

# DATORKOMMUNIKATION

## Laboration 2 Nät

—

## Protokoll för Internet och Ethernet

### Laborationens förhållande till teorin

Under laborationen **Nät – Protokoll för Internet och Ethernet** används paketsniffern Wireshark för att undersöka några utvalda protokoll. Dessa protokoll används för Internet- och LAN-kommunikation.

### Redovisning

Laborationen **Nät – Protokoll för Internet och Ethernet** redovisas med en skriftlig rapport. I denna rapport ska du beskriva laborationen och redovisa dina svar. Använd kursens rapportmall. Rapporten ska bestå av följande delar:

1. Titelsida  
Fyll i rubrik (laborationens titel) och personuppgifter. Gör en kort sammanfattning av laborationen.
2. Innehållsförteckning  
(Höger-klicka på innehållsförteckningen för att uppdatera.)
3. Bakgrund  
Laborationens koppling till datorkommunikation och Wireshark
4. Resultat  
Uppgift 1: beskrivning av uppgiften och svar på uppgifter  
Uppgift 2: beskrivning av uppgiften och svar på uppgifter  
Uppgift 3: beskrivning av uppgiften och svar på uppgifter

# Nät – Protokoll för Internet och Ethernet

## Syfte

Under denna laboration kommer du att få se hur protokoll används för kommunikation och undersöka några detaljer i sådana. Pedagogiken kan enkelt beskrivas med ett kinesiskt ordspråk: *Berätta för mig och jag ska glömma. Visa mig och jag kommer att komma ihåg. Gör mig delaktig och jag kommer att lära mig.*

## Protokoll

För att kommunikationen ska fungera i datanät, krävs speciella program och hårdvaror. Sådana beskrivs av **protokoll**. Ett protokoll är helt enkelt en beskrivning över hur något ska fungera. Protokoll skrivs som bekant under möten för att man ska komma ihåg de beslut som har fattats, exempelvis under ett föreningsmöte. I detta fall är det inte möten som dokumenteras. Istället är det beslut om hur datautrustning ska fungera under kommunikation som skrivs ned. I många fall utfärdas protokoll som standarder (rekommendationer). Fördelen med att utfärda protokoll som standarder är att programmerarna och hårdvarutillverkarna får vetskap om hur produkter ska fungera för att passa ihop med annan utrustning. För oss konsumenter innebär detta att vi kan köpa datautrustning som passar för sitt ändamål. Annars är detta inte givet. Utan standarder skulle det kunna finnas lika många protokoll för ett ändamål som det finns tillverkare.

De protokoll som är generella brukar kallas öppna. Man talar om **öppna system** (publika) som innebär att oavsett tillverkare av program och hårdvara så fungerar produkten i sitt sammanhang. Några exempel är protokollen för **Internet**, det lokala nätet **Ethernet** och persondatorns **Universal Serial Bus** (USB).

Motsatsen är **slutna system** (tillverkarspecifika system, plug-compatible systems). Det finns en rad nät som är slutna system. För att kunna använda sådana, krävs att man har utrustning som just den aktuella tillverkaren har tagit fram. Några exempel är bl.a. System Network Architecture (SNA) från IBM, Distributed Network Architecture (DNA) från Digital Equipment, Distributed Communication Architecture (DCA) från Univac och Distributed System Architecture (DSA) från Honeywell Bull.

- Protokoll är i detta fall beskrivningar över hur kommunikation ska fungera. Alla datorer (parter i samtal) måste "tala samma språk".
- Protokoll kan skrivas ned i olika former: klartext, tillståndsgraf, tillståndstabell, pseudoprogram (program som är skrivna med klartext) och program som är skrivna med något programspråk.
- Protokollen realiseras (förverkligas) oftast i form av program men även hårdvaror förekommer (i synnerhet på fysisk nivå).

## Wireshark

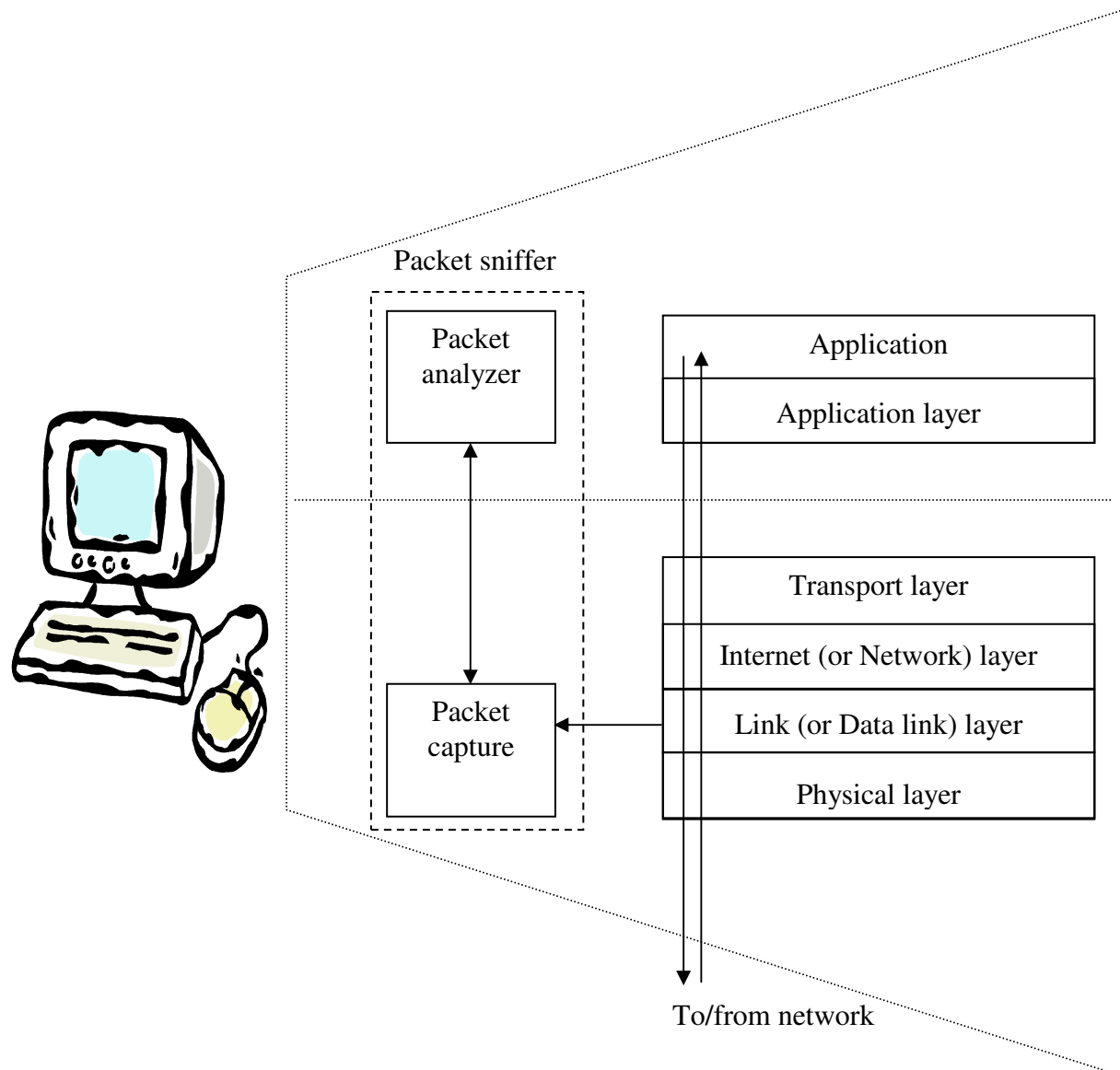
Programmet för analys av kommunikerande protokoll, paketsniffen, som ska användas kallas **Wireshark**. Från början kallades paketsniffen **Ethereal**. Programmet skapades av Gerald Combs som fick sin utbildning vid University of Missouri-Kansas City. Det var 1998 som Ethereal släpptes. Gerald arbetade först vid Network Integration Services (NIS), men under maj månad 2006 flyttade han över till CACE Technologies. Under juni månad samma år skapades **Wireshark network protocol analyzer** från Ethereal-koden. Anledningen var helt enkelt att varumärket Ethereal® inte fick användas av CACE Technologies. Sedan dess är det paketsniffen Wireshark som underhålls med uppdateringar. Gerald ska inte betraktas som den enda programmeraren av Wireshark eftersom över 500 personer har bidragit med kod. Hans roll är numera att vara huvudprogrammerare.

