I detta fall följer en fråga och den byggs upp. Frågan innehåller vår söknyckel samt två fält: typ och klass. Klassen är enkel då den i praktiken alltid är IN, det vill säga Internet. Typen av fråga varierar. Det finns cirka 50 olika posttyper (type) definierade varav några av de vanligaste är:

Typ	Förklaring	Anm
A	Host Address	IP-adress
AAAA	IPv6 Host Address	IPv6-adress
NS	Name Server	Namn på authoritative server
PTR	Pointer	För baklängesuppslagning
SOA	Start Of Authority	Administrativ information för en domän
CNAME	Canonical Name	Alias, används för att t ex styra www.bolag.se till 2GXeon.bolag.se
MX	Mail eX-changer	Används för e-post. Namnservrar frågar om det finns en MX-server för en viss domän.

Några vanliga posttyper.

Paket 158
Ethernet II, Src: Fujitsu_c3:2f:98, Dst: 3comEuro_9f:4a:fa
Internet Protocol, Src: 192.168.3.245, Dst: 192.168.3.1
User Datagram Protocol, Src Port: 1633 (1633), Dst Port: domain (53)
  Source port: 1633 (1633)
  Destination port: domain (53)
  Length: 46
  Checksum: 0x2217 [correct]
Domain Name System (query)
  Transaction ID: 0x0006
  Flags: 0x0100 (Standard query)
    0... .... = Response: **Message is a query**
    .000 0... .... = Opcode: **Standard query** (0)
    .... ..0. .... = Truncated: Message is not truncated
    ....1 .... = Recursion desired: Do query recursively
    .... .0.. .... = Z: reserved (0)
    ....0 .... = Non-authenticated data OK
  Questions: 1

```
Answer RRs: 0
 Authority RRs: 0
 Additional RRs: 0
 Queries
      Name: www.mittbolag.se
      Type: A (Host address)
      Class: IN (0x0001)
```

I svaret ser vi inte bara själva IP-adressen utan också information om hur länge svaret kan mellanlagras (TTL Time To Live). Vi ser också att den server som svarat inte är ansvarig (authority) för domänen. Servern svarar också med att den tillåter rekursiva frågor.

```
Paket 159
Ethernet II, Src: 3comEuro_9f:4a:fa, Dst: Fujitsu_c3:2f:98
Internet Protocol, Src: 192.168.3.1, Dst: 192.168.3.245
User Datagram Protocol, Src Port: domain (53), Dst Port: 1633 (1633)
Domain Name System (response)
   [Request In: 148]
   [Time: 0.029266000 seconds]
   Transaction ID: 0x0006
   Flags: 0x8180 (Standard query response, No error)
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative:
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 1... .... = Recursion available:
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated
      .... .... .... 0000 = Reply code: No error (0)
   Questions: 1
   Answer RRs: 1
```

```
Authority RRs: 0
Additional RRs: 0
Queries
   www.mittbolag.se: type A, class IN
      Name: www.mittbolag.se
      Type: A (Host address)
      Class: IN (0x0001)
Answers
   www.mittbolag.se: type A, class IN, addr 130.239.8.25
      Name: www.mittbolag.se
      Type: A (Host address)
      Class: IN (0x0001)
      Time to live: 9 hours, 21 minutes, 58 seconds
      Data length: 4
      Addr: 130.239.8.25
```

Om man använder kommandot ”dig” i Unix för att felsöka i DNS får man svar som stämmer väl med hur DNS-frågor och svar ser ut. Om allt går bra erhålls svaret NOERROR, vid problem får man ofta meddelandet NXDOMAIN samt information om vilken DNS som svarat detta. På detta sätt kan man se hur långt namnupplösningen fungerade.

Baklängesuppslagning

DNS-strukturen kan användas åt andra hållet. Vi kan allstå få svar på frågan ”Om vi har en IP-adress vad motsvarar det för domännamn?” Baklängesuppslagning görs av flera applikationer i samband med loggning, felsökning och verifiering. DNS bygger på att den mest signifikanta delen finns till höger (med början i toppdomänen). IP-adresser har den mest signifikanta delen till vänster så man löser upp adresser från vänster till höger. En konvention är att man har reserverat domänen ”in-addr.arpa” för baklängesupp-

slagning. Normalt tar DNS och även verktyg som nslookup och dig hand om detta själv. Vill vi veta domännamnet för exempelvis 192.3.4.5 så kommer själva frågan ha formen ”5.4.3.192.in-addr.arpa” men det slipper vi som användare oftast tänka på.

Stöd för IPv6 i DNS

DNS är en nyckelfunktion för införandet av IPv6. Posttypen AAAA för IPv6-adresser kom med tidigt i arbetet med version 6. Även baklängesuppslagning togs fram ganska tidigt, idag sker det via domänen ip6.arpa som är reserverad för baklängesuppslagning (PTR-poster) inom IPv6. Däremot skedde DNS-frågor i många år fortfarande via IPv4. (Det var samma sak med routingprotokoll först, de kunde hantera IPv6-information men data utbyttes via IPv4.) Med Windows Vista och ett par Linuxvarianter kan din DNS-resolver idag ställa DNS-frågor över IPv6.

DNS-funktionen blir som sagt viktig med IPv6. Ett skäl är att IPv6-adresser kan vara långa och svåra att komma ihåg, vi behöver DNS för att kunna arbeta med namn istället. Ett annat skäl är att DNS kan styra vilken adress och vilket protokoll som kommer att användas. Om noden www.mittbolag.se har både en A- samt en AAAA-post men DNS bara frågar efter A-posten kommer applikationen att styras mot IPv4. Om DNS frågar efter bägge posterna kan applikationen välja, men då måste informationen hanteras mellan resolver och applikation och inte bara använda en enkel förfrågan typ GetHostByName.

Om DNS fortsätter att bara fråga efter A-poster kommer införandet av IPv6 att fördröjas. Det är alltså viktigt för framtiden att DNS kan hantera IPv6-förfrågningar.

Säkerhet och DNS

DNS är en vital del av Internet och strukturen har utsatts för flera attacker, främst mot rotservrarna. Hittills har angriparna inte lyckats

och Internet skulle inte sluta fungera om en rotserver slogs ut. (De flesta frågorna till rotservrarna är fel delvis beroende på manuella felstavningar.)

Men DNS är känsligt och med ett lite mindre perspektiv kan en angripare ställa till med mycket. I en dåligt skyddad DNS kan någon ändra informationen så att frågor efter till exempel www.internetbank.se styrs till fel ställe. På samma sätt kan ett virus ändra i hosts-filen. Dessa metoder har använts historiskt för angrepp. Ytterligare ett sätt är att bygga ett litet program som lyssnar efter DNS-frågor och sedan skickar ett snabbt felaktigt DNS-svar. Det spelar då ingen roll att rätt svar kommer senare, DNS-klienten tar det första svaret.

För att förbättra situationen med DNS så finns en förbättrad variant som heter DNSsec. I DNSsec används digitala signaturer för att verifiera svaren.

Information från DNS-servrar har också kunnat användas vid attacker. Angripare hämtar information från en DNS och ser vilka servrar som har intressanta namn och får direkt information om dess IP-adress. Den mest använda programvaran för DNS-servrar, BIND, har historiskt sett haft flera brister som har rättats till.

Referenser

RFC 1034 och 1035	DNS
www.iis.se	Information .se-domänen Här finns även viss information om DNSsec.
www.isc.org	ISC utvecklar programmet BIND, en populär DNS-server. Manualen till BIND ger en bra bild av modern DNS.

Adressöversättning

- Det här kapitlet behandlar adressöversättning relativt djupt. Olika tekniker som NAPT och PAT tas upp. Likaså olika varianter av NAPT för att förklara problem som uppstår på grund av adressöversättning. Kapitlet är ett utdrag, konfigurationsexempel har utelämnats.
- Kapitlet fungerar bäst om du vet hur TCP/IP fungerar.

Att använda adressöversättning har blivit det vanligaste sättet att ansluta sig till Internet för privatpersoner och företag. Få bekymrar sig om att adressöversättning bryter mot ett av fundamenten inom IP: kommunikation ska ske ”end-to-end”. Detta gör att flera applikationsprotokoll utvecklats helt utan hänsyn till adressöversättning. Ur dessa utvecklares ögon slutar Internet vid den publika adressen. Om datorn eller gatewayen med en publik adress sedan skickar dataflödet vidare till en annan nod så sker detta inte på Internet längre.

Adressöversättning bryter mot principen att kommunicera ”end-to-end”.

Men adressöversättning har fördelar. Fyra miljarder adresser är en begränsning och de som delar ut adresser idag är restriktiva med utdelning av publika adresser, se kapitlet IP-nivån. Två andra skäl till att adressöversättning slagit igenom är säkerhet och administration. Med adressöversättning har din dator bytt adress på Internet och är inte en del av Internet. Alltså kan inte heller en inkräktare skicka paket hur som helst till din dator. Med publika adresser följer krav på dokumentation. Om ditt nät har vuxit ur de publika adresser du tilldelats kan det också bli svårt. Du tvingas byta adresser på 254 noder när den 255:e ska installeras. Problemet finns i princip även med privata adresser men väljer du adresser ur nätet 10.0.0.0 så har du ett par miljoner adresser att arbeta med.

Prova om du använder adressöversättning. Med kommandona ipconfig eller ifconfig kan du se den IP-adress som din dator har just nu. Men hur ser det ut på Internet? Surfa till www.ripe.net och se vilken IP-adress de ser på Internet. (Andra sajter som visar din externa IP-adress är till exempel whatsmyip.org eller www.ipv6.org.) Om du har en annan adress externt än internt så används adressöversättning.

Testa själv

Det finns flera webbsidor som visar vilken IP-adress som efterfrågat webbsidan. Ett exempel är www.ripe.net (beskuren ovan).

