Labbrapport - Datorkommunikation 7.5hp (Umeå)

Inledning

I den här labbrapporten redogör jag svaren på de förberedande frågorna och resultaten på de praktiska uppgifterna som jag har blivit ombedd att utföra och analysera. De praktiska uppgifterna utfördes med hjälp av Wireshark för att analysera nätverkstrafiken i realtid. Målet är att visa min förståelse för nätverksprinciper, dess samband och dess funktion.

OBS: Jag har redigerat bort känsligt innehåll för att värna om skolans integritet.

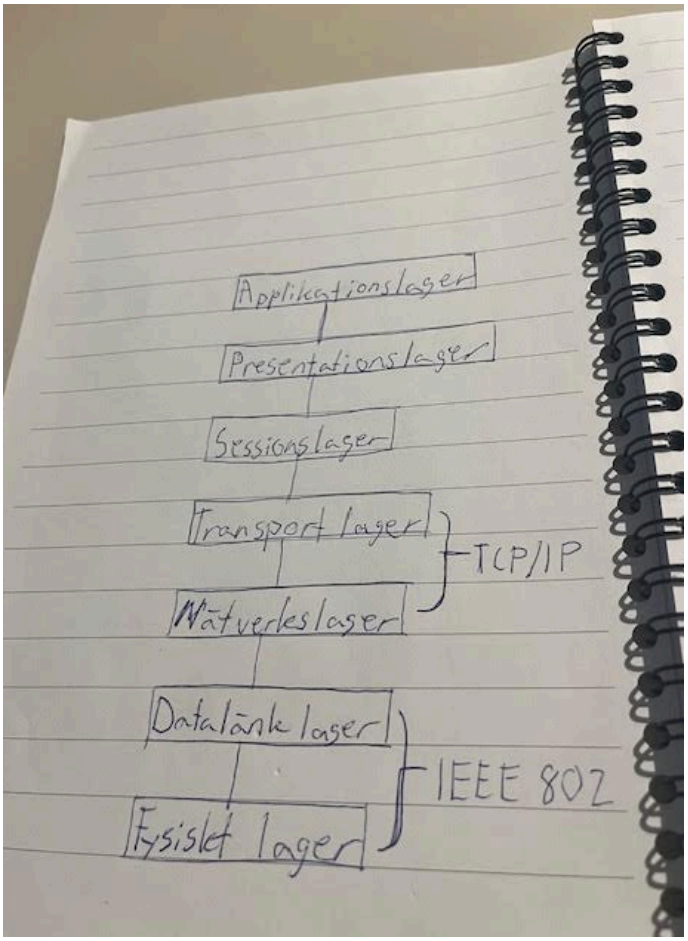
En MAC-adress är uppbyggd i två delar med 24 bitar i respektive del[1]. Den första delen används för OUI, eller Organizationally Unique Identifier, vilket är bestämd av IEEE och vars funktion är att identifiera en tillverkare. Den andra delen är ett unikt nummer som tillverkaren tilldelar nätverkskortet för att identifiera den i nätet.

Det finns totalt 2^{48} möjliga kombinationer för att skapa en MAC-adress. Vissa adresser är reserverade och kan inte användas för enskilda NIC, till exempel broadcast-adressen FF:FF:FF:FF:FF:FF samt multicast-adresser.

I figur 1.1 ritade jag upp OSI-modellen och placerade IEEE 802 protokollen och TCP/IP protokollen vid respektive lager de arbetar på. IEEE 802 valde jag att placera vid fysiska lagret och datalänk lagret eftersom ethernet definieras av de fysiska medium som används såsom twisted pair kopplar kablar samt logiken med mac-adressering i switchade nätverk.

Jag valde att placera TCP/IP protokollen vid nätverkslagret och transportlagret då IP-adressering arbetar på nätverkslagret i form av routing, och TCP logiken arbetar på transportlagret där den hanterar flödeskontroll.

[1] Wikipedia, MAC-address, https://en.wikipedia.org/wiki/MAC_address [HÄMTAD: 27 september 2025]



Figur 1.1: OSI-modellen med protokoll placerad vid tillhörande nivå.

ARP används för att översätta en nods lokala IP-adress till en MAC-adress.[2] Detta är möjligt eftersom nodens IP-adress ligger på lager 3 i OSI modellen medan dess MAC-adress ligger på lager 2. En nod kommer in på nätet uppifrån och ned, det betyder att noden alltid tar reda på IP-adressen innan MAC-adressen.