Wireshark har givits ut under GNU General Public License och kan köras i flertalet operativsystem: Windows, Unix och Unix-liknande system som exempelvis Linux, Solaris, FreeBSD, NetBSD, OpenBSD samt Mac OS X. (I Mac OS X behöver användaren köra en X Server som exempelvis X11.app.) Sniffen kan tas hem (gratis) från Wiresharks hemsida: <http://www.wireshark.org>

Principen för en paketsniffer visas i figur 1. Alla ramar till/från datanätet kopieras till **packet capture** (pcap). Ramarna kopieras från något **capture interface** som är en nätadapter med tillhörande drivrutiner. Nätadaptern behöver nödvändigtvis inte vara ett nätverkskort i datorn, det kan lika gärna vara en nätadapter som exempelvis ansluts via en serieport eller USB-port. Nätadaptern kan vara tillverkad för exempelvis Ethernet, Token-Ring, FDDI, seriell kommunikation (MODEM för protokollen PPP och SLIP), IEEE 802.11 (WLAN) eller ATM.

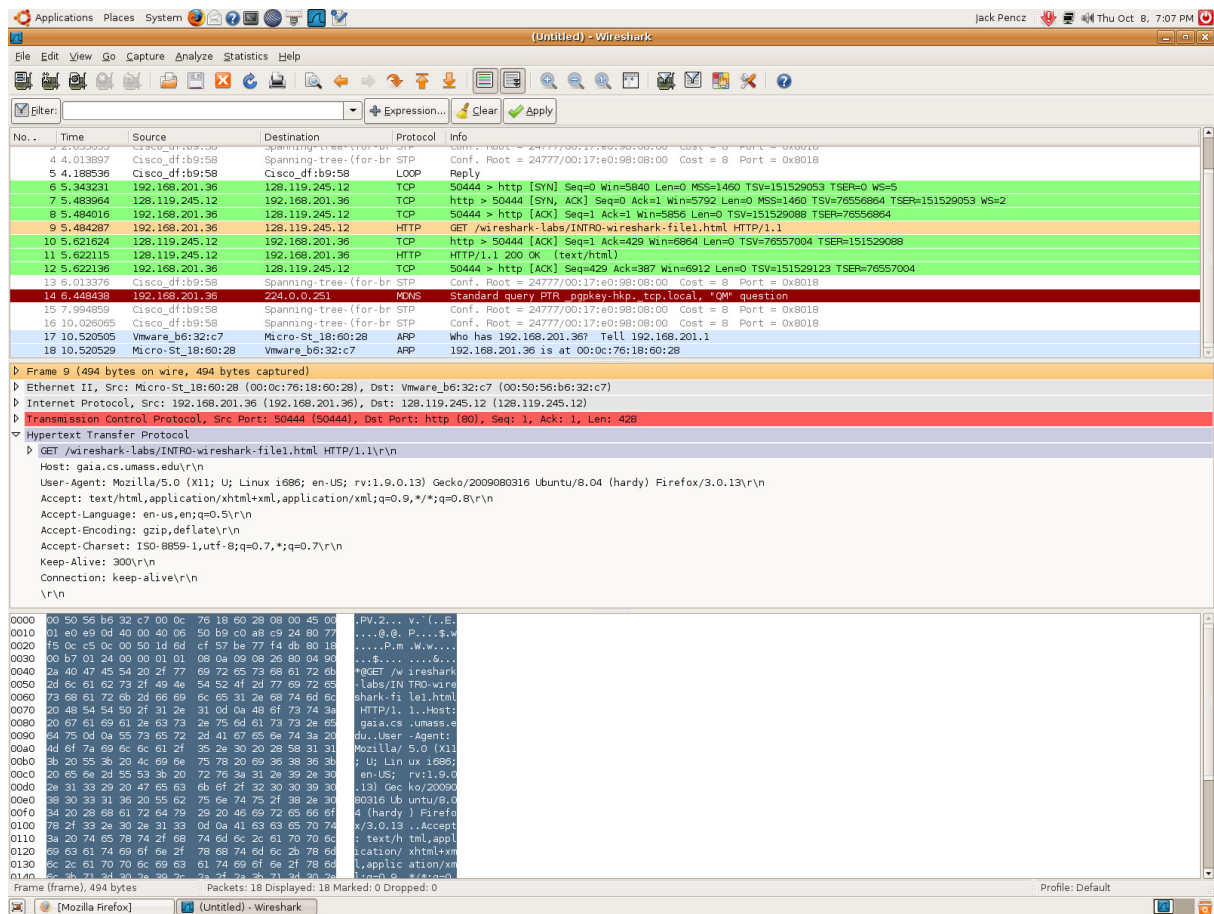
Den andra komponenten i en paketsniffer är **packet analyzer**. Denna komponent visar fältens innehåll som finns i ett protokollmeddelande. Det är nödvändigtvis inte hela ramar som visas utan innehållet i vissa delar av dessa. Man exempelvis studera IP-paketets innehåll som kan vara del av en inkommande Ethernetram. IP-paketet är relativt stort eftersom det i sin tur innehåller andra protokollmeddelanden. Nästa steg kan vara att studera ett TCP-segments innehåll och i detta ett HTTP-meddelande. HTTP-meddelandet finns i TCP-segmentet som i sin tur återfinns i IP-paketet. Allt detta överfördes i en inkommande Ethernetram. Detta enkla exempel kan ge förståelse för det stora antalet programmerare har bidragit med kod till Wireshark.

