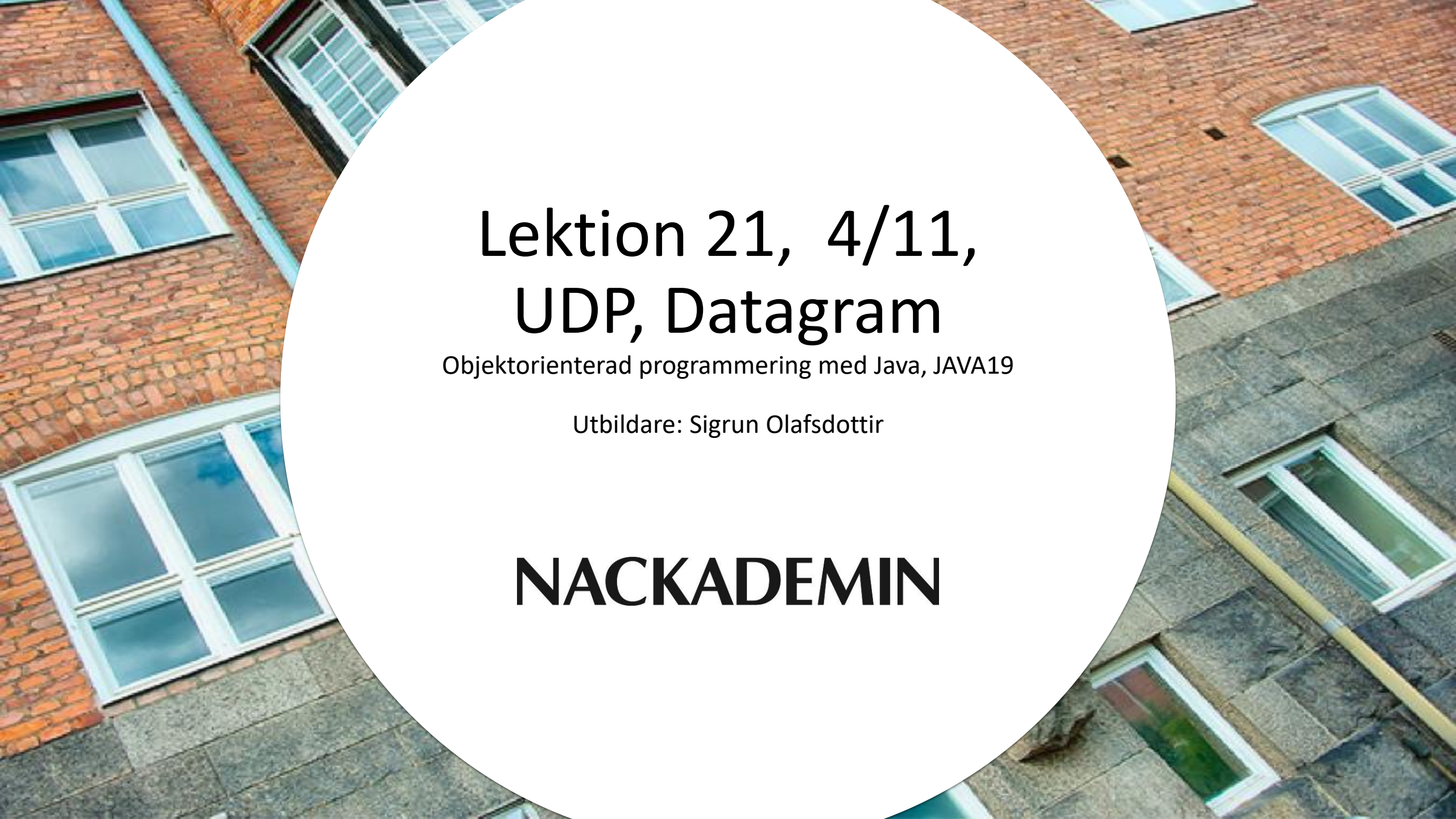


Lektion 21, 4/11, UDP, Datagram

Objektorienterad programmering med Java, JAVA19

Utbildare: Sigrun Olafsdottir

NACKADEMIN

Lektionstillfällets mål och metod

Mål med lektionen:

- Allmänbildning nätverk
- Datagram
- Multicast

Lektionens arbetsmetod/er:

- Sigrun pratar
- Eleverna gör övningsuppgift 1 och 2

NACKADEMIN

Plan Sprint 4

Lektion	Datum	Fokus	Aktivitet
21	4 nov	Intro Nätprogrammering, Datagram, Multicast	Varvat prat och övningsuppgifter hela dagen.
22	6 nov	Client-Server-arktiktur, Socketar, protokoll	Varvat prat och övningsuppgifter hela dagen.
23	7 nov	Client-server, multiuser	Varvat prat och övningsuppgifter hela dagen.
24	11 nov	Agilt arbete, uppstart grupparbete	FM: föreläsning, EM: Uppstart grupparbete
25	14 nov	URL:ar	FM: prat, EM: gruppuppgiften
26	15 nov	Properties, dragning bash/PowerShell	FM: prat, EM: gruppuppgiften