Adressöversättning består egentligen av flera delar:

- mekanismen för att byta adress och eventuellt även portnummer
- privata adresser
- översättning av IP-adresser på applikationsnivå.

Privata adresser

Adressöversättning har sina största förtjänster om det kombineras med privata adresser. Det finns flera privata adressblock, men de vanligaste är de som definierats i RFC 1918 ”Address Allocation For Private Internets”:

10.0.0.0 – 10.255.255.255
172.16.0.0 – 172.31.255.255
192.168.0.0 – 192.168.255.255

Dessa adressblock kallas även RFC 1918-adresser och svarta adresser. Tanken är att dessa adresser aldrig ska routas vidare ut på Internet. Dessa adressblock ska istället kunna återanvändas gång på gång. Ett företag kan ha tusentals noder, men utåt sett används bara en eller ett fåtal publika IP-adresser.

Adressöversättning fungerar även utan privata adresser.

Adressöversättning är fullt möjlig även utan privata adresser. Tekniskt sett kan en organisation använda en annan organisations

adresser internt bara de översätts när de skickas ut på Internet. Tekniken benämns ibland ”masquerading” och ligger nära spoofing. (Spoofing innebär att man medvetet byter IP-adresser. En nod kan störas ut med en massa paket, för att inte kunna spåra vem som skickat dessa byter angriparen avsändaradress.) Med RFC 1918 blev det lite mer ordning på detta och risken att adresser ska dyka upp på fel ställe minskade betydligt.

Med IPv6 försvinner privata adresser

Det finns funktioner motsvarande privata nät i IPv6, men de innehåller en förbättring. Privata IPv4-adresser ställer till med problem om de dyker upp i DNS eller om de råkar dyka upp på Internet. Tanken lär ha varit att Internetoperatörerna alltid skulle filtrera bort RFC 1918-adresser, men det blev svårare när operatörerna själva började använda dem internt i sina egna driftnät.

Så kallade Universal Local-Adresser (ULA) i IPv6 förbättrar möjligheten att hantera motsvarande privata nät. ULA-adresser ska inte routas på Internet eller förekomma i DNS, de är enkla att filtrera bort då de börjar med strängen FC00::/7. Nästa bit kan variera så även FD00::/8 är en giltig ULA-adress. Nästa 40 bitar sätts slumpvis (fram till /48). På det här sättet minskar risken för adresskrockar enormt jämfört med RFC 1918-adresser. ULA-adresser kan även kombineras med globala prefix (de sista 80 bitarna avseende subnät och interface är desamma).

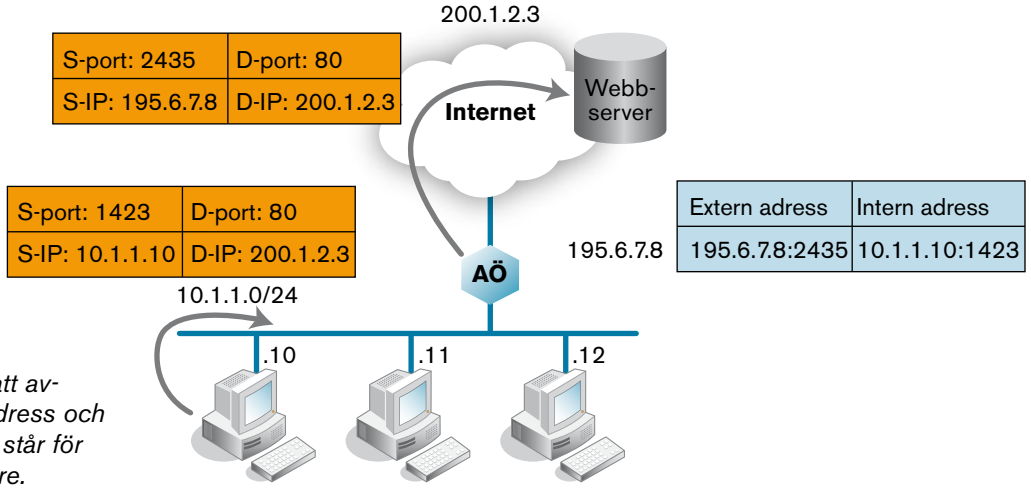
NAPT

Det finns flera olika tekniker för adressöversättning och flera olika akronymer som bland annat NAT, NAPT, NPAT och PAT. De används lite olika av olika tillverkare (till exempel Cisco) och beskrivs olika i olika publikationer. Även termen masquerading används för generell adressöversättning. På datakomjargong används adjektivet ”nattad adress” men exakt vilken teknik som avses är ofta oklart. I

detta kapitel används begreppen huvudsakligen i enlighet med RFC 2663 och 3022.

NAPT är den vanligaste formen av adressöversättning. Och den benämns ofta bara NAT.

NAPT (Network Adress Port Translation) är den vanligaste formen av adressöversättning. Den kallas tyvärr ofta bara NAT. Med NAPT kan flera hundra noder dela på en extern IP-adress. Att det fungerar är inte mycket konstigare än att man på en dator kan starta fler fönster (eller processer) och hämta ner filer från fler olika servrar. Med hjälp av portnummer kan man adressera rätt program eller process.



NAPT innebär att avsändarens IP-adress och portnr byts. AÖ står för adressöversättare.

Webbklienten 10.1.1.10 skickar ett paket till den externa webbservern 200.1.2.3. Webbklienten har adressöversättaren som default gateway. Adressöversättaren (AÖ) är konfigurerad för att byta adresser (en tabell där den externa adressen 195.6.7.8 ersätts med 10.1.1.10 och vice versa). Med NAPT kommer även portnummer i avsändarfältet att bytas. Adressöversättaren lägger även till en ny rad i sin tabell, den externa adressen 195.6.7.8:2435 ska mappas mot 10.1.1.10:1423.

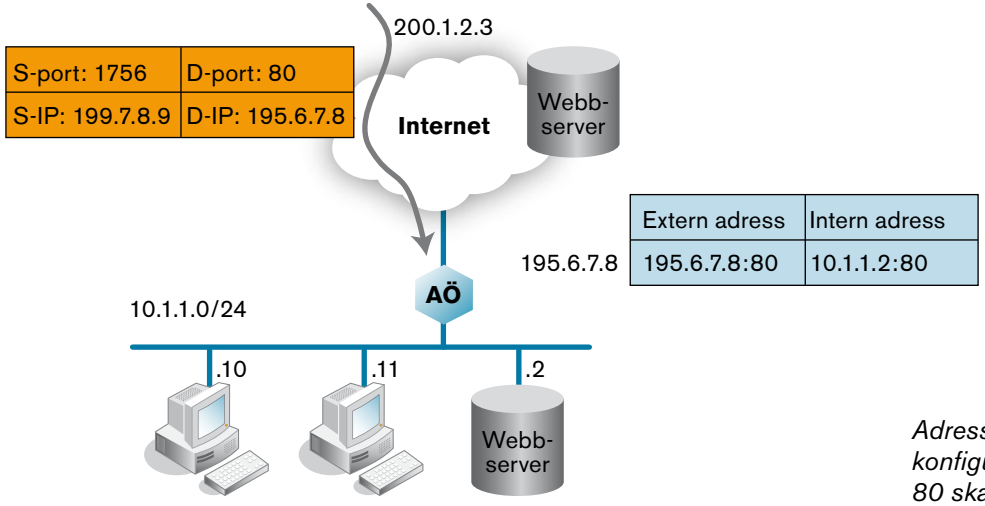
När paket kommer tillbaka från en server på Internet kan adressöversättaren utifrån portnumret avgöra vilken IP-adress och port på

den interna sidan som ska ha paketet. Varje gång ett paket passerar som använder aktuell rad i tabellen så uppdateras en timer. När timern räknat ner till noll raderas raden, timern arbetar i storleksordningen minuter. Adressöversättaren kan när TCP används även analysera sessionen och leta efter FIN och RST för att radera en sessionspost.

Webbservern uppfattar kommunikationen som en session initierad med adressöversättarens externa adress, det är denna adress som sparas i loggar och dylikt. En nod kan starta tusentals sessioner, det går inte att avgöra om de hanteras internt eller om de distribueras vidare på ett annat nätverk.

Den här typen av adressöversättning bygger på att datorn med en privat adress är den som initierar kommunikationen. Den fungerar alltså utmärkt när en webbklient ska hämta en webbsida från en server. Om kommunikationen istället skulle initieras från Internetsidan fungerar inte modellen. För servrar och peer-to-peer-nät behövs någonting annat.

Port Forwarding eller Port Address Translation – PAT



Adressöversättaren konfigureras med hur port 80 ska skickas vidare.

Port Forwarding är ett sätt att göra datorer tillgängliga på Internet för anrop. I det här fallet konfigurerar vi adressöversättaren så att varje gång den får en anropsbegäran på port 80 så skickar adressöversättaren frågan vidare till 10.1.1.2 på insidan. Konfiguration kan sättas upp port för port så till exempel anrop på port 23 skickas till en annan privat adress. Port Forwarding fungerar som NAPT med den skillnaden att adressöversättaren konfigureras med hur portar ska översättas.

Tekniken har begränsningar. Hur ska vi konfigurera om vi vill sätta upp två publika webbservrar? Bägge använder samma välkända portnummer 80. Den vanligaste lösningen är att vi använder en ny extern port. Till exempel 79 som vi styr om på insidan till exempelvis 10.1.1.4:80. En annan lösning är att adressöversättaren tillåter att vi har flera externa adresser som vi kan styra till olika interna adresser.

