I detta dokument hänvisas till kursbokens 6:e upplaga.

1. Blandade frågor 6p

1a) **(2p)**

Värddatoorn C kör DNS-servern med UDP-socket och portnummer 53 och den skapar inte några anslutningar med DNS-klienterna på värddatorerna **A** och **B. DNS-**kommunikationen är förbindelselös. DNS-servern har en enda socket för alla förfrågningar från DNS-klienterna. DNS-förfrågan kommer till servern med IP-adresser och samma portnummer 53. När servern svarar, den bara byter om IP-adresserna i IP header och skickar svaret via samma UDP-socket. Med andra ord att det är en gemensam server-socket på värddaorn C för alla klienter. På varje klient är det också bara en enda klient-socket för alla förfrågningar. Det är IP-adressen tillsammans med transaktion-ID som identifierar utbytet av DNS-informationen i båda riktningar.

1b) **(2p)**

Default route är den väg (via interface, next hop) som routern vidarebefordrar alla paket med mottagaraddresser som inte matchar specifika vägar (routes) i tabellen.

Destination	**Mask**	**Nästa hopp**	**Via interface**
0.0.0.0	0.0.0.0	x.y.z.w	interface “n”.

Varje rad i routingtabellen talar om ”för att nå en destination/subnätmask anlitas nexthop router och paketen skickas via interface”. Denna information kallas för en ”route”. Se ovan det speciella fallet om ”default route”.

1c) **(2p)**

Vägen; mellan slutanvändarna, genom Internet består av antal routrar sammankopplade med fysiska länkar som utgör delsträckor. De två viktigaste fördröjningarna som varierar från hopp till hopp är relaterade till de följande tiderna:

Transmissionstid: det tar tid att överföra databitarna för ett paket över en länk (antalet bitar/bithastigheten) och denna tid påverkas av begränsad bithastighet på den fysiska länken.
Kötid: inkommande paket skall behandlas av routern för att tas emot på ett interface och vidarebefordras till utgående interface. Det är väntetid när paket finns i routerns kö som orsakas av hög trafikintensitet så att det är för många paket som skall skickas vidare på ett utgående interface.

2. HTTP och Webben 6p

Se avsnitten 2.2.2 och 2.1.2 i kursboken

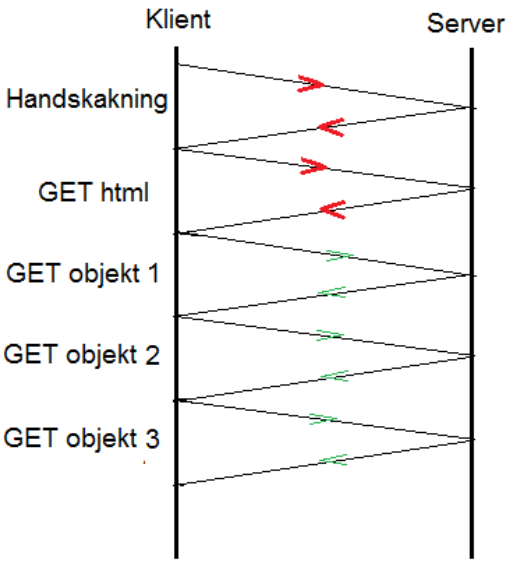
*För mer förklaring se **slides 18-22** i Kapitel_2 föreläsningsbilder.*
Se också kursbokens frågor (Ch.2 Problems: P7-8 och P10-11)

2a) **(1p)**

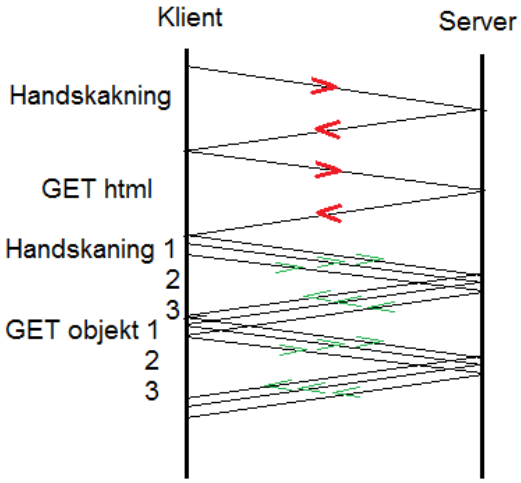
HTTP med beständiga ”persistent” anslutningar innebär att webbklienten först skapar en TCP-anslutning för att hämta HTML-filen och servern behåller denna anslutning öppen (Connection: Keep-Alive) tills klienten har hämtat de tillhörande objekten. Detta medför att klienten slipper skapa en TCP-anslutning för att hämta varje objekt. På detta sätt minskar man de allokerade resurserna.