NACKADEMIN

Upplägget idag

- 9.15-9.30 Intro Sprint 4+retrospektiv
- 9.30-10.00 Genomgång Allmänt om nätverk och nätverksprotokoll
- 10.15-10.30 Övningsuppgift 1
- 10.30-11.00 Genomgång Datagram
- 11.15-11.45 Övningsuppgift 2a och 2b
- 11.45-12.00 Genomgång Övningsuppgift 2
- 12.00-13.00 Lunch
- 13.00-13.30 Genomgång Multicast
- 13.30-16.00 Övningsuppgift 3
- 13.30-16.00 I mån av tid: övningsuppgift 4a och 4b
- 15.00-16.00 Rester inlämningsuppgift 3

NACKADEMIN

Dags för retro

IP-nummer

- En dators adress på nätet
- IP står för Internet Protocol
- IP4: 192.168.1.49
  - Äldre adressformat
- IP6: 2001:0:4137:9e76:8ae:aa9c:6151:f737
  - Nyare adressformat
- It’s complicated
  - Din dator kan ha flera IP-adresser samtidigt som tilldelas dynamiskt beroende på vilket nätverk du befinner dig på
  - Innan vi labar, kolla vilket IP du har varje dag under sprinten, det ändrar sig
  - Kör ipconfig/ifconfig i kommandotolken eller det program vi kommer att bygga som Övn 1

NACKADEMIN

Ipconfig

Kommandotolken

```
^C
C:\Users\s_ola>ipconfig

Windows IP Configuration


Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::90dc:497b:5845:6b09%13
   IPv4 Address. . . . . . . . . . . : 192.168.1.49
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Tunnel adapter Teredo Tunneling Pseudo-Interface:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2001:0:4137:9e76:34fe:aa9c:6151:f737
   Link-local IPv6 Address . . . . . : fe80::34fe:aa9c:6151:f737%10
   Default Gateway . . . . . . . . . : ::
```

Ifconfig (Mac)

- På Mac lär man, i ett terminalfönster, kunna göra
  - ifconfig
  - ifconfig |grep inet (för att filtrera antalet rader som visas)

NACKADEMIN

Minilab 1

- Starta upp en kommandotolk (för Windows) eller en terminal (för mac)
- Kör ipconfig eller ifconfig
- Notera resultatet

Localhost

- IP-nummer 127.0.0.1 och datornamnet ”localhost” är reserverat
- Pekar alltid på den egna datorn
- Mycket praktiskt när man utvecklar nätverksapplikationer och ännu inte är redo att låta flera datorer kommunicera
- Man kan leka nätverk för sig själv på ett autentiskt sätt genom att låta apparna kommunicerar via 127.0.0.1

NACKADEMIN

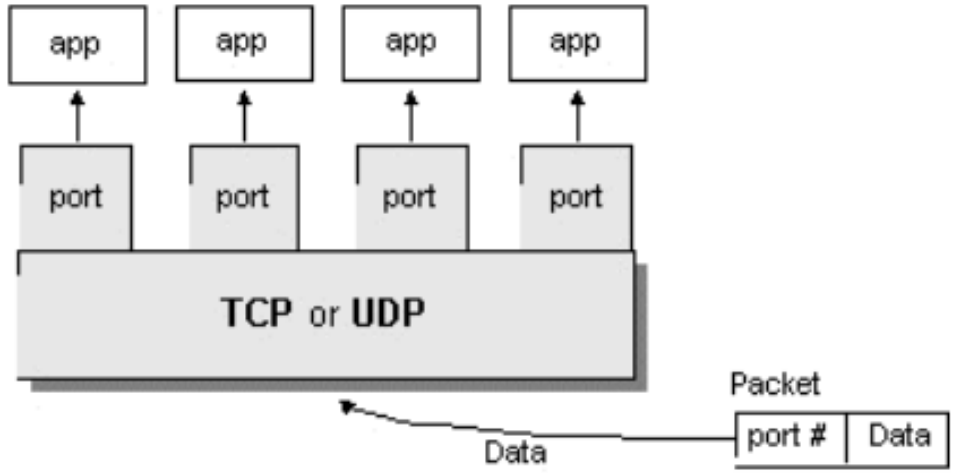
Portar

- ALL information från nätverket utifrån kommer in på ett ställe i datorn
- För att datorn ska veta vilket program som ska få vilken information används portnummer.
- Data som sänds över nätet är märkt med
  - IP-nummer som berättar till vilken dator informationen ska till
  - Portnummer som indikerar till vilken applikation datat ska till
- När vi bygger nätverksapplikationer anger vi IP-nummer och portnummer (ofta i konstruktorn till våra socketar) för att sändare och mottagare ska kunna hitta varandra.