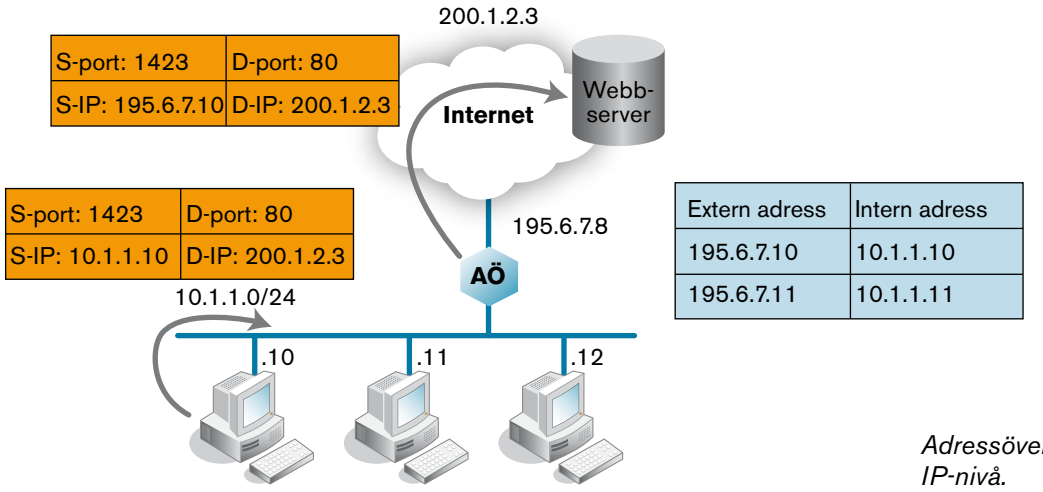
Begreppet PAT är inte helt konventionellt, man kan se tekniken som vanlig NAPT med den egenskapen att portöversättningen konfigureras för hand.

En del tillverkare använder begrepp som DMZ eller virtuell server då en dator på insidan får ta emot alla anrop från utsidan. Ur ett säkerhetsperspektiv sitter denna nod direktansluten mot Internet på utsidan och ska ha en säkerhetsnivå som motsvarar detta. Enligt min personliga åsikt bör man använda denna lösning med försiktighet. Noder som sitter på Internet logiskt sett bör även göra det fysiskt.

Network Adress Translation – NAT

En annan adressöversättningsteknik använder inte byte av portnummer utan arbetar bara på IP-nivå. Adressöversättaren innehåller en tabell som antingen innehåller en tabell hur varje IP-adress ska översättas statiskt i förhållandet ett till ett, eller en pool av adresser som dynamiskt ska användas externt.

Denna typ av adressöversättning kräver lite mer konfigurationsarbete men möjliggör att flera servers av samma typ kan nås utifrån.



Adressöversättning på IP-nivå.

Tekniken kan också råka ut för att poolen av externa IP-adresser tar slut.

Observera, som sagt, att begreppet NAT ofta används även på tekniken NAPT. (I RFC 2663 och 3022 benämns denna teknik som Basic NAT.)

Adressöversättning ger ett visst skydd

En nod som sitter bakom en adressöversättare är svår att nå från Internet. Om en inkräktare försöker skicka paket till IP-adress 10.1.1.10 så stoppas det av Interleverantörer på vägen. Försöker man istället skicka paket till den externa adressen så finns det normalt bara slumpartade klientportar upplagda och de har redan en session igång så de svarar inte på vanliga SYN-anrop.

När virusen Sasser och Blaster härjade som värst i början av 2000-talet hade man problem på stora nätverk med publika adresser. När nya maskiner driftsattes hann man inte uppdatera operativ-systemen innan maskinerna blev smittade. En lösning blev då att först sätta maskinerna på privata nät, uppdatera operativsystemen via Internet och sedan flytta dem till det publika nätet. Detta

förfarande är fortfarande ”god sed” vid driftsättning av nya maskiner.

Problem med adressöversättning

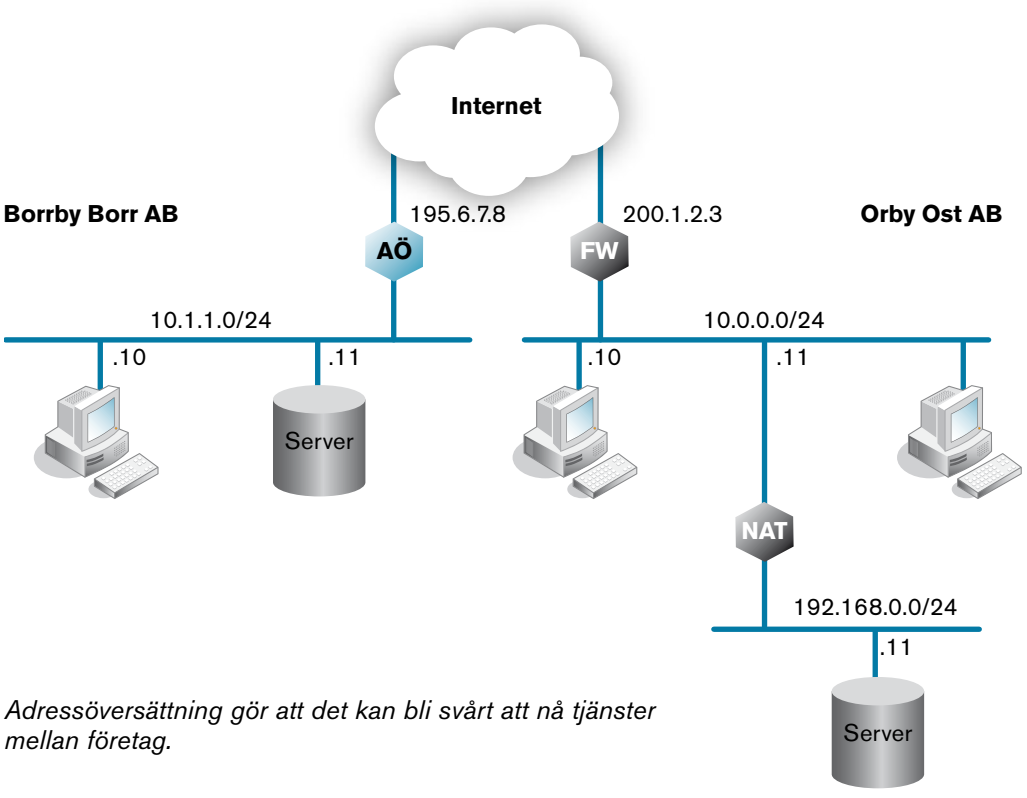
Adressöversättning slog igenom snabbt på 1990-talet trots problem och tveksamheter. Citat från RFC 1631 från år 1994:

> *NAT may be a good short term solution to the address depletion and scaling problems. This is because it requires very few changes and can be installed incrementally. NAT has several negative characteristics that make it inappropriate as a long term solution, and may make it inappropriate even as a short term solution. Only implementation and experimentation will determine its appropriateness.*

Det som var en förtjänst med Internet, snabb kommunikation nod till nod kan idag bli mycket komplicerat.

Borrby Borr och Orby Ost vill koppla ihop sina affärssystem. Med publika adresser hade det varit lätt, speciellt innan brandväggarnas tid. Nu krävs en del arbete. Vi måste ta reda på vilka portar respektive affärssystem använder och öppna för dem i brandväggarna samt adressöversättarna. I Orby Ost-nätet har man dessutom dubbel adressöversättning, tekniskt sett fungerar detta men vid felsökning blir det svårarbetat. Ofta har också företag outsourcat drift av brandväggen, varje ändring ska lämnas på en viss blankett tre arbetsdagar i förväg för att sedan först ifrågasättas av driftleverantören som till slut motvilligt konfigurerar om enheten, och sedan toppar med att göra det felaktigt. Det blir en ny blankett och flera arbetsdagar till. Att få ihop en sådan här koppling kan i praktiken ta flera veckor.

En annan liten avart som dykt upp är att operatörer börjat använda privata adresser. Vid en traceroute ser man resultat liknande:



Adressöversättning gör att det kan bli svårt att nå tjänster mellan företag.

```
C:\>tracert www.chalmers.se

Spårar väg till www.chalmers.se [129.16.221.8]
över högst 30 hopp:

1     1 ms     1 ms     1 ms  192.168.3.1
2    44 ms    31 ms    31 ms  10.244.128.193
3    21 ms    21 ms    21 ms  vlan6.sto21.se.isp.com
                              [195.54.116.245]
4    21 ms    21 ms    21 ms  sto2.se.isp.com [195.54.116.34]
```

Traceroute.

Operatörens användning av privata adresser medför dels att man riskerar krockar, operatörens kunder kanske tänkte använda nät 10.244.0.0/16 till något annat. Det medför också att inkommande översättning till servers (PAT) kräver medverkan från operatören.

Två andra problem med adressöversättning är så intressanta att de behandlas separat: applikationer och ”wrap around”. Trots de problem som finns med adressöversättning så är adressöversättning mycket vanligt idag. En stor del av den nätverkutrustning som säljs klarar inte ens publika adresser på det interna nätet – adressöversättningen är alltid påslagen. Adressöversättning skyddar också svaga operativsystem och dåligt uppdaterade maskiner, vi har mer eller mindre gjort oss beroende av den.

Med dagens användning av IPv4 har vi gjort oss beroende av adressöversättning.

Klienter som loggar på

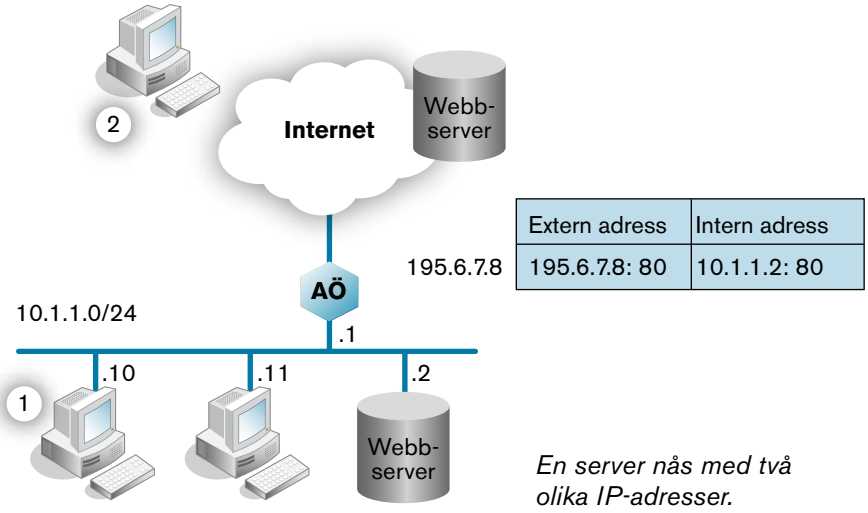
Många tjänster som innebär att en klient ska kunna anropas från Internet bygger på att klient-programmen registrerar sig på en publik server.