Vi kan placera in nätverkskortet och protokollen från ovanstående exempel i figur 1. I det fysiska skiktet placeras Ethernetkortet med tillhörande drivrutin. För att Ethernetkortet ska fungera behövs också program för hela MAC-delen. (MAC är akronymen Media Access Control.) Den delen går upp i länkskiktet och ger pcap sina ramkopior. IP hör naturligtvis hemma i Internetskiktet. TCP är inte heller svårt att placera – det hör till transportskiktet. Överst i protokollstacken läggs HTTP i applikationsskiktet. Protokoll i detta skikt brukar integreras med respektive applikation. I detta fall kan applikationen exempelvis vara en webbläsare (*browser*).



Figur 1. Paketsniffer.

Wireshark är försett med ett grafiskt användargränssnitt (GUI). Ett exempel på körning av paketsniffern visas i figur 2.



Figur 2. Körexempel.

GUI är uppbyggt av komponenter på ett traditionellt sätt. Det är fråga om menyer och fönster. Problemet kan till att börja med vara att förstå avsikten med de olika fönstren i detta GUI. Därför ges här en kort förklaring av samtliga komponenter:

- Kommandomenyer

Kommando ges genom att välja alternativ i någon av menyerna som är placerade längst upp i huvudfönstret. Se figur 2. Till att börja med är menyn **Capture** av intresse. Den är viktig för där styrs datafångsten: val av nätadapter (*interface*), start, stopp, m.m.

Det finns också ett verktygsfält under menyerna.

- Visningsfilter

Under menyerna och tillhörande verktygsfält finns plats för gällande visningsfilter. (Det finns två typer av filter i Wireshark, visningsfilter och paketinfångningsfilter.) Ett visningsfilter kan skrivas in manuellt eller väljas från **Analyze** → **Display Filters**. Med visningsfiltret (**filter string**) bestäms vilka/vilket protokoll som ska visas, exempelvis **http**, eller inte ska visas, exempelvis **not arp**, i **listningsfönstret**. Visningsfiltret är inte aktiverat i figur 2. Det går bra att lägga till filter som behövs, exempelvis att enbart **arp** ska visas.

- Fönstret för paketlistning – **listningsfönstret**

Detta fönster visar en sammanfattning per paket och rad. Observera att de nummer som visas ges av Wireshark, dvs. numren finns inte som information inne i respektive paket. Se figur 2 som visar paket nr. 4-18. Dessutom visas tiden för infångning, avsändar- och mottagaradress, protokolltyp och protokollspecifik information i varje paket.

Genom att klicka på en kolumnrubrik (*No.*, *Source*, *Destination*, *Protocol*, *Info*) sorteras innehållet i fönstret av vald kategori.

Under övningarna i T002 kommer Wireshark att visa flera paket som bl.a. har med **Address Resolution Protocol** (ARP) och **Spanning-Tree Protocol** (STP) att göra. ARP ger adressupplösning (näadress → MAC-adress, speciellt i T002 gäller det IP-adress → Ethernetadress) och STP spänner upp ett logiskt kommunikationsträd för att eliminera alternativa vägar och därmed flera ramkopior. Det är standard-gateway som skickar många av dessa paket. (Standard-gateway är en router, men den är också försedd med brandvägg, DHCP-server och switch. DHCP är akronymen för **Dynamic Host Configuration Protocol** som är protokollet för utlåning av privata IP-adresser.)

Under övningarna kan det verka störande att Wireshark visar paket som har med det lokala nätets administration att göra. För att inte visa flertalet sådana, rekommenderas att paketinfångningen filtreras och/eller visningen begränsas. Filtrering/visning hanteras från menyerna **Capture** (paketinfångning) och **Analyze** (visning). Enklast är att skriva in filtersträng direkt i **Filter**-fältet under listningsfönstret, exempelvis **http**, **tcp** eller **arp**. (Klicka ett par gånger på [Enter] så att filtersträngen verkligen blir aktiverad.) Det går också att stänga av (stoppa) efter de önskade paketen. Start/stopp sköts från menyn **Capture**.

- Fönstret för detaljer i pakethuvuden – **detaljfönstret**

För att få fram detaljerad information i ett huvud, klicka på önskat paket i **listningsfönstret**. I figur 2 är paket nr. 9 markerat. Detta paket har skickats av HTTP som har använt **GET-metoden**. Även de bärande paketen kommer att visas, dvs. de paket som har kapslat in HTTP-paketet. I det markerade exemplet är de bärande paketen i tur och ordning följande: **TCP**, **IP** och **Ethernet II**.