2b) **(3p)**

I detta dokument hänvisas till kursbokens 6:e upplaga.



Seriell hämtning med beständiga (persistent) anslutningar



Parallell hämtning med icke-beständiga (non-persistent) anslutningar

I båda fallen skapas en TCP-anslutning för att först hämta html-basfilen, vilket tar tid motsvarande 2*RTT.

I- Efter det och med beständig anslutning, dvs. att den behålls öppen skickar klienten GET-meddelande för att hämta det första objektet som servern kommer att svara med HTTP-respons OK. När klienten fått det första objektet skickar den ett andra GET-meddelande för att hämta det andra objektet och väntar på att servern svarar med respons OK. Slutligen skickar klienten ett tredje GET-meddelande för att hämta det tredje objektet som servern kommer att svara med HTTP-respons OK. Förloppet därmed tar minst (2 + 3)*RTT för att webbläsaren kan visa hela webbsidan.

II- Med icke-beständiga anslutningar och efter hämtningen av html-filen stängs av anslutningen och klienten skapar **tre** olika nya anslutningar och genom varje anslutning begär klienten ett av de tre objekten med ett oberoende GET-meddelande. Servern kommer att svara med HTTP-respons OK på varje GET och förhoppningsvis direkt efter varandra. Förloppet därmed tar minst (2 + 2)*RTT för att webbläsaren kan visa hela webbsidan.

Förklaring:

Icke-beständigt ”non-persistent” HTTP innebär att webbklienten skapar separata TCP-anslutningar för att hämta olika objekt och servern stänger av varje anslutning när den svarar klart (Connection: close). Detta innebär; för att hämta varje objekt, måste klienten skapa en TCP-anslutning genom att handskakning (SYN-SYN/ACK-ACK) görs innan någon GET-meddelande kan skickas till servern.

Parallell hämtning (beständiga anslutningar): *klienten hämtar klart HTML-filen och sedan begär objekten ”back-to-back” dvs. klienten behöver* ***inte*** *vänta på svar* ***OK-respons*** *om ett objekt från servern innan det begär nästa med GET-meddelande. Klienten tillåts att begära fler objekt samtidigt (parallellt) men* ***oberoende*** *av varandra för att hämta de tillhörande objekten på webbsidan.*

I detta dokument hänvisas till kursbokens 6:e upplaga.

Parallell hämtning (icke-beständiga anslutningar): klienten hämtar klart HTML-filen och sedan skapar TCP-anslutningar parallellt dvs. klienten behöver **inte** vänta på svar om ett objekt innan de begär nästa. Klienten tillåts skapa parrallella anslutningar för att begära fler objekt (ofta 5 – 10) samtidigt oberoende av varandra men med varsin TCP-anslutning.

Seriell hämtning (beständiga anslutningar): klienten hämtar klart HTML-filen och sedan begär objekten **ett efter ett** dvs. klienten **väntar på svar OK-respons** om ett objekt från servern innan det begär nästa med GET-meddelande.