Adressöversättning har blivit så vanligt att tillverkarna måste få serverbaserade tjänster att fungera utan ”port forwarding” eller ”virtuella servrar”. En vanlig lösning är att tillverkarna lägger in att programmet alltid startas vid systemstart. Programmet börjar då med att anropa en server med en publik adress. Denna fråga passerar i princip alla brandväggar. Den publika servern håller reda på vilken IP-adress och port som skickade frågan och ute på Internet registreras det att ”Håkan Lindberg is on line”. Vill någon kontakta mig går frågan via den publika servern som skickar frågan vidare till min adressöversättare som vet vilken port och adress som ska få frågan på insidan.

Denna teknik fungerar givetvis men bygger på att klientprogrammen har en lista med kända servrar eller att användarna själv väljer en server. En annan lösning är att de klienter som sitter med publika adresser i princip blir mellanhänder (proxies) för klienter som använder adressöversättning.

Wrap around-problemet

I följande fall har vi en server med den interna IP-adressen 10.1.1.2. Vi har även konfigurerat upp en portöversättning så servern kan nås från Internet på IP-adress 195.6.7.8. Detta fungerar utmärkt men för noder som flyttar sig från det interna nätet till Internet eller åt andra hållet blir det problem. Vilken IP-adress som ska användas beror på var man befinner sig. Vilken IP-adress ska vi använda om vi vill nå vår interna server?



Extern adress	Intern adress
195.6.7.8: 80	10.1.1.2: 80

En server nås med två olika IP-adresser.

Klassik Internetteknik ger ett enkelt svar: vi ska använda den publika adressen. Det är bara den här adressen som är unik och som kan användas vid routing etc. Flera adressöversättare klarar att hantera det här problemet, som ofta kallas ”wrap around”. När en fråga kommer från insidan (fall 1 i bilden ovan) efter IP-adress 195.6.7.8 så översätts frågan. Adressöversättaren skickar ett paket från 10.1.1.1 till 10.1.1.2 som returneras av webbservern till 10.1.1.1. Lösningen i sig blir inte optimal, trots att klient och server sitter på samma nät så går alla frågor via default gateway.

Vissa adressöversättare kan inte hantera det här problemet, de

skickar inte frågan vidare på det privata nätet. Då får man lösa det via DNS eller ett routingkommando. En lösning är att ha två namnservers, en på insidan om adressöversättaren och en annan på utsidan. Dessa två namnservers ger olika svar på frågan vad servern har för IP-adress.

Problemet med ”wrap around” är generellt på Internet men det saknas ett vedertaget begrepp för det. Det är även svårt att ta reda på hur tillverkare stöder denna funktion. Problemet kommer knappast minska då användningen av privata adresser ökar och man samtidigt vill göra servers tillgängliga från utsidan.

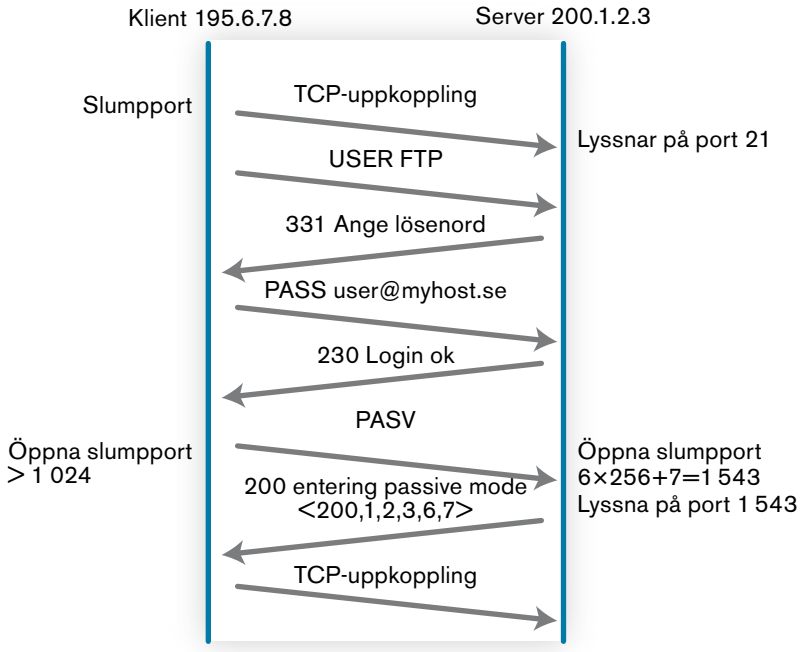
Adressöversättning och applikationer

En adressöversättare behöver kunna mer än att byta adresser. I och med att IP-adressen ändras måste IP-headerns checksumma räknas om. Faktum är att även TCP-headerns checksumma behöver räknas om. Ett protokoll som får alldeles uppenbara problem med adressöversättning är ICMP. ICMP:s uppgift är ju att skicka meddelanden när till exempel routing inte fungerar. Om vi till exempel tänker oss ett felmeddelande av typ ”Destination unreachable” så bör en adressöversättare titta igenom innehållet och eventuellt byta ut adresser.

Det finns även en mängd applikationer och protokoll som stöter på problem vid adressöversättning: FTP, NetBIOS, Kerberos, SNMP, IPsec, SIP med flera. Problemet ligger huvudsakligen i att IP-adresser skickas som data på nivå 5–7. I flera fall kan problemet lösas ifall adressöversättaren har stöd för en så kallad applikationsgateway (ALG – Application Layer Gateway). En annan lösning är självklart att använda protokollen internt och inte låta dem passera en adressöversättning överhuvudtaget, detta gäller i viss mån NetBIOS och SNMP. Fler lösningar finns, se referenslistan.

För att belysa problemets komplexitet kan vi titta på hur FTP fungerar.

Oavsett om Passive Mode används eller inte (se kapitlet IP-baserade program) så skickas IP-adresser över på applikationsnivå.



Handskakning under en FTP-session. I detta fall med passive mode.

(Formatet som används är att 200,1,2,3,6,7 motsvarar IP-adress 200.1.2.3 portnummer 6 × 256 + 7 = 1 543.) IP-adresser skickas alltså inte i hexadecimalt format. Detta medför att en översättning kan ändra på längden (från 200.1.1.3 till 192.193.194.195). Inte bara checksummorna måste räknas om, datamängden påverkar även hur sekvens- och kvittensummer beräknas på TCP-nivån.

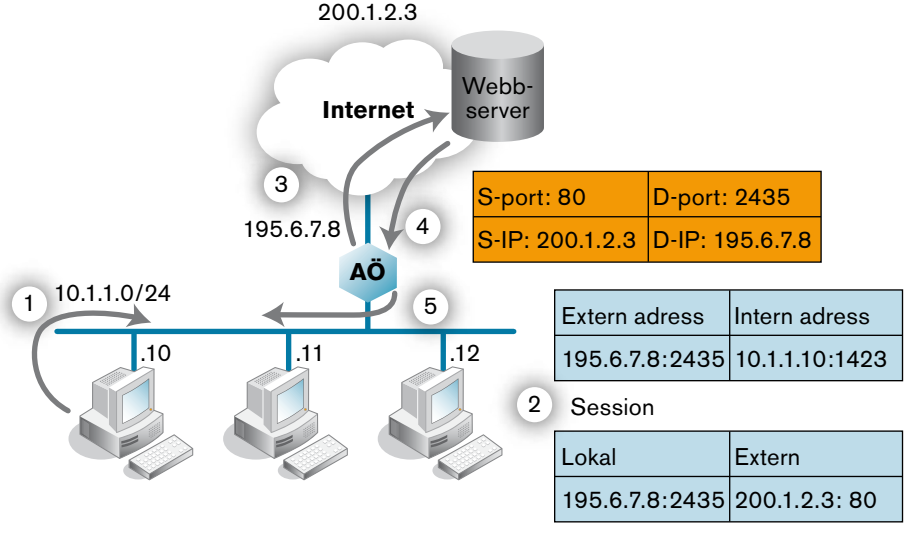
Adressöversättning fungerar inte bra med vissa applikationer.

En applikationsgateway som säger sig hantera FTP måste hantera ett flertal olika fall: traditionell (aktiv) FTP och Passive Mode, bakomliggande servrar och bakomliggande klienter. Och då är FTP ett ganska enkelt protokoll. Adressöversättning behandlas även i avsnitten om IP-telefoni och VPN. Här blir problemen ibland så komplexa att man delvis måste försöka undvika adressöversättning.

Fördjupning: tre olika former av NAPT

IP-telefoni är ett användningsområde som ökat och ökar. IP-telefoni har som sagts ovan problem med adressöversättning. För IP-telefoni används huvudsakligen transportprotokollet UDP, detta gäller SIP (Session Initiation Protocol) såväl som RTP (Real-time Transport Protocol). UDP är en utmaning inom adressöversättning då protokollet är förbindelselöst, det finns inga egentliga sessioner att hålla reda på. Istället bygger man funktionen kring timers (ett paket som kommer från en avsändare motsvarande det anrop adressöversättaren skickat ut för ett par sekunder sedan kan härledas till motsvarande privat adress och port).