Informationsmängden kan utökas/minskas genom att klicka på ▸/▾. För det markerade exemplet anges bl.a. att HTTP har använt **GET**-metoden, att underkatalogen på aktuell webbplats kallas **wiresharks-labs**, att den fil som kommer att visas kallas **INTRO-wireshark-file1.html**, och att HTTP-versionen är **1.1**.

- Fönstret för paketinnehåll – **innehållsfönstret**

Längst ned i huvudfönstret finns fönstret för paketinnehåll. Här visas allt som finns i det infångade paketet, både i formatet ASCII och hexadecimalt. Genom att markera tecken i det ena av de två fönstren (**detaljfönstret** eller **innehållsfönstret**) visas motsvarande markering i det andra.

## Uppgifter

Under övningarna kommer du att studera följande protokoll:

1. HTTP
2. Ethernet
3. ARP

Uppgifterna är inspirerade av Wireshark-laborationerna som finns på webbstödet för kursboken *Computer Networking – A Top-Down Approach*, av James Kurose och Keith Ross.

Då Wireshark startas är den första åtgärden att **välja nätadapter**. Vanligtvis används någon typ av Ethernetkort (**eth0** eller **eth1**) i datorerna. Nätadapter väljs i **Capture Options**. Bocka för val av nätadapter och klicka på **Start**. I datorlab T002 sänds många nätverksadministrerande paket. Dessa bör av praktiska skäl inte visas i **listningsfönstret**. Därför anges **rekommenderat visningsfilter** i början av varje övningsdel.

Ibland kan det vara lämpligt att göra **restart** av **listningsfönstret**. Detta görs med **Capture → Restart**. I samma meny finns möjlighet till **Start** respektive **Stop**.

Utöver nätadministrativa paket skickas också broadcasts i T002. **Därför bör rekommenderat visningsfilter används i respektive övning**. I annat fall kommer Wireshark att visas många paket som inte har med övningarna att göra.

*Trevlig laboration!*

# 1 HTTP

## 1.1 Inledning

I denna del av övningarna ska **HyperText Transfer Protocol** (HTTP) studeras. Protokollet används för att överföra information på **World Wide Web** (WWW). Det är alltså fråga om dokument som i grunden byggs upp av **HyperText Markup Language** (HTML) och/eller anpassade skriftspråk som exempelvis **JavaScript**, **VBScript** och **PHP: Hypertext Preprocessor**. HTML-sidorna kan innehålla flertalet objekt. Det är i grunden fråga om text och bild, men även andra typer av objekt kan förekomma som exempelvis av typen **Java-applet** och **VB ActiveX Controller**.

Transportprotokellet som används ihop med HTTP är **Transmission Control Protocol** (TCP). Detta transportprotokoll ger säker överföring med hjälp av bl.a. upp- och nedkoppling samt kvittering. Därför är det av intresse att i samband med HTTP också se hur TCP fungerar.

HTTP omfattar åtta metoder som också kallas *verbs*. En metod bestämmer den typ av aktion som ett HTTP-meddelande ska medföra hos mottagaren. De åtta metoderna är HEAD, GET, POST, PUT, DELETE, TRACE, OPTIONS och CONNECT.

GET-metoden undersöks relativt tidigt under övningarna. Den används för att hämta en bestämd resurs, exempelvis en webbsida som kan visas på användarens bildskärm. Av de åtta metoderna är GET den mest använda.

Vidare ska följande studeras: HTTP-meddelandets format (kallas HTTP-paket i Wireshark), mottagning av stora HTML-filer, mottagning av HTML-filer med inbäddade objekt och autentisering med HTTP.



## 1.2 Upp- och nedkoppling med TCP

Upp- och nedkoppling med TCP ska analyseras med hjälp av Wireshark. Eftersom kommunikationen med klienter i T002 av säkerhetsskäl måste gå via proxyservern (*web cache*) på 130.243.96.54, kan det vara svårt att identifiera upp- och nedkopplingar med TCP. Webbläsaren och proxyservern strävar nämligen efter att hålla TCP-förbindelsen uppe genom att skicka **TCP Keep-Alive** till varandra. I normala fall görs uppkoppling i samband med att något ska skickas med TCP, men i T002 görs inte alltid sådana, då det redan finns en uppkopplad TCP-förbindelse. (Proxyservern måste givetvis göra en TCP-uppkoppling innan varje HTTP-begäran kan skickas vidare. Det är endast internt mellan klienter i T002 och proxyservern som det redan finns uppkopplade TCP-förbindelser.) Webbläsaren och proxyservern kopplar ihop sig då webbläsaren startas.

Starta webbläsaren av typen **Mozilla Firefox** (MF). Rensa webbläsarens cache med **History** → **Clear Private Data**, men låt **Cookies** vara kvar i cachen så att du inte behöver svara på frågor om dessa – det kan störa övningen. (Däremot måste cachen ovillkorligen tömmas, men låt övriga inställningar vara kvar.) Avsluta MF.

Starta Wireshark. Välj nätadapter (**eth0** eller **eth1**) under **Capture Options** och klicka sedan på **Start**. Paketinfångningen ska nu påbörjas automatiskt. Det rekommenderade visningsfiltret är **tcp**. Skriv detta under **Filter**-fältet och tryck ett par gånger på [Enter]. Starta om paketinfångningen (utan att spara föregående fångst).

Starta MF och vänta tills alla objekt visas för startsidan. Stäng sedan MF. Gå tillbaka till Wireshark och gör omedelbart **Capture** → **Stop**.

Om det visas alltför många paket i **listningsfönstret**, kontrollera filtret och starta på nytt. På frågan om "Save capture file...", svara **continue without saving**. (Ge samma svar under alla övningar.)

Beskriv enligt följande uppgifter:

1. A. Skrolla tills du kommer till början av paketlistningen. Där bör den inledande TCP-uppkopplingen mellan webbläsaren 192.168.201.x och proxyservern 130.243.96.154 finnas. Det skickas TCP-segment med flaggorna [SYN], [SYN, ACK] och [ACK]. (Möjligtvis visas det många uppkopplingar. Det är den allra första som ska redovisas.)