När en nod vill skicka ett paket inom ett nätverk arbetar den på lager 3 i OSI-modellen och har därför inte tillgång till MAC-adresser. För att ta reda på vilken MAC-adress en annan nod har kan den istället göra en förfrågan till lager 2 med hjälp av IP-adressen. Detta kallas för en ARP-Request.

ARP-Request innebär kortfattat att ett broadcast meddelande skickas ut till samtliga noder i nätet som säger “Vem har denna IP-adress och vad är din MAC-adress?”. Endast den noden som har IP-adressen kommer att svara med sin MAC-adress, och på så vis vet sändaren nu vart den ska adressera paketet.

Sändaren kommer sedan spara denna koppling i sitt ARP-cache så att den inte behöver göra fler förfrågningar.

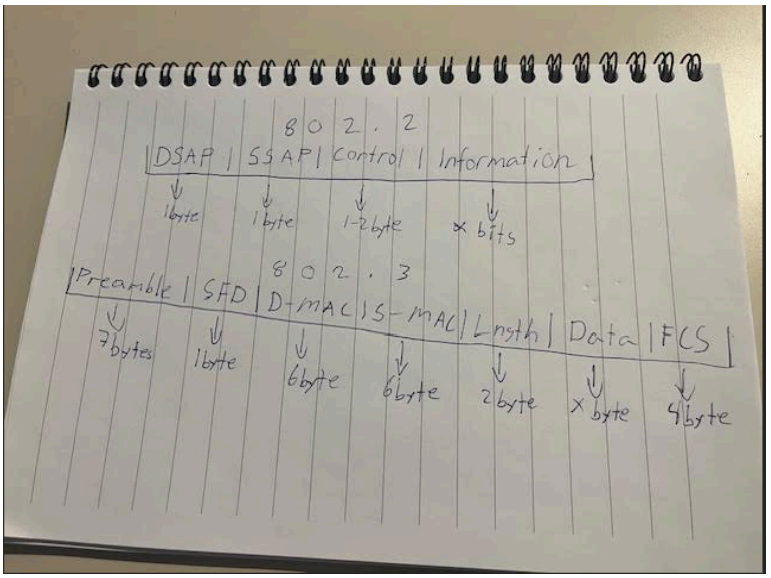
---

[2] Wikipedia, Address Resolution Protocol, https://en.wikipedia.org/wiki/Address_Resolution_Protocol [HÄMTAD: 28 september 2025]

I figur 1.2 har jag skissat hur ethernetpaket enligt IEEE 802.2 och 802.3 är utformade.[3] Nedan följer min beskrivning av respektive fält i dom.

802.2
DSAP = Destination Service Access Point, beskriver vilket protokoll på nätverkslagret som ska ta emot paketet.
SSAP = Source Service Access Point, beskriver vilket protokoll på nätverkslagret hos sändaren som skickade paketet.
Control = Beskriv vilket typ av paket det är och vilken information mottagaren kan förvänta att den innehåller. Lägger en grund till förståelse för olika nätverksprotokoll.
Information = Den faktiska informationen som sändaren vill sända.

802.3
Preamble = Används för att tidssynkronisera mottagarens nätverkskort med sändarens nätverkskort så att paketet avläses korrekt.
SFD = Start Frame Delimiter, används för att markera slutet på preamble och början på den viktiga informationen i paketet.
D-MAC = Destination Media Access Control Adress, mottagarens NIC-adress.
S-MAC = Source Media Access Control Adress, sändarens NIC-adress.
Length = Berättar hur många byte som datamängden är.
Data = Den faktiska informationen som sändaren vill sända.
FCS = Frame Check Sequence, använder CRC för att upptäcka error i paketets sändning.