Det finns tre huvudsakliga former av NAPT, vi ska ta och undersöka dem lite närmare. En adressöversättare startar nya sessioner mot de servrar den arbetar med (motsvarande information som man ser på en enstaka nod om man ger den kommandot netstat -an). Vi lägger därför till en tabell med fälten ”lokal” respektive ”extern” och noterar IP-adress och portnummer. Vi betecknar tabellen session, tekniskt sett finns ingen session om UDP används men vi hanterar som sagt detta med timers.



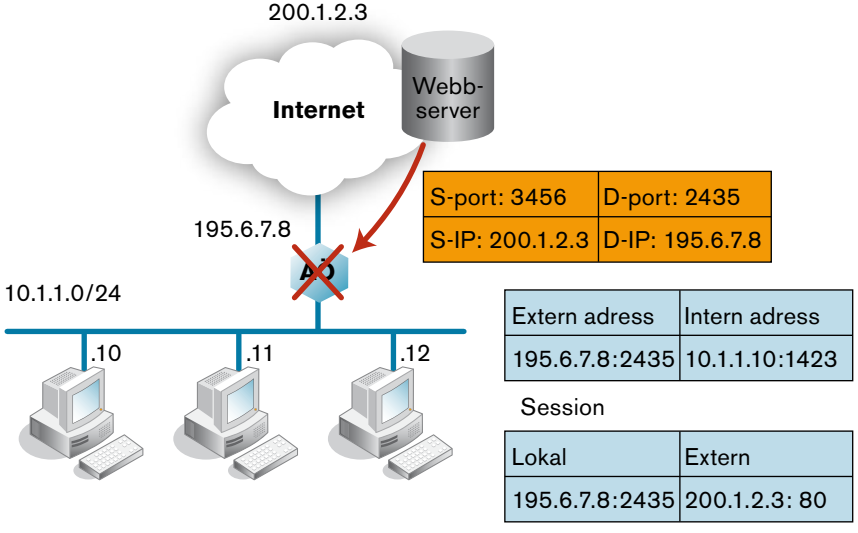
NAPT och adressöversättarens externa sessioner.

Den vanligaste typen av NAPT kallas för strutformad (full cone) . I samband med att paketet skickas ut externt lägger adressöversättaren upp en datapost med extern och intern IP-adress och port (punkt 2 i bilden). Efter att adressöversättaren lagt upp denna post så accepterar adressöversättaren alla inkommande paket på port 2 435 och skickar dem vidare till 10.1.1.10.

En annan typ av NAPT kallas för strikt, även begreppen partiell eller begränsad förekommer (restricted cone). Adressöversättaren accepterar alla inkommande paket på port 2 435 och skickar dem vidare förutsatt att avsändaren har rätt IP-adress (200.1.2.3 ovan). Skillnaden mot strutformad NAPT ligger alltså i hur den externa IP-adressen hanteras.

Den tredje typen kallas för symmetrisk. Adressöversättaren accepterar inkommande paket på port 2 435 och skickar dem vidare förutsatt att avsändaren har rätt IP-adress och port. Anrop till 2 435 men med fel avsändaradress eller fel avsändarport släpps inte igenom. Denna typ av NAPT är implementerad i vissa ADSL-enheter och vissa råbarkade brandväggar.

Symmetrisk NAPT fungerar inte med IP-telefoni. Och flera större brandväggar, som används i stora organisationer, använder symmetrisk NAPT.



Symmetrisk NAPT accepterar bara paket från en viss IP-adress och ett visst portnummer.

Symmetrisk NAPT fungerar inte med inkommande IP-telefoni, medan de andra två metoderna går att få att fungera.

För att få IP-telefoni att fungera genom en adressöversättare behövs flera funktioner. Klienten behöver:

1. Ta reda på sin externa adress innan den registrerar sig i växeln
2. Ta reda på vilken typ av adressöversättning som används: NAT eller NAPT, typ av NAPT
3. Hålla sessionen uppe så den inte ”timas ut” av adressöversättaren.

Source	Destination	Protocol	Info
192.168.3.245	62.80.200.56	STUN	Message: Binding Request
192.168.3.245	62.80.200.56	STUN	Message: Binding Request
192.168.3.245	62.80.200.56	STUN	Message: Binding Request
. . .			
192.168.3.245	62.80.200.56	STUN	Message: Binding Request
192.168.3.245	62.80.200.56	STUN	Message: Binding Request
192.168.3.245	62.80.200.56	STUN	Message: Binding Request
192.168.3.245	62.80.200.56	STUN	Message: Binding Request
192.168.3.245	**62.80.200.55**	**STUN**	**Message: Binding Request**
62.80.200.55	**192.168.3.245**	**STUN**	**Message: Binding Response**
192.168.3.245	62.80.200.56	STUN	Message: Binding Request
62.80.200.56	**192.168.3.245**	**STUN**	**Message: Binding Response**
192.168.3.245	62.80.200.55	STUN	Message: Binding Request
192.168.3.245	62.80.200.56	STUN	Message: Binding Request
192.168.3.245	62.80.200.56	STUN	Message: Binding Request
. . .			
192.168.3.245	62.80.200.56	STUN	Message: Binding Request
192.168.3.245	62.80.200.55	STUN	Message: Binding Request
192.168.3.245	62.80.200.55	STUN	Message: Binding Request
192.168.3.245	62.80.200.55	STUN	Message: Binding Request
62.80.200.55	192.168.3.245	STUN	Message: Binding Response
62.80.200.55	192.168.3.245	STUN	Message: Binding Response

Dessa tre funktioner hanteras av protokollet STUN (Simple Traversal of UDP through NATs) beskriven i RFC 3489. STUN har blivit populärt då det inte kräver ändringar av hur brandväggar eller protokoll fungerar.

Till vänster visas en inspelning av hur STUN fungerar. Vissa återkommande ”binding requests” har utelämnats. En privat adress anropar en STUN-server, servrarnas adresser har konfigurerats på klienten. En STUN-server måste inte svara, enligt RFC 3489 ska klienten fortsätta att skicka förfrågningar i drygt 30 sekunder innan den uppfattar kontakten som bruten. Klienten bör också skicka förfrågningar regelbundet för att upptäcka förändringar.

För att förstå vilken typ av NAPT som klienten sitter bakom behövs information från två STUN-servers, vilket syns i andra halvan av listan ovan. (de slutar på 55 respektive 56). Om vi undersöker de två typerna av paket från samma inspelning hittar vi:

Paket nr. 262
Ethernet II, Src: 00:0b:5d:c3:2f:98, Dst: 00:0f:cb:9f:4a:fa
Internet Protocol, Src: 192.168.3.245 Dst: 62.80.200.55
User Datagram Protocol, Src Port: 65340, Dst Port: 3478
  Source port: 65340
  Destination port: 3478
  Length: 28
  Checksum: 0xfc10 [correct]
    [Good Checksum: True]
    [Bad Checksum: False]
Simple Traversal of UDP Through NAT
  Message Type: Binding Request (0x0001)
  Message Length: 0x0000
  Message Transaction ID: FC160CF17571424CB2437EE5B7D081EC

Paket nr 263

Ethernet II, Src: 00:0f:cb:9f:4a:fa, Dst: 00:0b:5d:c3:2f:98
Internet Protocol, Src:.80.200.55, Dst: 192.168.3.245
User Datagram Protocol, Src Port: 3478 , Dst Port: 65340
  Source port: 3478
  Destination port: 65340
  Length: 96
  Checksum: 0x25c3 [correct]
    [Good Checksum: True]
    [Bad Checksum: False]
Simple Traversal of UDP Through NAT
  Message Type: Binding Response (0x0101)
  Message Length: 0x0044
  Message Transaction ID: FC160CF17571424CB2437EE5B7D081EC
  Attributes
    Attribute: MAPPED-ADDRESS
      Attribute Type: MAPPED-ADDRESS (0x0001)
      Attribute Length: 8
      Protocol Family: IPv4 (0x0001)
      Port: 65340
      IP: 85.224.171.125
    Attribute: SOURCE-ADDRESS
      Attribute Type: SOURCE-ADDRESS (0x0004)
      Attribute Length: 8
      Protocol Family: IPv4 (0x0001)
      Port: 3478
      IP: 62.80.200.55
    Attribute: CHANGED-ADDRESS
      Attribute Type: CHANGED-ADDRESS (0x0005)
      Attribute Length: 8
      Protocol Family: IPv4 (0x0001)
      Port: 3479
      IP: 62.80.200.56

Det första paketet, nr 262, innehåller inte mycket mer information än att det just är en ”Binding Request”. Portnumret som används är 3 478 för STUN-servern. En klient kan sätta flaggor i sin förfrågan och be servern att svara på till exempel en annan port.

Den IP-adress som kontaktade STUN-servern lyfts upp två nivåer och blir applikationsdata. Därför kan det nu skickas till klienten i form av ett ”Binding Response” under attributet ”MAPPED-ADDRESS. Vidare skickas information om vilken IP-adress och port STUN-servern har samt den IP-adress (CHANGED-ADDRESS) som kan användas om klienten vill testa mot en annan server för att bedöma vilken adressöversättningsteknik som används.

IPv6 och adressöversättning

Till viss del har små brandväggar och hemmabrandväggar blivit synonymt med adressöversättning. En stor mängd tillverkare säljer brandväggar som inte kan routa mellan det interna och det externa nätet, utan adressöversättning är enda möjligheten att överföra trafik. En del av poängen med IPv6 försvinner om vi fortsätter bygga nät på detta vis. Istället behöver vi publika adresser (förslagsvis IPv6) och lite kontroll på våra brandväggsregler.

Det går att sätta upp en brandvägg som skyddar det interna nätet på samma sätt som adressöversättning, vi blockerar helt enkelt frågor som initieras utifrån. Med lite kontroll på brandväggsreglerna slipper vi problem som wrap around och att brandväggen får agera gateway på applikationsnivå. Men det här scenariot kommer med ett pris och det är att brandväggarna under en övergångsperiod kommer att vara tvungna att hantera både IPv4 och IPv6 och dessutom tunnlar för kopplingar mellan dessa protokoll.