### Flaggor i TCP-segment

**SYN** = Synchronization, begära om uppkoppling

**ACK** = Acknowledge, positiv kvittering av mottagna bytes

**FIN** = Finnish, begäran om nedkoppling

Beskriv sekvensen av TCP-segment (satta flaggor och riktning) för **uppkoppling** (3-vägs handskakning). Rita helst en figur för kommunikationen mellan webbläsaren och proxyservern.

Du gör detta relativt enkelt genom att rita tidsdiagram över de flaggor som är satta i segmenten och anger respektive trafikriktning (från webbläsaren och till webbservern, respektive från webbserver och till webbläsaren). (Rita ungefär som i figur 3.40, **Closing a TCP connection**, i kursboken, upplaga 6. Men, kom ihåg att vara noga med

vem av klienten eller servern som begär upp- respektive nedkoppling.)

B. Skrolla tills du kommer ned till slutet av paketlistningen. Det skickas TCP-segment med flaggorna [FIN, ACK], [ACK], [FIN, ACK] och [ACK]. (Möjligtvis visas det flera nedkopplingar. Försök hitta någon som är så komplett som möjligt.)

Beskriv sekvensen av TCP-segment (flaggor och riktning) för **nedkoppling** (4-vägs handskakning). Rita helst en figur för kommunikationen mellan webbläsaren och proxyservern.

### 1.3 GET-metoden

Gör följande:

Byt visningsfilter till **http**. Starta om paketinfångningen.

Surfa till <http://basen.oru.se/datorkom/enkel.html>

Stoppa paketinfångningen.

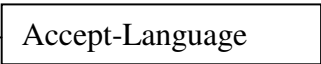
De två paketen av intresse är en begäran (HTTP *request* med GET-metoden) och ett svar (HTTP *response*). Markera den första HTTP-begäran som du bedömer har med övningen att göra.

I **detaljfenstret** visas flera protokoll för varje HTTP-paket. I detta fall ska man kunna se att följande packningsordning: HTTP → TCP → IP → Ethernet II. Om endast HTTP ska studeras, bör du se till att Ethernet II, TCP och IP är kollapsade (stängda, ▸) och att endast HTTP är expanderat (öppnat, ▾). Se figur 2.

Besvara följande frågor:

2. A. Vilken HTTP-version (1.0 eller 1.1) använder webbläsaren?  
B. Vilken HTTP-version (1.0 eller 1.1) använder den svarande webbservern?
3. Vilket språk accepteras enligt HTTP-begäran? Se figur 3 för att få tips om var detta kan avläsas i Wireshark. Tecknen `\r\n` betyder *Carriage Return* (CR eller r) respektive *Line Feed/New Line* (LF eller n) och används för radbrytning i bl.a. HTTP-huvudena.

```
GET /start____2232.aspx HTTP/1.1\r\n
Accept: */*\r\n
Accept-Language: sv\r\n
UA-CPU: x86\r\n
-----: -----\r\n
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322)\r\n
Host: www.oru.se\r\n
Connection: Keep-Alive\r\n
Cookie: Vizzit=PLPDZ6/H6UyC0Opfu4Y/4g==:1169232181;ASP.NET_SessionId=2pad45;__utma=26;
__utmz=21.1.1.utmccn=(direct)|utmcsr=(direct)|utmcmd=(none);
\r\n
```



Figur 3. Exempel på HTTP-huvud för begäran.

4. A. Vilken är avsändarens IP-adress för HTTP-begäran?  
B. Vilken är mottagarens IP-adress för HTTP-begäran? (Mottagaren sett inifrån T002-nätet är en proxyserver. Det är denna IP-adress som visas, inte IP-adressen till basen.oru.se.)

5. A. Vilken *status code* översänder webbservern i HTTP-svaret?
- B. Vilken *phrase* översänder webbservern i HTTP-svaret?

Se figur 4 för att få tips om var detta kan avläsas i Wireshark.

```
Hypertext Transfer Protocol
HTTP/1.1 200 OK\r\n
Date: Thu, 18 Oct 2007 15:47:06 GMT\r\n
Server: Microsoft-IIS/6.0\r\n
X-Powered-By: ASP.NET\r\n
X-AspNet-Version: 2.0.50727\r\n
Cache-Control: private\r\n
Expires: Thu, 18 Oct 2007 15:47:06 GMT\r\n
Content-Type: text/html; charset=iso-8859-1\r\n
Content-Length: 44990
\r\n
Line-based text data: text/html
```

Status line: version status code phrase [CR] [LF]

Figur 4. Exempel på HTTP-huvud för svar.

6. A. När sändes HTML-filen från webbservern och vilken tidszon anges?
- B. När blev HTML-filen förändrad senast?
7. Hur stort är innehållet i svaret räknat i antal byte?
8. Skickades något mer än HTML-filen till webbläsaren, exempelvis en favicon.ico?

*En **favicon** (favorites icon) är en ikon som representerar en webbsida. Den visas vanligen till vänster i adressfältet i de flesta moderna webbläsare när du är inne på sidan och kan även visas i menyer och listor om du väljer att lagra sidan som bokmärke. I många flikbaserade webbläsare visas även den ikonen i motsvarande flik i flikfältet. (Wikipedia: Favicon)*

9. Sök efter data i HTTP-svaret som visas i **innehållsfönstret** men **inte** i **detaljfönstret**:

Leta efter starttaggen <HTML> och tillhörande stopptaggen </HTML>. Mellan detta taggpar finns koden för allt som visas i webbläsarens fönster, dvs. webbsidan. (Försök hitta taggparet – ingen fråga att besvara om detta.)

## 1.4 Villkorlig GET-metod


De flesta webbläsarna ”cachar” (lagrar) webbsidornas objekt och använder därför **den villkorliga GET-metoden** för att hämta objekt.

Gör följande:

Rensa webbläsarens cache.

Behåll visningsfiltret **http**. Starta om paketinfångningen.

Surfa till <http://basen.oru.se/datorkom/tomten.html>

Gör omgående en uppdatering av webbsidan. (I MF klickar man på  eller trycker på [F5].)

Avbryt (**Stop**) paketinfångningen med Wireshark.

Besvara följande frågor:

10. Undersök den första HTTP GET-begäran från din webbläsare till webbservern. Kan du se villkoret **If-Modified-Since** som är en *header line* i HTTP-huvudet?

Om inte, varför det?