NACKADEMIN

Portar, forts

- Nummer 0-65535 (16 bitar)
- 0-1024 är reserverade för olika standardservicar
- Använd 1025 och uppåt för egna applikationer



NACKADEMIN

Netstat -a

Kommandotolken

```
C:\Users\s_ola>netstat -a

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            LAPTOP-JHO8EIUR:0      LISTENING
  TCP    0.0.0.0:445            LAPTOP-JHO8EIUR:0      LISTENING
  TCP    0.0.0.0:7680           LAPTOP-JHO8EIUR:0      LISTENING
  TCP    0.0.0.0:49664          LAPTOP-JHO8EIUR:0      LISTENING
  TCP    0.0.0.0:49665          LAPTOP-JHO8EIUR:0      LISTENING
  TCP    0.0.0.0:49666          LAPTOP-JHO8EIUR:0      LISTENING
  TCP    0.0.0.0:49667          LAPTOP-JHO8EIUR:0      LISTENING
  TCP    0.0.0.0:49668          LAPTOP-JHO8EIUR:0      LISTENING
  TCP    0.0.0.0:49674          LAPTOP-JHO8EIUR:0      LISTENING
  TCP    127.0.0.1:50422        LAPTOP-JHO8EIUR:0      LISTENING
  TCP    192.168.1.49:139       LAPTOP-JHO8EIUR:0      LISTENING
  TCP    192.168.1.49:61157     db5sch101101145:https  ESTABLISHED
  TCP    192.168.1.49:61294     db5sch101101419:https  ESTABLISHED
  TCP    192.168.1.49:63889     arn02s05-in-f6:https   TIME_WAIT
  TCP    192.168.1.49:63890     arn02s05-in-f6:https   TIME_WAIT
  TCP    192.168.1.49:63895     a-0011:https           TIME_WAIT
^C
C:\Users\s_ola>
```

Minilab 2

- Kör netstat –a i kommandotolken/terminalen
  - Ska funka både på Mac och Windows
- Notera hur portnumren skrivs efter IP-numren
  - Kolon är skiljetecken
  - 127.0.0.1:3455

Nätverksprotokoll

- När datorer kommunicerar på nätet använder de ”protokoll”.
- Ett protokoll är en specificerad standard för hur kommunikationen ska gå till mellan olika enheter på ett nätverk
- Massor med protokoll finns
  - HTTP (Hypertext Transfer Protocol)
  - UDP (*User Datagram Protocol*)
  - TCP (*Transmission Control Protocol*)
  - IP (Internet Protocol)

Nätverksprotokoll, vanliga frågor

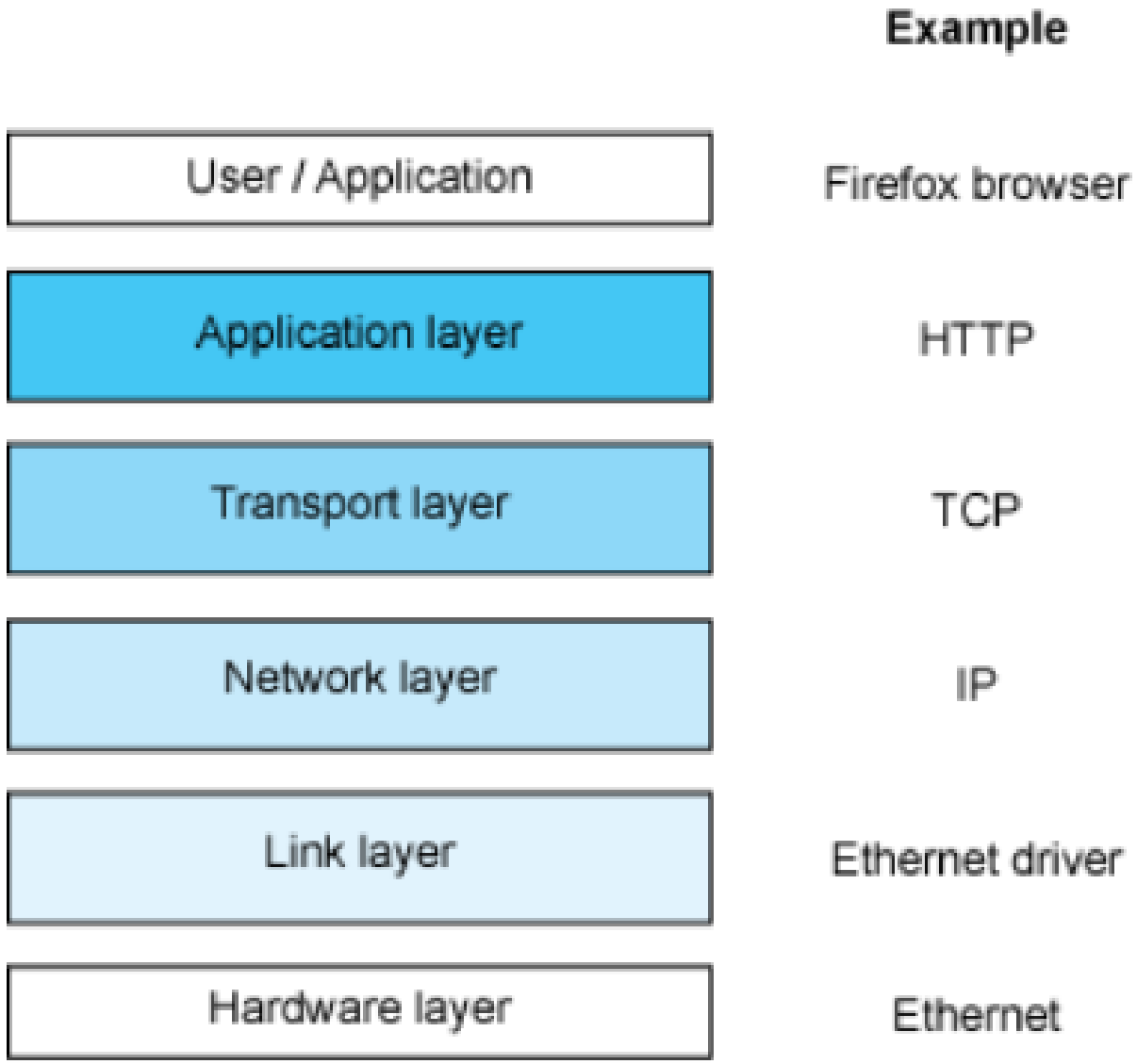
- Ibland hör man uttryck som
  - ”vi kör HTTP och TCP/IP”
  - ”vi kör HTTP över TCP/IP”
- Varför kör man många nätverksprotokoll samtidigt? Blir inte det förvirrat?
- Vad menas med att HTTP är **ÖVER** TCP/IP?