Seriell hämtning (icke-beständiga anslutningar): *klienten hämtar klart HTML-filen med en TCP-anslutning och sedan begär klienten de tillhörande objekten på samma sätt ett efter ett dvs. klienten väntar på svar om ett objekt innan det begär nästa. Detta medför att en tid på RTT behövs att skapa TCP-anslutningen och en tid på RTT behövs att hämta bas-filen och sedan på samma sätt en tid 2*RTT för varje objekt.*

2c) **(2p)**

I-

en lokal socket hos klienten:
Källadress: 129.16.216.14. **portnummer**: 53980
Destination: 128.119.245.12. **portnummer:** 80

Omvänt hos servern där skapas klientens s.k. ”connection socket”
Källadress: 128.119.245.12. **portnummer**: 80
Destination: 129.16.216.14. **portnummer:** 53980

II-

För att hämta html-filen är det samma som ovan. Sedan etableras tre nya sockets på varje sida, med skillnaden på tre nya klient-portnummer:

Hos klienten: tre sockets med samma **Destination:** 128.119.245.12. **portnummer:** 80
Källadress: 129.16.216.14. **portnummer**: 53981
Källadress: 129.16.216.14. **portnummer**: 53982
Källadress: 129.16.216.14. **portnummer**: 53983

Hos servern: tre sockets med samma **Källadress:** 128.119.245.12. **portnummer**: 80
Destination: 129.16.216.14. **portnummer:** 53981
Destination: 129.16.216.14. **portnummer:** 53982
Destination: 129.16.216.14. **portnummer:** 53983

I detta dokument hänvisas till kursbokens 6:e upplaga.

3. DNS 6p

Se avsnitt 2.5 i kursboken och Wireshark-labb (DNS)

3a) **(3p)**

DNS-namn: Värddators namn ”hostname” består av registrerade namn separerade med punkt ”dot”. Exempelvis: **www.cse.chalmers.se**

Ordningen i namnet från höger är:
TLD-namn för ett land enligt ISO-standard (**.se**, .uk, .dk, ..) eller verksamhet (.com, .org, .edu, ..)
Domännamn för organisationens ägare (**.chalmers**, .google, .kth, .vasttrafik, ...)
eventuellt **Subdomän** under domänen (**.cse**, .ita, .student, ...)
Namn på värddator, server, router, .. som väljs av organisationens administrator (**www**, webmail, pingpong, ...)

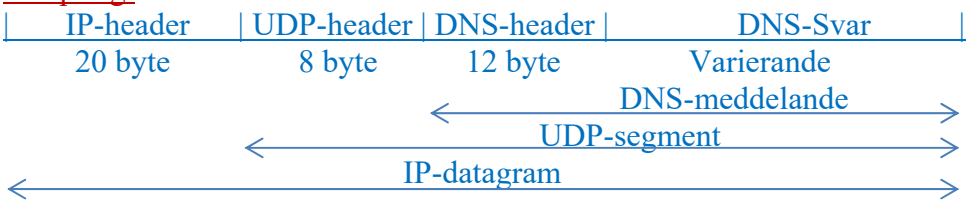
DNS-servrarna: Root, TLD, Auktoritativa
Root-servern har databas med RRs för namnen och IP-adresser av TLD-servrarna.
TLD-servern har databas med RRs för namnen och IP-adresser av de auktoritativa servrarna.
Den **auktoritativa** servern har databas med alla RRs som tillhör organisationens nätverk (t.ex. namnen och IP-adresser för de olika servrarna som tillhör organisationen).

3b) **(3p)**

DNS använder främst User Datagram Protocol (UDP) på port nummer 53 för att transportera förfrågningar och svaren. DNS-meddelanden består av antingen en enda DNS-förfråga från klienten eller av ett enda DNS-svar från servern. Dessutom är DNS-meddelandet kompakt och kort dvs. innehåller data på ett antal fördefinierade fält som vanligen inte överstiger 512 byte. UDP header är 8 byte och innehåller portnummer för DNS (53). IP-header är normalt 20 byte och innehåller I-adresserna för klienten (destination) och servern (avsändare).

.....
Applikation, DNS, Meddelande
.....
Transport, UDP, Segment
.....
Nätverk, IP, Datagram
.....
Nätverksaccess

Inkapling:



I detta dokument hänvisas till kursbokens 6:e upplaga.