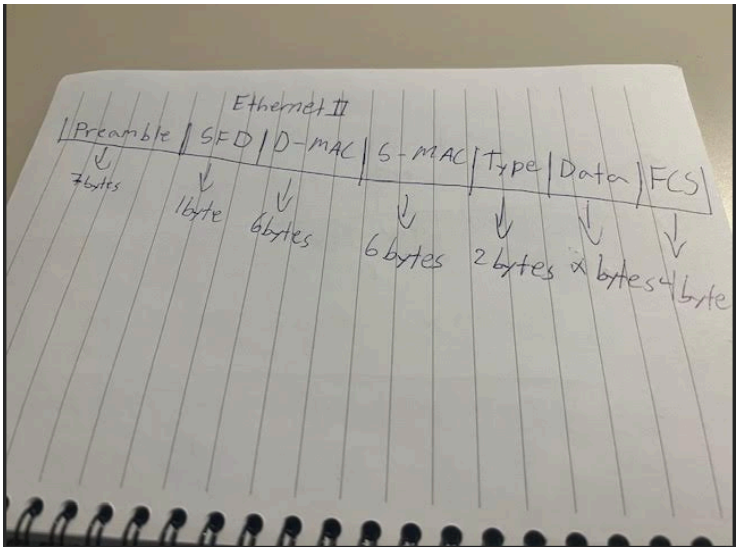


Figur 1.2: Skissade ethernetpaket enligt IEEE 802.2 och 802.3

[3] Wikipedia, Ethernet Frame, https://en.wikipedia.org/wiki/Ethernet_frame [HÄMTAD: 28 september 2025]

I figur 1.3 har jag skissat hur ethernetpaket enligt Ethernet II, eller RFC 894, är utformade. Nedan följer min beskrivning av respektive fält i den.

Preamble = Används för att tidssynkronisera mottagarens nätverkskort med sändarens nätverkskort så att paketet avläses korrekt.
SFD = Start Frame Delimiter, används för att markera slutet på preamble och början på den viktiga informationen i paketet.
D-MAC = Destination Media Access Control Adress, mottagarens NIC-adress.
S-MAC = Source Media Access Control Adress, sändarens NIC-adress.
Type = Berättar vilket protokoll som ligger i datan, exempelvis IPv4 eller IPv6.
Data = Den faktiska informationen som sändaren vill sända
FCS = Frame Check Sequence, använder CRC för att upptäcka error i paketets sändning.



Figur 1.3: Skissat ethernetpaket enligt Ethernet II

Resultat och analys av praktiska övningar

Uppgift 1:

Denna uppgift bestod av att utforska och bekanta mig med Wireshark och dess funktioner såsom att förstå hur paket kan fångas, filtreras och analyseras. Capture filter filtrerar det man aktivt spelar in i realtid och kan användas för att effektivt begränsa mängden trafik som hanteras. Display filter filtrerar däremot det som man redan har spelat in för att effektivt analysera en hög mängd data.

Sökfunktionerna i Wireshark använder samma syntax för operatörer som är standardiserade i de populariserade programmeringsspråken. Det finns hårdkodade operander som kan användas för att filtrera den fångade datatrafiken. Till exempel så kan man skriva “ICMP” för att endast se ICMP paket, eller “ICMP && ARP” för att se ICMP och ARP paket.

När jag utförde denna uppgift så analyserade jag mina första ethernet frames och undersökte vilken statistik jag kunde hitta. Jag kunde bland annat se ping-förfrågningar och

svar när jag pingade Cloudflares DNS-server. Jag kunde också se hur min dator kommunicerade med nätoperatörens DHCP-server via min router, och hur vissa TCP paket behövde skickas om på grund av krockar i nätet. Jag kunde även se slutförda TLS handshake messages som visar att en krypterad förbindelse har skapats, och ARP-förfrågningar och svar mellan min dator och nätverket. En sak jag noterade är att preamble, sfd och fcs inte syns i Wireshark.

Jag provade också att använda filtret “ether host BC:EE:7B:8D:45:BF” i Display Capture innan jag började en fångst för att endast visa trafik till och från min dator. Med detta filter försvann ARP-förfrågningarna men ARP-svaren fanns kvar. Jag kunde även fortfarande se mina kompletta ping-tester och TLS-messages.

Uppgift 2:

Denna uppgift bestod av att tömma ARP-tabellen och sedan fånga upp och analysera ett ping-test till routern. När jag gjorde detta såg jag att innan ARP och ICMP meddelanden syns annan nättrafik, troligen för att jag har en webbläsare igång. Bland annat står DNS-query responses, TLS protected payloads, och STP meddelanden.

Sedan såg jag att datorn började med att skicka en ARP Request broadcast meddelande, “Who has 192.168.1.2? Tell 192.168.1.1”. Routern svarade med en ARP Reply och angav sin MAC-adress. Efter ARP Reply skickades ett Echo Ping Request som togs emot av ett Echo Ping Reply.