- Först och främst, HTTP, TCP och IP är olika kommunikationsprotokoll
  - Men de berör helt olika delar av kommunikationen
  - Man kan köra flera protokoll samtidigt

NACKADEMIN

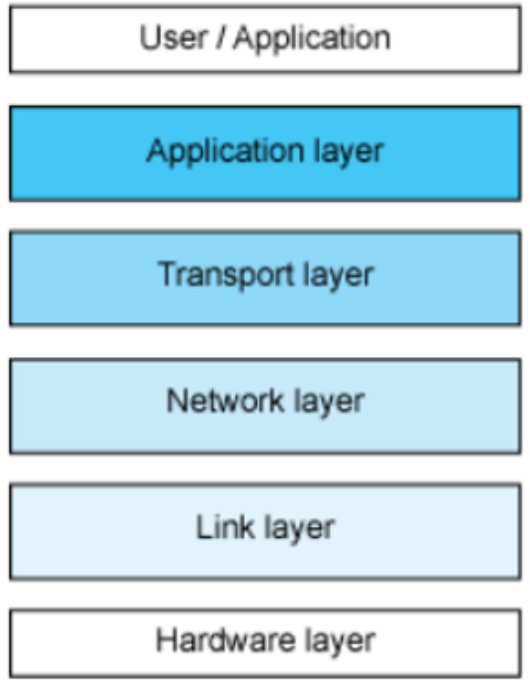
Nätverks-stacken

- För att få all den funktionalitet man behöver har en stack-modell vuxit fram
- Varje lager har ett ansvarsområde för att nätverkskommunikationen ska funka
- Till sin hjälp har de ett antal protokoll och mjukvara (nedersta lagret har hårdvara)