4. TCP-protokollet 8p

4a) Se avsnitt 3.2 i kursboken (2p)

Dessa nummer är sändar-portnummer och mottagar-portnummer och används av både TCP och UDP för att identifiera vilket socket (applikation) som segmentets data kom ifrån på en host och till vilket socket (applikation) på den andra hosten skall data levereras. På server-sida är portnummret allmänt känt (0-1023) för en specifik applikation, medan på klient-sidan ofta slumpas numret från ett interval (49152–65535).

4b) (2p)

- *ICMP-meddelande om fel-rapportering (direkt inkapslas i IP-paket)*
- DHCP-meddelande om IP-konfiguration
- *HTTP-meddelande med metoden HEAD(använder TCP, kontrollerad transport)*
- RIPv2-uppdateringsmeddelande

- **DHCP-meddelande om IP-konfiguration**
  Se avsnitt 3.3 i kursboken
  DHCP är ett applikation-lagers protokoll (klient/server förhållande) för bootstrap och **använder UDP** för att transportera meddelanden i IP-paket till **broadcast**-adressen (255.255.255.255) som mottagaradress. Dessutom kan TCP-anslutningen inte skapas från en klient som **saknar IP-konfiguration**.

- **RIPv2-uppdateringsmeddelande**
  Se avsnitten 3.3 och 4.6.1 i kursboken samt Lab-2 (Network Addressing)
  RIPv2 är ett applikation-lagers protokoll för att skicka **regelbundna** routing-uppdateringar till routerns grannroutrar oavsett hur många och därför **använder UDP** för att transportera meddelanden i IP-paket till **multicast**-adressen (många gruppmedlemmar i RIPv2-routrar) som mottagaradress. Dessutom är ett RIP-meddelande **aktuellt just nu**, så är det opraktiskt och meningslöst med att använda TCP för att försöka rätta till eventuella fel eller förlust, det kommer att uppdateras ändå av ett senare meddelande.

4c) Se avsnitt 3.7 i kursboken (4p)

- Det är TCP-protokollet på varje sändare (varje TCP/IP-värd) som kontrollerar stockningen på end-to-end basis. Det är kollektivt ansvar för alla Internet-anslutna TCP/IP värdar, till skillnad från en nätverksbaserad stockningskontroll.

Genom att varje TCP-sändare håller en variabel kallas för ”Congestion Window” CongWin som anger hur mycket data (t.ex. i antal segment) som sändaren får skicka i väg på en gång utan att behöva vänta på ACK på varje segment.

- Händer det timeout när sändar-TCP väntar på ett ACK indikerar detta antingen långa köer eller paketförlust, allvarlig stockning. Om ett trippel duplikat ACK tas emot för ett och samma segment, indikerar detta en kortvarig stockning. Detta innebär att ett tidigare segment bland de sända segmenten är förlorat.

Syftet med stockningskontroll är att förhindra TCP-sändaren from att överbelasta nätverket dvs. de mellanliggande **routrarna** med paket som dem inte hinner med att

vidarebefordra vilket leder till långa köer och eventuell paketförlust. När varje TCP-sändare deltar i åtgärderna för att hantera, undvika stockningen oavsett orsaken, garanterar protokollet att snabbt lösa eventuella stocknings situationer.

- ”Slow Start” är det då sändar-TCP börjar försiktigt genom att sända bara ett segment först och sedan ökar sändningshastighet genom att fördubbla antalet segment i CongWin efter varje RTT dvs. exponentiell ökning (om det får ACK på alla tidigare sända segmenten) tills det når en tröskel. Vid början av sändningen sätts tröskeln till ett default värde.

”Congestion Avoidance” är det då sändar-TCP har nått tröskeln och börjar öka CongWin med ett segment i taget efter varje RTT dvs. linjär ökning (om det får ACK på alla tidigare sända segmenten) för att undvika stockningen.

Händer det timeout när sändar-TCP väntar på ett ACK återgår TCP i alla fall till ”Slow Start” med ett nytt tröskelvärde, men om ett trippel duplikat ACK tas emot för ett och samma segment, indikerar detta en kortvarig stockning. Detta innebär att ett tidigare segment bland de sända segmenten är förlorat. TCP övergår till ”Congestion Avoidance” efter ”Fast Recovery” dvs. sända om segmentet. Nya värdet på tröskeln anger storleken i antal segment på halva “Congestion Window” vid senaste händelse.