Referenser

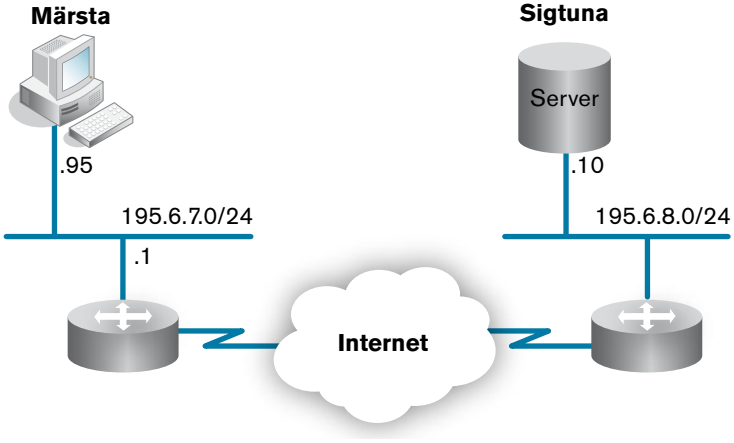
RFC 1631	The IP Network Address Translator (NAT)
RFC 1918	Address Allocation for Private Internets
RFC 2663	IP Network Address Translator (NAT) Terminology and Considerations
RFC 2694	DNS extensions to Network Address Translators (DNS_ALG)
RFC 3022	Traditional IP Network Address Translator (Traditional NAT)
RFC 3027	Protocol Complications with the IP Network Address Translation”
RFC 3235	NAT Friendly Application Design lines
RFC 3489	STUN – Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)

Felsökning i TCP/IP-miljö

- I detta kapitel går vi igenom felsökning på IP-nivå med hjälp av ping och traceroute. De funktioner som är inblandade i routing och namnuppslagning gås igenom samt hur man kan fastställa var i kedjan felet ligger. Felsökning av webb och e-post gås igenom kortfattat.
- Kapitlet kan läsas fristående.

Om vi tänker oss följande nätverk där en klient i Märsta ska nå en server i Sigtuna så finns det en massa komponenter och program som måste fungera. Servern och klienten måste självklart fungera, men även alla routrar till och från servern, namnuppslagning samt normalt även ett par brandväggar.

Grundläggande felsökning är att prova från en annan klient i Märstanätet. Eller att prova om man från klienten kan nå en annan server. En annan åtgärd är att undersöka länk-indikatorn på Ethernet, om den inte lyser finns ett fel på Ethernet-nivå. Men vi ska i detta läge utgå från IP-nivån.



Nätskiss hur IP-klienten i Märsta når servern i Sigtuna.

Vi börjar med att ta reda på den konfiguration som gäller med kommandot ”ipconfig /all” i Windows, ”ifconfig -a” i Unix. Vi kan då

kontrollera att standard gateway går att nå, observera dock att DNS inte behöver ligga på samma subnät.

Använd ipconfig eller ifconfig och kontrollera att routern kan nås. Fel på IP-nivån är ofta rena felkonfigurationer. Vi avsåg att mata in 195.6.7.95 men det blev 195.6.79.5 eller något liknande. Ett annat problem är vanan att alltid ställa masken till 255.255.255.0, det behöver inte alltid stämma.

```
Ethernet-kort Local Area Connection:

        Anslutningsspecifika DNS-suffix . . :
        Beskrivning . . . . . . . . . . . . : Intel(R) LAN 2435
        Fysisk adress . . . . . . . . . . . : 00-13-CE-26-F9-18
        DHCP aktiverat  . . . . . . . . . . : Nej
        Autokonfiguration aktiverat  . . . : Nej
        IP-adress . . . . . . . . . . . . . : 195.6.7.95
        Nätmask . . . . . . . . . . . . . . : 255.255.255.0
        Standard-gateway  . . . . . . . . . : 195.6.7.1
        DNS-servrar . . . . . . . . . . . . : 195.6.17.2
```

Börja med att kontrollera IP-konfigurationen.

Ipconfig-kommandot med tillägget /all ger dessutom information ifall DHCP används eller inte. Eftersom många nätverk använder samma privata IP-adressrymd är det lätt att komma från ett nätverk med nästan rätt konfiguration till ett annat. Bara ipconfig visar rätt resultat, IP-adresserna är samma i bägge nätverken. Med tillägget /all kan vi till exempel se om vi har rätt DNS och hur aktuellt DHCP-lånet är.

Det enklaste sättet att felsöka vidare är sedan att pinga sig ut. Vi börjar med vår egen adress (localhost), sedan pingar vi den normala adressen. Nästa steg är att pinga standard gateway (eller en annan nod på det lokala nätverket), namnservern och sist den server vi vill nå.

Exempel på hur du kan pinga dig ut, från localhost, via standard gateway till rätt server.

```
Ping 127.0.0.1
Ping 195.6.7.95
Ping 195.6.7.1
Ping 195.6.17.2
Ping 195.6.8.10 alt. ping www.server.se
```

I en mening är ping trivialt, men de mekanismer som krävs för att ping ska lyckas är inte triviala. Routing mellan klient och server måste fungera bägge vägarna. Det kan mycket väl vara så att klienten hittar till servern. Men när servern ska tillbaka till din klient har Märsta-nätet gjort ändringar som de inte meddelat sin Internetoperatör. Därför är nätet inte känt hos operatören så paketet kommer inte fram.

Även arp-kommandot kan var användbart. I bilden ovan ska arp-cahchen innehålla MAC-adressen för standard gateway. Använd kommandot ”arp -a”.

Ett annat sätt att felsöka är att utgå från vilka portar som är öppna, vilket vi kan kontrollera med kommandot ”netstat -an”. Detta är väldigt användbart på servrar. Aktuell port ska vara öppen och status på processen är vanligtvis ”listening”. Om en process gått igång och är i status listening men den externa adressen är 127.0.0.1 så accepterar servern bara inkommande förfrågningar från den egna maskinen. Kontrollera i så fall konfigurationsfilen.

Kommandot netstat -a kan även ge dig information om du tror att du har fått virus. Ett virus (eller en så kallad zoombie) kan ligga och vänta på en port. När någon sedan kopplar sig mot din dator kan han/hon använda den för egna syften.

```
Aktiva anslutningar

  Prot.  Lokal adress            Extern adress          Status
  TCP    0.0.0.0:80              0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5225            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5226            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:8008            0.0.0.0:0              LISTENING
  TCP    127.0.0.1:1026          127.0.0.1:5225         CLOSE_WAIT
  TCP    127.0.0.1:1042          0.0.0.0:0              LISTENING
  TCP    127.0.0.1:1115          127.0.0.1:5225         CLOSE_WAIT
  TCP    127.0.0.1:1119          127.0.0.1:5225         CLOSE_WAIT
  TCP    127.0.0.1:5226          127.0.0.1:1030         ESTABLISHED
  TCP    127.0.0.1:5679          0.0.0.0:0              LISTENING
  TCP    127.0.0.1:8005          0.0.0.0:0              LISTENING
```

```
TCP    127.0.0.1:10110        0.0.0.0:0              LISTENING
 UDP    0.0.0.0:445            *:*
 UDP    0.0.0.0:500            *:*
 UDP    0.0.0.0:4500           *:*
 UDP    127.0.0.1:123          *:*
 UDP    127.0.0.1:1900         *:*
```

Etablerade sessioner visas som ”established”. På klientsidan har vi inte kontroll på vilken port som används men vi kan se på den externa adressens port om det stämmer.

Ungefär år 2003 inträffade också en stor förändring i både Windows- och Unix-miljöer. Alla produkter levereras med olika typer av personliga brandväggar. Ibland fungerar det automatiskt men flera produkter har problem, även om man sätter igång serverprodukten så är brandväggen stängd. Även ICMP och specifikt ICMP echo är oftast avstängt som förvalt värde. Säkerheten ökar men felsökning blir komplexare.

Kontrollera att den interna (personliga) brandväggen är rätt inställd och öppen på rätt portnummer.

Namnservern

Alla IP-baserade protokoll klarar att användas med IP-adresser eller nodnamn. ”Ping www.sunet.se” eller ”ping 130.239.8.25” ska ge samma resultat. Ett enkelt sätt att kontrollera namnuppslagning blir alltså att pinga ett nodnamn och se att namnuppslagning fungerar.

Namnuppslagningen ser ut att fungera (se överst på nästa sida), vi ser att den försöker kontakta 130.239.8.25. Vill man göra en ytterligare kontroll kan man använda kommandot nslookup. Ur resultatet nedan kan vi se två saker förutom själva svaret på namnfrågan. Vi kan se vilken namnserver som levererat svaret (ns.local.teliamobile.net nedan) och vi kan se att svaret är mellanlagrat och inte har verifierats av den server som är ansvarig för domänen (det står att svaret är ”Non-authoritative”). Eftersom DNS bygger på

```
C:\>ping www.sunet.se

Skickar signaler till www.sunet.se [130.239.8.25] med 32 byte data:

Begäran gjorde timeout.
Begäran gjorde timeout.
Begäran gjorde timeout.
Begäran gjorde timeout.

Ping-statistik för 130.239.8.25:
    Paket: Skickade = 4, mottagna = 0, Förlorade = 4 (100 %),
```

mellanlagring finns ett problem när information ändras, det tar tid innan ändringen slår igenom.

För att kunna göra dig oberoende av fel information i DNS behöver du ett par saker och dessa bör du skaffa dig innan DNS fungerar dåligt. Dels bör du kunna IP-adresser till ett par servrar på Internet, eller centralt på företagsnätet. så du kan hoppa över namnuppslagningstjänsten.

```
C:\>nslookup www.sunet.se
Server:  ns1.local.teliamobile.net
Address:  10.0.0.1

Non-authoritative answer:
Name:    www.sunet.se
Address:  130.239.8.25
```

Om du misstänker problem med din namnserver kan du med nslookup eller kommandot dig ställa frågan till en annan namnserver. Leta därför upp en namnserver på Internet som tillåter rekursiva frågor, eller använd en webbsida som tillåter nslookup- eller dig-frågor.