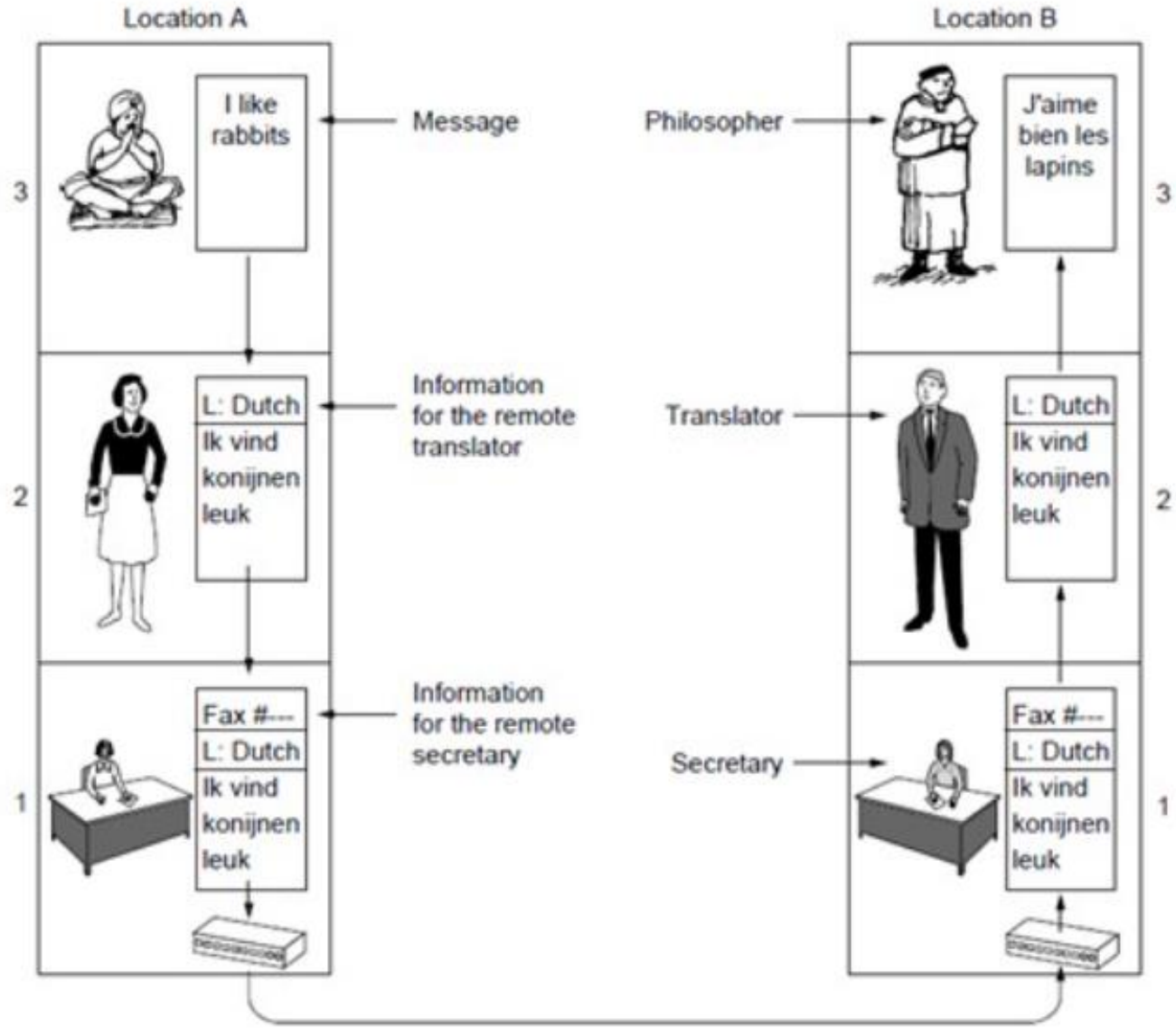


Kommunikation mellan stackar

- Antag att vi har två parter som kommunicerar i ett nätverk
- **Varje lager hos sändaren kommunicerar med samma lager hos mottagaren**
- Men faktiskt data skickas alltid från ett lager till närmaste lagret ovanför eller nedanför i stacken
- **Protokoll** avser kommunikationen mellan samma lager på olika devices
- Mellan varje lager finns ett **interface** som definierar hur kommunikationen mellan två lager på samma device ska gå till (vilka servicar de erbjuder varandra)