11. A. Vilken *status code* översänder webbservern i det första HTTP-svara?

B. Vilken *phrase* översänder webbservern i det första HTTP-svara?

12. Undersök den andra HTTP GET-begäran från din webbläsare till webbservern. Skriv av raden för **If-Modified-Since**.

13. A. Vilken *status code* översänder webbservern i det andra HTTP-svaret?

B. Vilken *phrase* översänder webbservern i det andra HTTP-svaret?

## 1.5 Långa HTML-filer

Hittills har undersökningar gjorts på korta HTML-filer (HTML-dokument). Därför kan det vara av intresse att undersöka nedladdning av en längre fil.

Gör följande:

Rensa webbläsarens cache.

Behåll visningsfiltret **http**. Starta om paketinfångningen.

Surfa till <http://basen.oru.se/datorkom/dataterm.html>

Avbryt (**Stop**) paketinfångningen med Wireshark.

Besvara följande frågor:

14. A. Hur många HTTP GET-begäran skickade din webbläsare för att få ihop en komplett webbsida med dataterm.html som grund?

Tips: Finns det bilder på webbsidan som kan orsaka fler objekt förutom HTML-koden?

B. Vilka komprimeringsmetoder accepteras i svaret, enligt begäran?

15. A. Ändra visningsfiltret till **tcp**. Hur många TCP-segment behövdes för att överföra HTML-filen?

Räkna från det första TCP-segmentet (protocol = TCP) i svaret (**direkt efter** HTTP GET-begäran) till och med det sista (HTTP/1.1 200 OK). Räkna inte in kvittenserna (ACK:arna) från webbläsaren. Räkna endast de som Wireshark har märkt med **[TCP segment of a reassembled PDU]** och det sista segmentet som innehåller HTTP-huvudet (protocol = HTTP). (PDU är förkortningen för *Protocol Data Unit*.)

Om du får < 2 antal TCP-segment inklusive det sista med HTTP-huvudet, säg till läraren. (Nätadaptern kan vara felinställd.)

Hur stort är innehållet i svaret räknat i antal byte? Ge svar på B, C och D.

B. Originalstorlek (utan komprimering)

C. Om komprimering används, ange komprimerad storlek.

D. Vilken komprimeringsmetod användes i sådant fall?

## 1.6 HTML-filer med inbäddade objekt

Du ska undersöka överföring av mer än ett enda objekt för en och samma webbsida. Tidigare har endast ett textobjekt per HTML-meddelande överförts.

Gör följande:

Rensa webbläsarens cache.

Använd visningsfiltret **http**. Starta om paketinfångningen.

Surfa till <http://basen.oru.se/datorkom/japan.html>

Avbryt (**Stop**) paketinfångningen med Wireshark.

Besvara följande frågor:

16. A. Hur många HTTP GET-begäran för **japan.html** inklusive innehållande objekt skickades från webbläsaren till webbservern?  
  
B. Till vilken adress (komplett URL) skickades för webbsidan respektive objektet?
17. A. Skriv av den *header line* i HTTP-svaret som anger bildtypen.  
  
B. Ändra visningsfiltret till **tcp**. Hur många TCP-segment behövdes för att överföra bilden? (Se 15 angående bestämmande av antal TCP-segment.)
18. A. Vilken HTTP-version användes för att begära (med GET) **japan.html** och innehållande objekt?  
  
B. Används flyktig (*non-persistent*) eller varaktig (*persistent*) TCP-koppling för den versionen, enligt svaret till A? (Ange de initiala inställningarna, *default*.)  
  
C. Genom att söka efter eventuell nedkoppling följt av ny uppkoppling innan HTTP GET-begäran av bilden kan typ av TCP-koppling bestämmas. Gjordes nedkoppling följt av ny uppkoppling innan bilden överfördes? (T002-nätet gör analysen komplicerad eftersom en proxyserver används, men vi kan anta att det vi ser eller inte ser beträffande kopplingar, stämmer för överföring av bilden.) Vilken typ av TCP-koppling användes alltså i praktiken?

## 1.7 Autentisering med HTTP

Till sist återstår att undersöka webbsidor, och hela webbplatser, som kräver inloggning för access.

Gör följande:

Rensa webbläsarens cache.

Behåll visningsfiltret **http**. Starta om paketinfångningen.

Surfa till <http://dkn.comli.com/auktorisering.php>

Webbläsaren kommer att visa ett inloggningsfönster.

Logga in med...

Användarnamn:      ole

Lösenord:            pizza

Avbryt (**Stop**) paketinfångningen med Wireshark.

Besvara följande frågor:

19. Vilken *status code* och *phrase* översändes av webbservern i det första HTTP-svaret?
20. Webbläsaren får inte tillbaka någon webbsida i det första HTTP-svaret. När du har skrivit in användarnamn och lösenord, då skickas en ny HTTP GET-begäran till samma adress.
  - A. Vilken typ av *header line* har tillkommit i den andra begäran?
  - B. Vilken typ av autentisering har gjorts? Det finns *basic* (utan kryptering, s.k. *plaintext*) och *digest* (med kryptering).
  - C. Expandera denna *header line*. Vilka kreditiv skickades med?