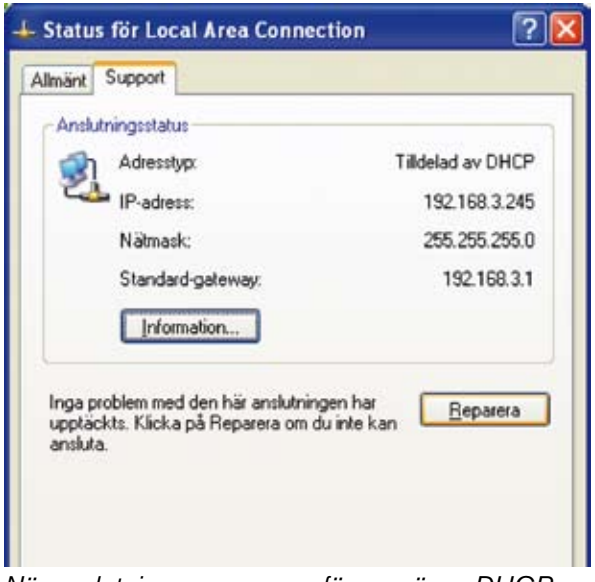
När PING inte går att använda

Tyvärr svarar inte alla noder och nätverk på ping. Organisationer kan ha stängt av ICMP echo eller till och med all ICMP trafik i

brandväggar eller med hjälp av filter. Ett alternativ är då att använda programmet telnet, men att anropa den port som servern förväntas svara på. Sedan förväntas man använda rätt kommando för att servern ska svara, men även om man skriver ett felaktigt kommando så brukar man få svar (servern svarar med att kommandot inte finns eller att syntaxen är fel).

```
C:\>telnet www.firma.se 80
```

Grundläggande funktioner i HTTP, SMTP, POP-3, FTP är inte svåra att lära sig. Det är bara att öppna rätt RFC och läsa. HTTP version 1.1, MAPI, SIP med flera är svåra att hantera. Men även om vi får ett felmeddelande så tyder ju det på att paket kommer fram till servern och tillbaka. Och det är ju den information vi får via ping.



När anslutningen repareras förnyas även DHCP-lånet.

Fel i DHCP

Dynamisk tilldelning av IP-adresser via DHCP har drygt tio år på nacken och är beprövad teknik. De flesta DHCP-klienter fungerar stabilt idag. I Windows 2000 och XP service pack 2 lade Microsoft till funktioner för att automatiskt förnya DHCP-lån varje gång förändringar sker på länknivån (till exempel om vi byter från ett WLAN-nät till ett annat). En förändring som i sin enkelhet gör att DHCP fungerar ännu bättre.

Om man fortfarande misstänker problem med den adress datorn erhållit via DHCP så kan man ge kommandot ”ipconfig /release” följt av ”ipconfig /renew”. I Windows XP är det enklare, DHCP-lånet förnyas om man väljer att reparera nätverksanslutningen.

Vad ska en nod göra om den ställs in på att använda DHCP men inte får kontakt med en

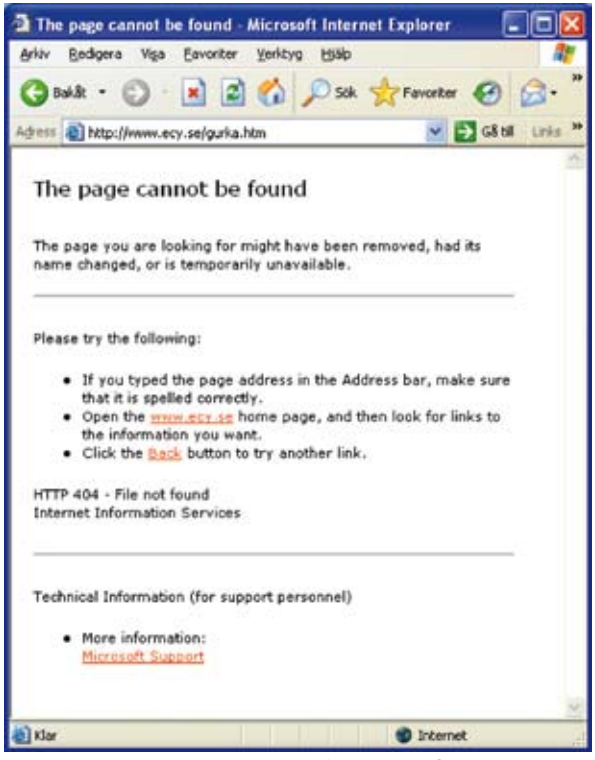
DHCP-server? Den skulle kunna låta bli att starta nätverkstjänsten. En annan lösning är att den tar en slumpgenererad adress ur adressrymden 169.254.0.0/16. Detta utrymme är reserverat för självkonfiguration. Microsoft kallar funktionen APIPA (Automatic Private IP Addressing), och den finns beskriven i tekniska dokument. Klienten tar en slumpartad adress och gör sedan en ARP-förfrågan för att kontrollera om någon annan maskin har denna adress. I så fall gör den ett nytt försök. Funktionen fungerar bra och gör att en ovan användare lätt kan få ett litet IP-nät att fungera internt då alla maskiner hamnar på samma subnät.

Om din maskin valt en IP-adress som börjar på 169.254. så tyder det på att den inte fått kontakt med en DHCP-server.

Webbservrar

Felsökning av webbservrar fungerar som vilken servertjänst som helst. På servern kan man kontrollera med netstat -an att servern lyssnar på port 80. Använd ping och traceroute för att kontrollera att paketen kommer fram. De vanligaste felen är felaktiga URL:er och felaktiga namn. Om vi skriver in en sökväg, eller klickar på en länk, som motsvarar en fil som inte finns så svarar servern med felmeddelande 404. Fel 404 visar att servern och routingen dit fungerar. (Jag undviker att säga något om kvalitén på Microsofts söktjänst om man skriver in fel nodnamn i Internet Explorer.)

De flesta webbsidor består ju av en mängd objekt. Var och en med sin egen sökväg. För att hämtning av webbsidor ska fungera effektivt och inte ge upphov till en mängd namnfrågor mellanlagras svaren. Men även misslyckade namnuppslagningar mellanlagras (så kallad negativ caching). Om du vet att det borde fungera så prova att stänga webbläsaren och öppna den på nytt en stund senare.



Fel 404 betyder att mycket fungerar. Servern är uppe och routingen dit fungerar.

Webbservrar är relativt lättkonverserade via telnet. Plocka fram kommandoläget och skriv in ”telnet webserver 80” samt slå enter två gånger. Du ska erhålla ett tomt fönster där servern väntar på ditt HTTP-kommando. Prova med något liknande (se till vänster):

```
GET /
GET /index.htm
GET /index.html HTTP/1.0
```

```
GET /index.html HTTP/1.1
host:  www.dinserver.se
```

Om du fick resultatet ”website not found” så prova med den modernare varianten av HTTP (se till vänster)::

Om det gick bra så får du en mängd HTML-kod som svar (utdrag nedan):

```
<li> <a hr
ef=”http://se.yahoo.com/”>Yahoo</a>, <a
href=”http://www.webcrawler.com/”>Webcrawler</a>, <a href=”http://www.altavista.
com/”>AltaVista</a>, <a href=”http://www.google.com/”>Google</a>,
<a href=”http://www.alltheweb.com/”>Alltheweb</a>
</
ul>
<HR>

This server is maintained by <A HREF=”MailTo:webmaster@sunet.se”>webmast
er@sunet.se</A>
<BR>
This page was last updated  2007-05-25, 14:43.
</body>
</html>
Connection to host lost.
```

E-post

E-post var en av de tidiga drivkrafterna för Internets utveckling. Det som först var enkelt och praktiskt, att man kunde nå sin SMTP-server från i stort sett vilken IP-adress som helst, har idag blivit svårt. SMTP-trafik filtreras och stoppas av operatörer och företagsnät. Spam i sig, men även rädslan för att få servrar svartlistade som spamservers, har gjort SMTP-trafiken komplicerad.

Om du väljer att sätta upp SMTP-server via en vanlig ADSL-anslutning kommer det i vanliga fall inte att fungera. Din Internetoperatör stoppar all SMTP-trafik till dig (port 25). En bakgrund till detta är att flera Linuxdistributioner startade en SMTP-server som standard, även om användaren främst tänkte sig servern för ett annat ändamål. Alltså fanns det en mängd e-post servrar som var felaktigt uppsatta. De hade dålig säkerhet och gamla versioner med kända säkerhetshål. Detta utnyttjades av dem som skickade spam och oskyldiga maskiner blev mellanstationer för spam.

Ett problem med SMTP har också varit att säkerheten är låg. Förr kunde man logga in på en SMTP-server och utge sig för att vara en godtycklig avsändare och skicka e-post till vem som helst. Det står i RFC 821 hur man ska göra. Det är fortfarande enkelt att utge sig för att vara någon annan med e-post. (Detta gäller ju faktiskt även vanliga brev.) Så var försiktig med hur du följer instruktioner du fått via e-post.