5. Ethernet & Trådlöst LAN 8p

5a) Se avsnitt 5.4.3 i kursboken (4p)

- Värddatorn **A** börjar med att genomföra AND-operation mellan serverns IP-adress och subnätmasken och konstaterar att den tillhör inte samma subnät .

- Värddatorn **A** är konfigurerad med IP-adress för access-routern **R** som default gateway för att skicka paket utanför sitt eget subnät. IP-paket skall kapslas in i Ethernet-ramar inför överföringen inom det lokal nätverket . Då behöver värddatorn **A** veta **R**-MAC adress och med hjälp av **ARP** skickas en broadcast-förfråga så att den som har **R**-IP adress skall svara med sin **R**-MAC adress.

- När ramen innehållande ARP-förfrågan kommer till switchen via porten **A** där värddatorn **A** är ansluten, sparar switchen sändarens MAC-adress dvs **A-MAC** i sin tabell mappat till port **A** och eftersom mottagar-adressen är broadcast (FF-FF-FF-FF-FF-FF) kommer switchen att vidarebefordra kopia av ramen till alla andra porter (port **B**, port **C** och port **R**) utom port **A**.

Kopior av ramen når **B** och **C** som läser av IP-adressen i **ARP**-meddelandet och konstaterar att det inte är sin egen och inget görs. Samtidigt får **R** en kopia av ramen via sin switchport och läser av IP-adressen i ARP-meddelandet och konstaterar att det är sin egen. Då skickar **R** sin MAC-adress i ett ARP-svar inkapslat i en unicast-ram på det lokala nätverket, adresserat till **A-MAC**.

När ramen innehållande ARP-svar kommer till switchen via porten **R** där access-routern **R** är ansluten, sparar switchen sändarens MAC-adress dvs. **R-MAC** i sin tabell mappat till port **R**. Mottagar-adressen **A-MAC** finns ju redan i switchens tabell och därmed vidarebefordrar switchen ramen endast till port **A**.

I detta dokument hänvisas till kursbokens 6:e upplaga.

Värddatorn **A** tar emot ARP-svaret och sparar **R-MAC** i ARP-tabellen mappat till **R-IP**. Nu kan värddatorn **A** skicka IP-paket (innehållande TCP-segment) som är adresserade till serverns IP-adress genom att kapsla dessa paket i Ethernet-ramar adresserade till **R-MAC**. Access-routern **R** använder sedan sin routingtabell för vidareleverans av dessa paket över Internet. Switchen kommer att vidarebefordra ramarna direkt mellan port **A** och port **R**.

5b) **Se avsnitt 6.3.2 i kursboken** **(3p)**

I **WLAN** tillämpas **CSMA/CA** mekanismer kollektivt (Multiple Access) så att en trådlös station STA som vill sända en ram med *normal* storlek, skall först lyssna på radiokanalen (Carrier Sense) och se om det är ledig.

Är kanalen ledig, väntar STA en förbestämd tid **DIFS**, sänder **hela ramen** om kanalen är fortfarande ledig och sedan väntar STA på **ACK** från mottagaren (som i detta fall är den associerande accesspunkten AP). Om ramen tagits emot felfitt skickas ACK av AP efter att ha tillämpat samma regler som ovan men med en kortare väntetid **SIFS**. En positiv bekräftelse ”ACK” används av CSMA/CA för att informera sändaren om lyckad överföring över radiolänken. ACK är nödvändigt med anledningen av att radiolänken är mer utsatt för störningar, brus och interferens så att de sända ramarna kan lätt drabbas av bitfel och även **kollisioner** kan, trots dessa mekanismer, inträffas.

Är kanalen upptagen, backar STA och först **när kanalen blir ledig startar stationen nedräkningen** av en slumpmässigt vald tid (**back-off time**). Med olika valda tider undviker man kollisioner (Collision Avoidance) när två eller fler associerade trådlösa enheter försöker **samtidigt** sända och väntar på att kanalen blir ledig. Utebliven ACK är en indikation för sändaren om att försöka sända om samma ram och därför **ökas väntetiden** (**back-off time)** av sändaren **inför** nästa sändningsförsök.