## 2 Ethernet

### 2.1 Inledning

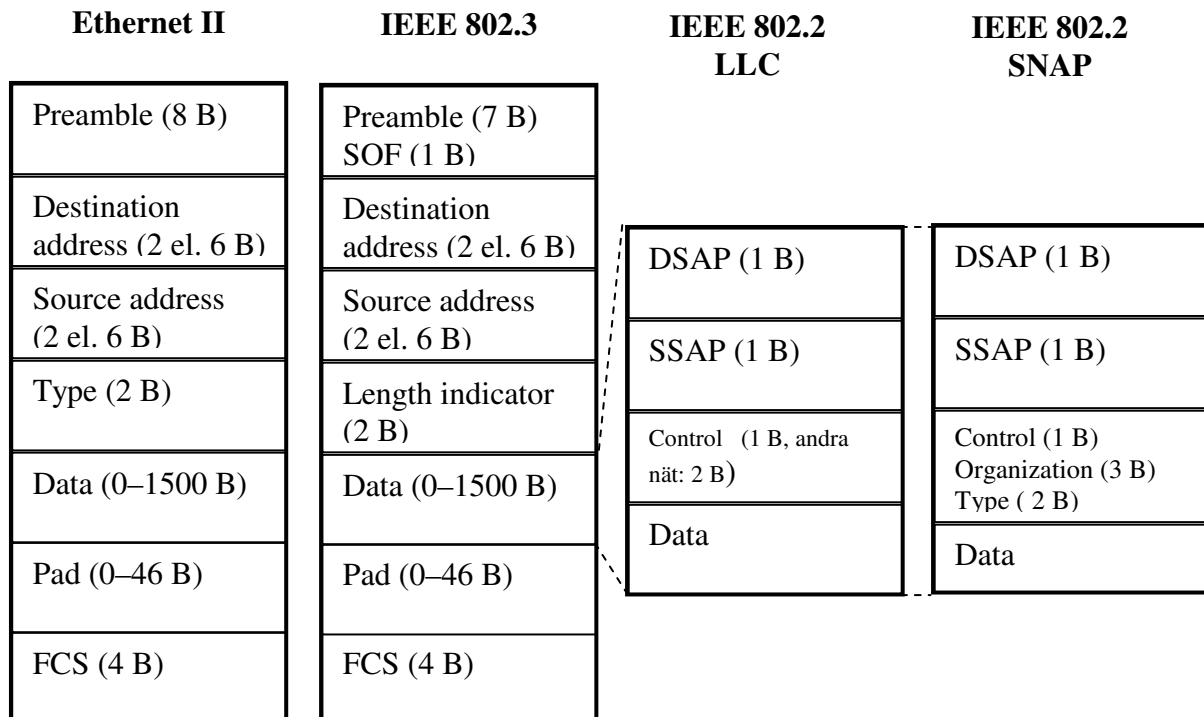
Bob Metcalfe och David Boggs är de som konstruerade Ethernet, ett av de lokala näten. (Nättypen kallas Local Area Network, LAN.) Redan under det tidiga 1970-talet arbetade Bob med ARPAnet vid MIT som doktorand från Harvard University. På så sätt var Bob välunderlättad om arbetet med Internet men det var det trådlösa ALOHAnet som senare inspirerade Bob och David till att påbörja arbetet med Ethernet. ALOHAnet utarbetades av Norm Abrahamson vid University of Hawaii. Norm och hans kollegor undersökte olika accessmetoder (ALOHA-familjen). Idén var att kunna kommunicera mellan Hawaiiis många öar med hjälp av radio på endast två kanaler. Bortsett från typiska radioproblem är det liknande problematik med trådlösa nät som med kabelbundna, i synnerhet för logiska/fysiska busstopologier. Norm hade gjort en noggrann utredning av detta.

Bob arbetade vid Xerox Parc (företagets forskningscentrum i Palo Alto) då experimenten med Ethernet påbörjades. Han var inte bara inspirerad av ALOHAnet utan också av persondatorerna från Alto Computers. Bob såg tydligen framför sig behovet av att kunna koppla ihop persondatorer i nät. Ingredienserna i arbetet med Ethernet var alltså **ARPAnet**, **ALOHAnet** och **persondatorer**. Egentligen var idén med nät för datorer inte ny. Principerna för ringnät hade presenterats av svensken Söderblom i ett patent på 1950-talet. Däremot dröjde det fram till hösten 1985 innan IBM lanserade sitt Token ring (IEEE 802.5). Det var nämligen utvecklingen av integrerade kretsar som medförde att det betydligt mer komplexa nätet Token ring kunde tillverkas på ett praktiskt sätt.

Den första versionen av Ethernet utvecklades redan under mitten av 1970-talet. Under år 1980 bildade Bob företaget 3Com som har blivit en av de största tillverkarna på marknaden av nätverkskort och sammankopplande enheter. Samma år (1980) publicerade Digital, Intel och Xerox (DIX) den första versionen av Ethernet. Redan 1983 kom den andra versionen av Ethernet från Digital. Den kallas Ethernet II och används av bl.a. DECnet och LAN för Internet-kommunikation. Då började det amerikanska ingenjörssförbundet IEEE att intressera sig för produkten. Detta ledde fram till standarden **IEEE 802.3**. IEEE (uttalas "I-tripple-E") är akronym för **Institutions of Electrical and Electronic Engineers**.

IEEE 802.3 skiljer sig något från Ethernet II beträffande ramformatet och dessutom omfattar standarden användande av **Logical Link Control (LLC)**. Ramformaten visas i figur 5. Observera att ramen för IEEE 802.3 omfattar fält för LLC, eller som alternativ **Data Link Control (DLC)** för IBM-baserade nät. Användandet av LLC ger som resultat att en LLC-ram placeras inne i IEEE-ramens info-fält.

International Organization for Standardization har en motsvarande standard som kallas **ISO 8802/3**.



Figur 5. Jämförelse av ramformat.

## 2.2 Ethernet II

Gör följande:

Rensa webbläsarens cache.

Använd inget visningsfilter. Starta om paketinfångningen.

Surfa till <http://basen.oru.se/datorkom/dataterm.html>


Avbryt (**Stop**) paketinfångningen med Wireshark.

Anteckna Wiresharks paketnummer för webbläsarens HTTP GET-**begäran** av ovanstående HTML-fil.

Anteckna Wiresharks paketnummer för det **första** TCP-segmentet i webbserverns HTTP-svar för ovanstående HTML-fil. Detta bör vara märkt med **[TCP segment of a reassembled PDU]** i **listningsfönstret**.

Anteckna Wiresharks paketnummer för det **sista** TCP-segmentet i webbserverns HTTP-svar för ovanstående HTML-fil. Detta bör innehålla kontrollinformationen **HTTP/1.1 200 OK** som visas i **listningsfönstret**.

Eftersom vi just nu är intresserade av protokoll på datalänknivån, vill vi inte visa protokollen ovanför. (Ethernetramen bär med sig ett IP-paket som i sin tur innehåller ett TCP-segment för del av ett HTTP-meddelande.) Ändra visningen genom välja *Enable Protocols* i **Analyze**-menyn och ta bort boken för **IPv4** i fönstret för **Enable Protocols**.