Det här har man försökt att lösa med att man idag behöver logga in för att få skicka e-post eller att SMTP-servern bara accepterar inkommande post från det lokala nätverk som den sitter på. Om vi använder telnet på port 25 ser vi att inloggning behövs:

```
C:\> telnet smtp.minserver.se 25
220 ironport.bredband.com ESMTP
EHLO mailtest.se
250-ironport.minserver.com
250-8BITMIME
250-SIZE 10485760
250-STARTTLS
250-AUTH PLAIN LOGIN
250 AUTH=PLAIN LOGIN
MAIL FROM: hakan@twoviews.se
250 sender <hakan@twoviews.se> ok
RCPT TO: hakan.lindberg@b3it.se
550 Sorry. SMTP AUTH required.
```

Hantering av e-post kräver ofta två protokoll. SMTP för att skicka e-post, POP eller IMAP för att hämta meddelanden. Detta gör konfigurationen lite svårare men även felsökning försvåras. Oftast är användning av POP

och IMAP öppen medan SMTP är stängd. Flera varianter finns också på hur säkerheten ska ökas. Till exempel att klienterna förväntas starta med att hämta e-post via POP, som kräver inloggning, för att sedan skicka med SMTP.

Ska man göra användning av e-post enkel så är det lättast att använda webbaserad e-post. Tyvärr är inte gränssnitten i webbläsare lika utvecklade.

Ping och traceroute

Det finns ett antal växlar till kommandot ping. Med växeln -l kan man påverka storleken på paketen. Ju längre paketet är desto längre tid tar det att skicka ut paketet. Kommandot är även användbart för att hitta problem med fragmentering och tunnling. Skillnaden i fördröjning och framkomlighet kan vara stor mellan paketlängderna 1 460 och 1 500 byte. Om en länk mellan server och klient använder tunnling (IP i IP eller IPsec VPN) så tillkommer headerinformation vilket gör att paketen behöver fragmenteras. En lösning är då att ställa ner MTU (Maximum Transmission Unit) på inblandade noder.

Med växlarna -i och -w kan man påverka intervallet mellan varje paket respektive hur lång tid man ska vänta på svaret. Vill man skicka ett flertal paket, ändå tills man avbryter med ctrl-C, lägger man till växeln -t (detta är standard i Linux och Unix).

Ping ger även information om fördröjning och antal routerhopp.

```
C:\>ping www.twoviews.se

Skickar signaler till www.twoviews.se [62.119.28.104] med 32 byte data:

Svar från 62.119.28.104: byte=32 tid=305ms TTL=50
Svar från 62.119.28.104: byte=32 tid=344ms TTL=50
Svar från 62.119.28.104: byte=32 tid=273ms TTL=50
Svar från 62.119.28.104: byte=32 tid=291ms TTL=50
```

```
    Ping-statistik för 62.119.28.104:
        Paket: Skickade = 4, mottagna = 4, Förlorade = 0 (0 %),
    Ungefärligt överföringstid i millisekunder:
        Lägsta = 273 ms, Högsta = 344 ms, Medel = 303 ms
```

Den genomsnittliga fördröjningen ovan är cirka 270 ms vilket är ett normalt värde för en länk via 3G, annars hade vi haft ett värde som varit cirka tio gånger lägre. När vi tog emot paketet hade den TTL = 50. De flesta avsändare sätter TTL till 128, 64 eller 32. Det är alltså rimligen 14 (64–50) routerhopp mellan vår klient och servern. Om vi använder traceroute kan vi verifiera detta.

```
C:\> tracert www.twoviews.se

Spårar väg till www.twoviews.se [62.119.28.104]
över högst 30 hopp:

  1  2445 ms   159 ms   160 ms  10.9.51.1
  2   137 ms   136 ms   159 ms  10.9.51.4
  3   136 ms  1630 ms   461 ms  10.9.50.2
  4   399 ms   488 ms   489 ms  212.181.222.131
  5   546 ms   509 ms   711 ms  vrrx50br2-fe2-0.teliamobile.net [192.36.252.137]
  6   376 ms   400 ms   398 ms  vrrx50ir1-fe2-0.teliamobile.net [192.36.252.214]
  7   402 ms   448 ms   451 ms  10.2.2.1
  8   417 ms   399 ms   419 ms  mobile-vrr.telia.net [192.36.252.198]
  9   354 ms   410 ms   408 ms  vrr-d2.link.se.telia.net [81.228.78.216]
 10   365 ms   459 ms   469 ms  g-ra-c1-link.se.telia.net [81.228.73.80]
 11   404 ms   479 ms   449 ms  g-br-peer1-link.se.telia.net [81.228.73.145]
 12   465 ms     *      493 ms  netnod-ix-ge-b-gbg-4470.utfors.net [195.69.116.66]
 13   348 ms     *      355 ms  ge-0-1-0.se-mlmms001-pe-1.tu.telenor.net [212.105.101.81]
 14   381 ms   469 ms   480 ms  62.119.244.106
 15   387 ms   499 ms   429 ms  www4.aname.net [62.119.28.104]

Spårning utförd.
```

Kommandot traceroute ger även värdefull information om hur trafiken kommer fram. Programmet gör även en baklängesuppslagning av de routrar som är inblandade och namnet ger en vägledning. Vi kan se att trafiken går via Telia Mobile via Internetknutpunkten (Netnod IX) till Telenors nät (vi ser rester av att Telenor köpt Utfors). Och vi ser att www.twoviews.se verkar ligga på ett webbhotell.

Referenser

Practical TCP/IP	Niall Mansfield Addison-Wesley 2003
How to troubleshoot TCP/IP connectivity with Windows XP	Microsoft, artikel-id 314067 (artikeln tar upp ett par aspekter som ej behandlats i detta kapitel)

Slutord

Du har just tagit dig igenom drygt hundrafemtio sidor om hur datapaket på Internet hittar fram. Förhoppningsvis har du nu en bra grund att stå på för att leta vidare. Många områden har inte behandlats så utförligt som de förtjänar i detta utdrag. Guiden rättar till några av dessa områden som IP-telefoni, mobilitet, routingprotokoll och VPN (Virtual Private Networks). Och cirka tio områden till.

Min ambition har varit att kontrollera saklig information, men visst kan det finnas fel i vad jag tagit fram. Om du hittat fel eller har synpunkter på denna guide kan du sända dem till publikationer @iis.se. Jag har också en hemsida med kontaktuppgifter om du vill kontakta mig: www.twoviews.se/alltoverip.

IP kanske inte är så märkvärdigt. Vi har bara gett datorer nummer och uppfunnit metoder för att hitta fram. Hade vi inte haft IP hade vi troligen haft något liknande, men det är inte säkert. Vi kunde fått en mängd proprietära lösningar. En eloge till alla som bidragit till utvecklingen och utrullningen av IP. Personligen använder jag inte begreppet ”webben 2.0”. Men begreppet speglar att för vissa grupper har Internet ändrats så mycket att de upplever nätet som nytt. Det är intressant och det kommer hända igen! I mitt jobb är en bärbar dator med webb och e-post samt en mobiltelefon mina huvudsakliga verktyg. De här verktygen fanns knappt för 15 år sedan, eller användes inte alls på samma globala sätt. Vi ses på Internet version 3.0 om några år.

Om Håkan Lindberg

Jag har arbetat med IP-teknik sedan tidigt 1990-tal och med data- och telekommunikation sedan slutet av 1980-talet. I botten är jag civilingenjör kombinerat med lite andra akademiska kurser. Jag har ett förflutet som konsult- och teknikchef hos operatören Sonera och

nätverksbolaget Cygate. Idag arbetar jag som seniorkonsult och partner på B3IT inom områdena kommunikation och mobilitet. Jag kombinerar konsultjobbet med att hålla utbildningar och föredrag.

Hjälp oss att förbättra
Om du hittat fel eller har synpunkter på denna guide kan du sända dem till publikationer@iis.se

Sakregister

K

L

M

N

O

P

R

S

T

U

V

.SE (Stiftelsen för Internetinfrastruktur) är en oberoende allmännyttig organisation som verkar för en positiv utveckling av Internet i Sverige. Utöver att ansvara för Internets svenska toppdomän bedriver .SE ett omfattande utvecklingsarbete:

- **.SE:s Internetguider** är en skriftserie som riktar sig till intresserade lekmannaanvändare och behandlar olika Internetfrågor. Denna publikation ingår i serien. Läs mer: iis.se/se-ar-mer/ses-publikationer.
- **Webbstjärnan** är en tävling i webbpublicering för skolan. Syftet är att dra nytta av Internets möjligheter i skolarbetet, genom att bygga en webbplats kring ett valfritt skolarbete. Läs mer: webbstjärnan.se och stjärnkikarna.se.
- **Internet för alla.** .SE bidrar till olika åtgärder för att förbättra tillgängligheten till Internet för de grupper som idag inte är anslutna till nätet.
- **Pålitlig e-post**. .SE undersöker vad som kan göras för att öka säkerheten och förtroendet för e-post för både företag och privatpersoner.
- **IPv6 och DNSSEC** är två viktiga teknikprojekt som ska säkerställa att Internets infrastruktur även i fortsättningen kan vidareutvecklas och fungera säkert. Läs mer: ipv6forum.se respektive iis.se/domaner/dnssec/.
- **Bredbandskollen** är ett konsumentverktyg som hjälper bredbandskunder att utvärdera sin bredbandsuppkoppling. Läs mer: bredbandskollen.se
- **Internetstatistik** .SE har tagit initiativ för att etablera ett samarbete kring statistik och fakta om Internet, vilket bland annat resulterat i den tryckta rapporten Svenskarna och Internet. Läs mer: iis.se/se-ar-mer/ses-publikationer och internetstatistik.se.
- **Internetfonden** bidrar till Internetutvecklingen genom att finansiera fristående projekt. Bland uppdragstagarna finns organisationer, privatpersoner och akademiska institutioner. Läs mer: iis.se/se-ar-mer/internetfonden.
- **Internetdagarna** är .SE:s årligen återkommande konferens för alla som arbetar med Internet. Läs mer: internetdagarna.se

.se

Som en del i .SE:s arbete med Internetutveckling producerar .SE ett antal skrifter under produktnamnet .SE:s Internetguider. Guiderna behandlar olika Internetrelaterade områden och riktar sig i första hand till intresserade lekmannaanvändare. En guide kan vara såväl en praktisk handbok som en mer beskrivande rapport.

.se
Stiftelsen för Internetinfrastruktur