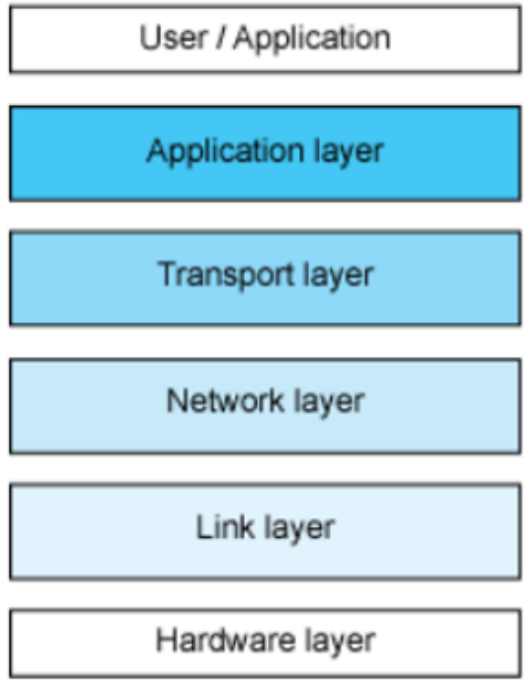
NACKADEMIN



The philosopher-translator-secretary architecture

User/Application

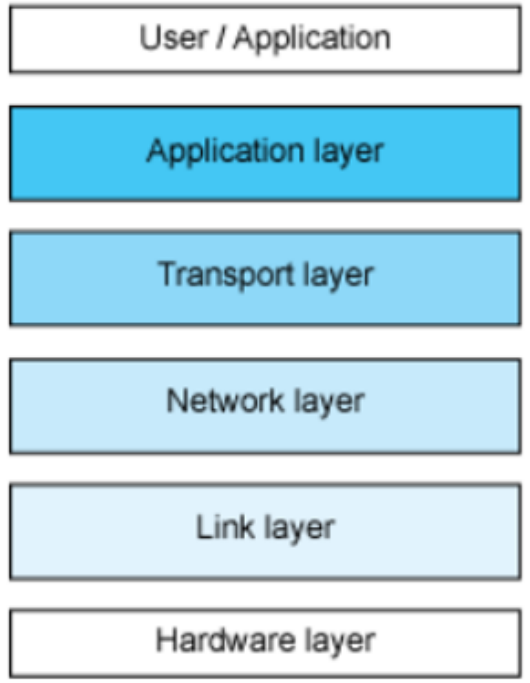
- Det finns olika sorters användare till datorprogram
- Människor kan vara användarna
- Men även program kan vara användare
  - Ett program kan anropa ett annat
- När ett program är användare i detta fall antas att programmet anropar på samma sätt som en mänsklig användare hade gjort
- Program kan även jobba direkt på de nedre lagren
  - Men vi bortser från det I detta exempel



NACKADEMIN

Applikationslagret

- Här verkar HTTP, SMTP m.fl.
- För HTTP skickas HTTP-requests och HTTP-responses
- Applikationslagret hos en sändare skickar en request till en mottagares applikationslager
- Mottagaren skickar tillbaka ett response till sändarens applikationslager
- Men för att datat ska kunna skickas måste det ner genom stacken på ena sidan, skickas fysiskt mellan devices och sen dras upp genom stacken på andra sidan



NACKADEMIN

HTTP

- Två sorters meddelanden finns
  - Request
  - Response
- Request: GET /index.html HTTP/1.1 Host: www.example.com
- Response: HTTP/1.1 200 OK

  Date: Mon, 23 May 2005 22:38:34 GMT
  Server: Apache/1.3.27 (Unix) (Red-Hat/Linux)
  Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
  …
  Content-Type: text/html; charset=UTF-8

  htmltext_htmltext_htmltext____

NACKADEMIN

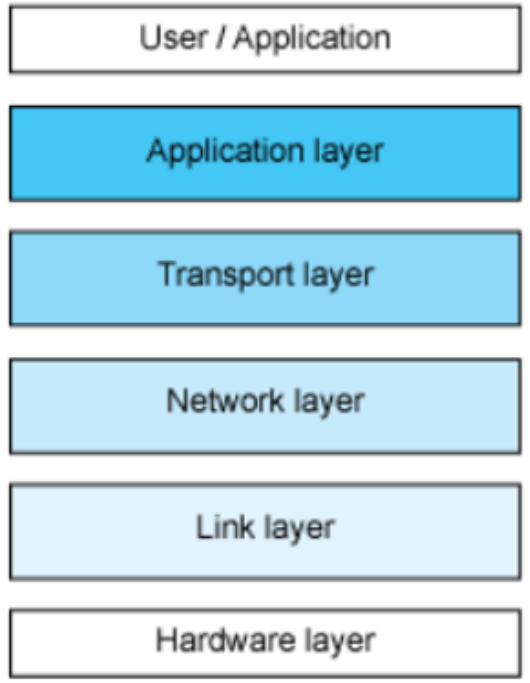
HTTP (2)

- För HTTP-kommunikation är reglerna att den ena kommunicerande parten skickar ett Request, som innehåller bl.a.
  - HTTP-verb (t.ex GET eller POST)
  - Version på HTTP
  - Host-namn
  - Resurs (den del av URL:en som inte är hostname)
- Vidare säger HTTP att ett Response skickas tillbaka som innehåller bl.a
  - Efterfrågad resurs
  - Timestamp
  - Status (200, 404 etc.)
  - Övrigt metadata

NACKADEMIN

Transportlagret

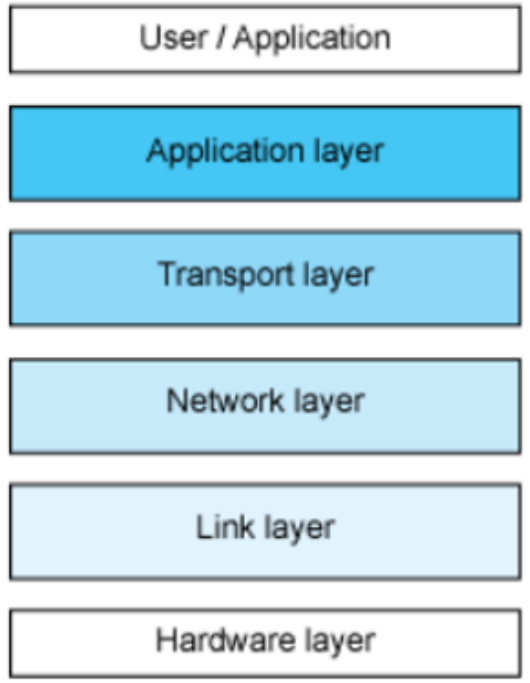
- Här finns logiken som bestämmer **hur** meddelanden ska skickas
- Vanligaste protokollen här är UDP och TCP
- UDP funkar som en radioutsändning
  - Meddelanden skickas, ingen koll på om de tas emot
  - Ingen koll på om datapaket kommer fram i ordning
- TCP har felhantering
  - Mottagaren ska skicka en ”acknowledge”
  - Annars skickas meddelandet om
  - Datapaketen garanteras komma fram i rätt ordning