Markera paketet som innehåller webbläsarens HTTP GET-begäran i **listningsfönstret**. Du skrev ned Wiresharks paketnummer nyligen. Öppna Ethernetramen i **detaljfönstret**. (Klicka på )

Besvara följande frågor:

21. A. Vilken fullständig mottagaradress (12 hexadecimala tecken) har Ethernetramen? (Avläs mottagaradressen i paketet som innehåller HTTP GET-begäran.)

Detta är inte Ethernetadressen till **basen.oru.se** trots att paketet har denna IP-adress.

- B. Vilken värd eller enhet har denna mottagaradress (Ethernetadress)?

Tips: Alla värdar finns naturligtvis inte inne i vårt LAN. Dessutom behövs en speciell enhet för att komma till omvärlden. Vilken?

22. Vilken fullständig Ethernet-adress (MAC-adress, 12 hexadecimala tecken, hårdvaruadress) har din dator? (Avläs avsändaradressen i Ethernetramen som innehåller HTTP GET-begäran.)

23. Under kommunikationen användes en Ethernet II-ram vars format beskriv i figur 5.

A. Hur många byte omfattar *Type*-fältet?


B. Hur många hexadecimala tecken får plats?

C. Avläs och anteckna *Type* (hexadecimalt värde) i detaljerna som hör till Ethernet-ramen. Det gör man i Wiresharks **detaljfönster**. Om du markerar raden **Type** i denna visning, markeras också värdena i **innehållsfönstret**.

(Samma hexadecimala värde återfinns också i kolumnen *Protocol* i Wiresharks **listningsfönster**. Hexadecimala värden markeras med **0x** i kolumnen.)

D. Vilken typ av paket överfördes i praktiken av Ethernetramen, dvs. vad står det avlästa hexadecimala värdet för?

24. Sök efter början av HTTP-meddelandet i Ethernetramens datafält. Meddelandet börjar med "GET", dvs. metoden anges. Använd Wiresharks **innehållsfönster**. (Försök hitta metoden – ingen fråga att besvara om detta.)

25. Markera (i **listningsfönstret**) den sista Ethernetramen (som bär det sista TCP-segmentet) i webbserverns HTTP-svar. Du skrev tidigare ned detta Wireshark-paketnummer. Öppna Ethernetramen i **detaljfönstret**. (Klicka på ) Hur stort (antal byte) är datafältet i Ethernetramen?

Tips: Datastorleken visas i **detaljfönstret** då IP-protokollet inte är "enabled". Den beräknas som

*Frame length (som ges utan preamble och FCS) – [storleken på DA + SA + Type]*  
 $= x - 14$ ,

eller fås direkt från IP-paketets *Total length*. (IP-paketet är ju det som stoppas in i ramens datafält.)

26. A. Gå igenom alla Ethernetramar från den första till den sista för HTTP-svaret. Hur stort (antal byte) är det största datafältet? (Du antecknade tidigare Wiresharks paketnummer för det första respektive det sista TCP-segmentet i HTTP-svaret.)

Storleken fås som i den föregående uppgiften.

B. Se figur 5. Hur stora kan datafälten maximalt vara i Ethernetramar?

### 3 ARP

#### 3.1 Inledning

**Address Resolution Protocol** (ARP) används för att översätta från en värds IP-adress till motsvarande MAC-adress. MAC är förkortningen för **Media Access Control**. Vanligt förekommande protokoll för MAC gäller trådbundna eller trådlösa lokala nät. I T002 används ett lokalt nät av typen Ethernet, och närmare bestämt Ethernet II.

Mottagare kan anges med IP-adresser inom ett lokalt nät. Däremot används MAC-adresser för att hitta till mottagarnas nätverkskort. Därför behövs ARP. (Det finns också ett protokoll för motsatt uppslag som kallas RARP, där R = **Reversed**.)

#### 3.2 ARP för Ethernet II

Gör följande:

Rensa webbläsarens cache.

Det rekommenderade visningsfiltret är **arp**. Bocka för **IPv4** i fönstret för **Enable Protocols**. Starta om paketinfångningen.

Be läraren tömma ARP-cachen.

Surfa till <http://basen.oru.se/datorkom/dataterm.html>

Avbryt (**Stop**) paketinfångningen med Wireshark.

Besvara följande frågor:

27. A. Starta **Terminal** och lista de nuvarande posterna i ARP-cachen med kommandot **arp**.

Vilka poster finns det i ARP-cachen? Gör en tabell över samtliga ARP-poster som följer nedanstående exempel:

<b>IP-adress</b>	<b>MAC-address</b>	<b>Funktion i nätet</b>
<i>192.168.201.1</i>	<i>00:1B:78:E3:C8:DA</i>	<i>Standard-gateway</i>
<i>192.168.201.50</i>	<i>D8:D3:85:76:24:19</i>	<i>Värd</i>

B. Se efter i Wireshark vilka värdar som ställde ARP-frågor och vilka värdar som besvarade frågorna. OBS! Undersök endast ARP-frågor och svar som din värd blev inblandad i. Ett par exempel på hur du bör svara (frågeutbytet kan förstås variera):

*Från Standard-gateway till alla (broadcastadressen):*

*Who has 192.168.201.44? Tell 192.168.201.1*

*Svar från min värd till standard-gateway:*

*192.168.201.44 is at D8:D3:85:77:36:A0*

*Från min värd till standard-gateway:*

*Who has 192.168.201.1? Tell 192.168.201.44*

*Svar från standard-gateway till min värd:*

*192.168.201.1 is at 00:1B:78:E3:C8:DA*

(Broadcastadressen: FF:FF:FF:FF:FF:FF, den s.k. 1-adressen, enbart 1:or.)

28. Surfa till <http://www.erg.abdn.ac.uk/users/gorry/course/inet-pages/arp.html>

Se den färgglada figuren ungefär mitt på webbsidan som visar var ett ARP-meddelande läggs in i en Ethernetram? Jämför med figur 5 i detta laborations-PM. Vad kallas Ethernet-ramens fält som innehåller ett ARP-meddelande?

Tips: ARP- och RARP-meddelanden skiljer sig inte från andra data beträffande placeringen.

29. Undersök ett av dina ARP-meddelanden i Wireshark. Vilken hexadecimal kod anges som *Type* för ARP i Ethernet II-ramar?