Detta visar det tydliga samspelet mellan lager 2 och lager 3 i OSI-modellen i LAN nätverk, då utan en MAC-adress kan inte ett IP-adress levereras.

I hexdumpen av det sista ICMP-paketet i pingtestet markerade jag med hjälp av färgkodning de olika fälten som bygger ethernetpaketet enligt RFC 894.

Destination MAC | Source MAC | Type | Data

```
0000  bc ee 7b 8d 45 bf d8 d8 e5 04 b1 00 08 00 45 00   ..{.E.........E.
0010  00 3c 37 96 00 00 36 01 88 ef 01 01 01 01 c0 a8   .<7...6.........
0020  01 92 00 00 54 c0 00 01 00 9b 61 62 63 64 65 66   ....T.....abcdef
0030  67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76   ghijklmnopqrstuv
0040  77 61 62 63 64 65 66 67 68 69                     wabcdefghi
```

Uppgift 3:

Denna uppgift bestod av att försöka logga in på två olika webbplatser med användarnamn och lösenord, och sedan analysera paketen för att försöka ta reda på om lösenordet gick att avslöja. En av sidorna skulle vara okrypterad via HTTP och den andra krypterad via HTTPS.

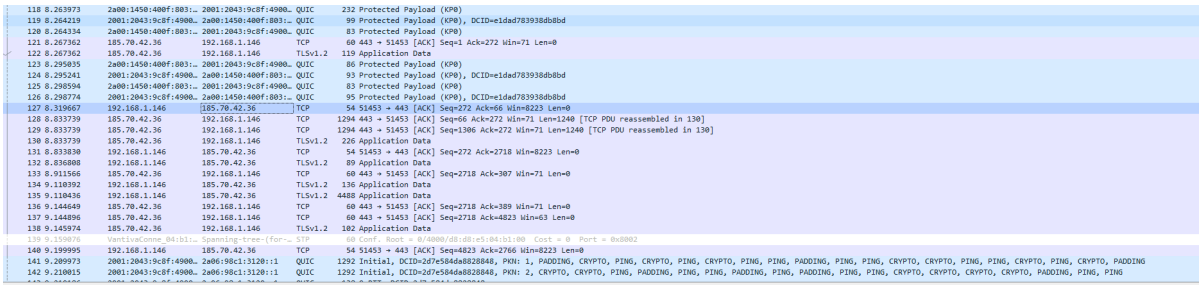
Enligt instruktionen skulle jag logga in på http://mail.imega.se med ett påhittat användarnamn och lösenord. Dessvärre var hemsidan inte tillgänglig så jag försökte skapa en egen enkel HTTP server på min lokala dator med IP-adress “192.168.1.146” och port

“8000”. Därefter skickade jag en post med “curl -v -d "user=username&pass=helloworld" http://192.168.1.146:8000” via kommandotolken.

I utförandet fick jag problem med portar där trafiken hamnade på port 8009 istället för 8000. På grund av detta kunde jag inte exakt utföra det ursprungliga HTTP-testet enligt instruktion.

För den krypterade inloggningen instruerades jag att logga in på https://webmail.umu.se med mitt umu-konto, men den hemsidan var också otillgänglig. Istället valde jag att göra ett test mot https://account.proton.me/mail eftersom inloggningsformuläret där tillät mig att skriva e-post och lösenord i samma fönster, vilket andra hemsidor såsom Gmail och Yahoo inte tillät mig. För testet använde jag påhittade inloggningsuppgifter som var,
user=test@proton.me
pass=testtest

När jag tryckte på logga in fångades TLS och QUIC trafik, som kan ses i figur 1.4. Efter QUIC Handshake visades endast Protected Payload. POST-datan såsom användarnamnet och lösenordet var krypterat och gick inte att avläsa genom att försöka följa TCP-strömmen.



Figur 1.4: Den fångade datatrafiken vid ögonblicket där inloggningsförsöket utfördes på https://account.proton.me/mail

Slutord

Denna laborationen lät mig analysera nättrafiken i realtid och målade upp en klarare bild av hur olika protokoll samverkar i OSI-modellen. Övningarna visade skillnaden mellan teori och praktik och gav mig en förståelse för varför protokoll såsom HTTPS, ARP och TCP/IP är så viktiga i dagens nätverk.