NACKADEMIN

Transportlagret, forts

- Ett meddelande kommer från Applikationslagret
- Transportlagret tar emot datat, sätter på en header som innehåller det data som behövs för att hålla reda på skickandet
  - Portnummer
  - Checksumma
  - Längd på meddelande



NACKADEMIN

Exempel UDP-header

UDP Header

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Length																Checksum															

- En header skapas med nödvändigt metadata för att skicka vårt HTTP-request
  - Portnummer för sändare och mottagare
  - Längd på meddelandet
  - Checksumma
- Headern sätts på HTTP-requestet

NACKADEMIN

Exempel TCP-Header

TCP Header

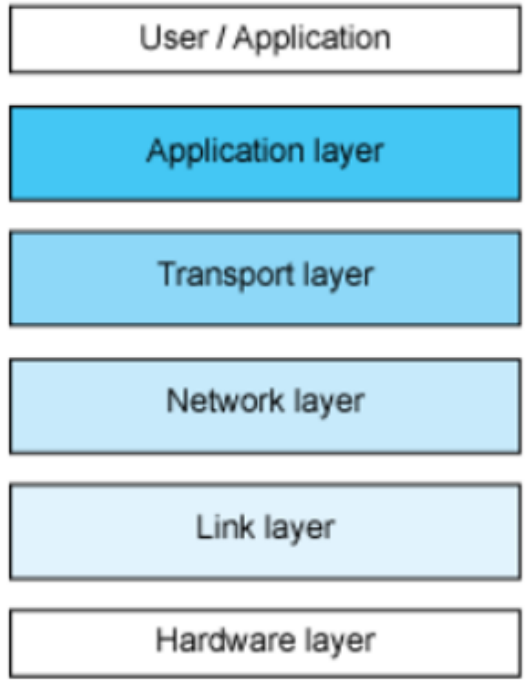
Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size														
16	128	Checksum																Urgent pointer (if URG set)															
20 ...	160 ...	Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.) ...																															

- Även denna sätts fast på HTTP-requestet

NACKADEMIN

Nätverkslagret

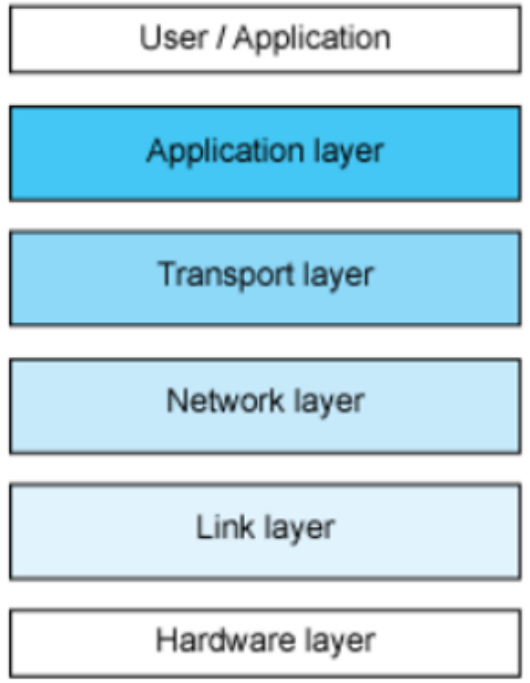
- Först här nere finns kunskap om Internet
- Host-namnet slås upp på en DNS
  - Vi får mottagande dators IP-nummer
- En ny header läggs till
  - Innehåller sändarens IP
  - Innehåller mottagarens IP
  - Med mera
- Paketet är klart att skickas ut på Internet!



NACKADEMIN

Länklagret

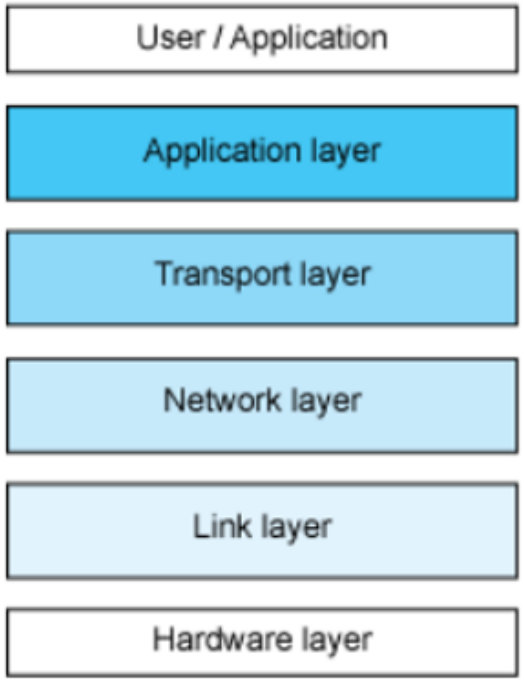
- Ansvarar för att skicka och ta emot ”datapaket”
- Paket som kommer uppifrån skickas ut på nätet
  - Helst i riktning mot en gateway
- Paket som kommer på nätet undersöks
  - Om det skulle till mig, passa det uppåt till nätverkslagret
  - Om inte, skicka vidare