5c) **Se avsnitt 6.3 i kursboken** **(1p)**

AP har förutom **trådlöst interface,** ett **Ethernet-interface** anslutet till en switch. AP:ens uppgift vid en sådan installation, är att förmedla all trafik mellan stationerna oavsett MAC-typen (802.3 eller 802.11) och omvandlar ramarna från ena sidan till den andra.

AP:en utför sina arbetsuppgifter genom att hantera innehållet i headerfälten på ramarna enligt ett MAC-protokoll som utför funktioner på länklagret (lager-2). AP arbetar aktivt och deltar i kommunikationen på länk-lagret genom att vara mottagare/sändare för 802.11 MAC-ramarna på radiolänken.

När AP får en Ethernet-ram med mottagare-MAC-adress som tillhör en av de associerade trådlösa STAs inom sitt täckningsområde, extraherar accesspunkten datafältet, skapar en ny .11-ram som adresseras med användning av adresserna i Ethernet-ramen och AP sänder ramen över den trådlösa radiolänken.

Omvänt om AP får en .11 ram från en associerad STA och med mottagare-adress som **inte** tillhör annan STA, extraherar accesspunkten datafältet, skapar en ny .3 ram som adresseras med användning av adresserna i .11-ramen och sedan skickas ramen över Ethernet till switchen den är kopplad till.

I detta dokument hänvisas till kursbokens 6:e upplaga.

6. IP-adresser och subnetting 6p

Se avsnitt 4.4.2 i kursboken och Lab-2 (IP-Adressering)

En lösning är att först dela upp prefixet **33.22.20.0/25** i två lika stora subnät:

- De första subnätet **33.22.20.0/26** skall subnättas ytterligare i två mindre subnät. Det ena sub-subnätet **33.22.20.0/27** kan användas för adresseringen av en av de mindre avdelningarna (del 1) medan det andra sub-subnätet **33.22.20.32/27** skall användas för adresseringen av det andra (del 2).

- Det andra subnätet **33.22.20.64/26** tilldelas den stora avdelningen (del 3)

(Alternativt delas subnätet **33.22.20.64/26** upp i två mindre subnät **33.22.20.64/27** resp. **33.22.20.96/27** för de två mindre avdelningarna och subnätet **33.22.20.0/26** tilldelas den stora avdelningen)

6a) **(3p)**

33.22.20.0, mask: 255.255.255.224 för den första mindre avdelningen, del 1 vilket ger 30 IP-adresser **33.22.20.1** upp till **33.22.20.30**

33.22.20.32, mask: 255.255.255.224 för den andra mindre avdelningen, del 2 vilket ger 30 IP-adresser **33.22.20.33** upp till **33.22.20.62**

33.22.20.64, mask: 255.255.255.192 för den tredje stora avdelningen, del 3 vilket ger 62 IP-adresser **33.22.20.65** upp till **33.22.20.126**

6b) **(1p)**

3st. Ethernet Interface 100/1000 Mbps:

33.22.20.**1**/27 33.22.20.**33**/27 33.22.20.**65**/26

Seriell Interface: 33.22.10.**22**/30 (Förutsatt att ISP använder 33.22.10.**21**/30)

6c) **(2p)**

Default route (0.0.0.0/0) är den väg som router väljer i sista hand när ingen match finns med de befintliga specifika routes i routingtabellen. Företagets router behöver en sådan default route därför att den inte deltar i någon routingprocess och behöver statiskt konfigurerad instruktion om vart den skall vidarebefordra paket med mottagaradress som inte tillhör företagets prefix.

Destination	mask	via next hop	interface
0.0.0.0	**0.0.0.0**	**33.22.10.21**	**seriell (mot ISP)**
33.22.20.0	255.255.255.224	direktansluten	Ethernet 1
33.22.20.32	255.255.255.224	direktansluten	Ethernet 2
33.22.20.64	255.255.255.192	direktansluten	Ethernet 3
33.22.10.20	255.255.255.252	direktansluten	Seriell