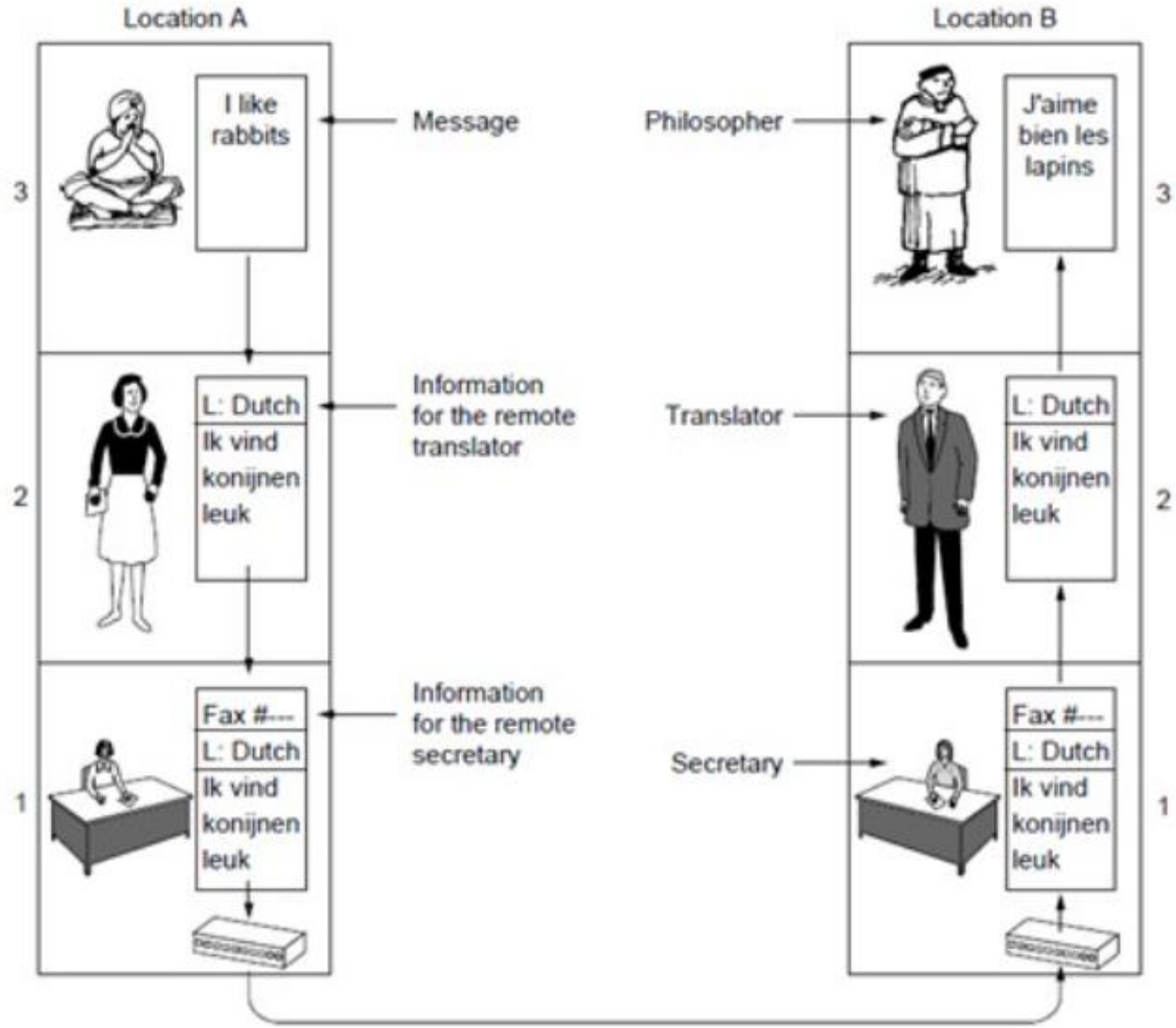
NACKADEMIN

Hårdvarulagret

- Här skickas ettor och nollor ut, och tas emot, från aktuellt transmissionsmedium
- Transmissionsmediumet kan vara
  - Fiber
  - Eter
  - Koppartråd



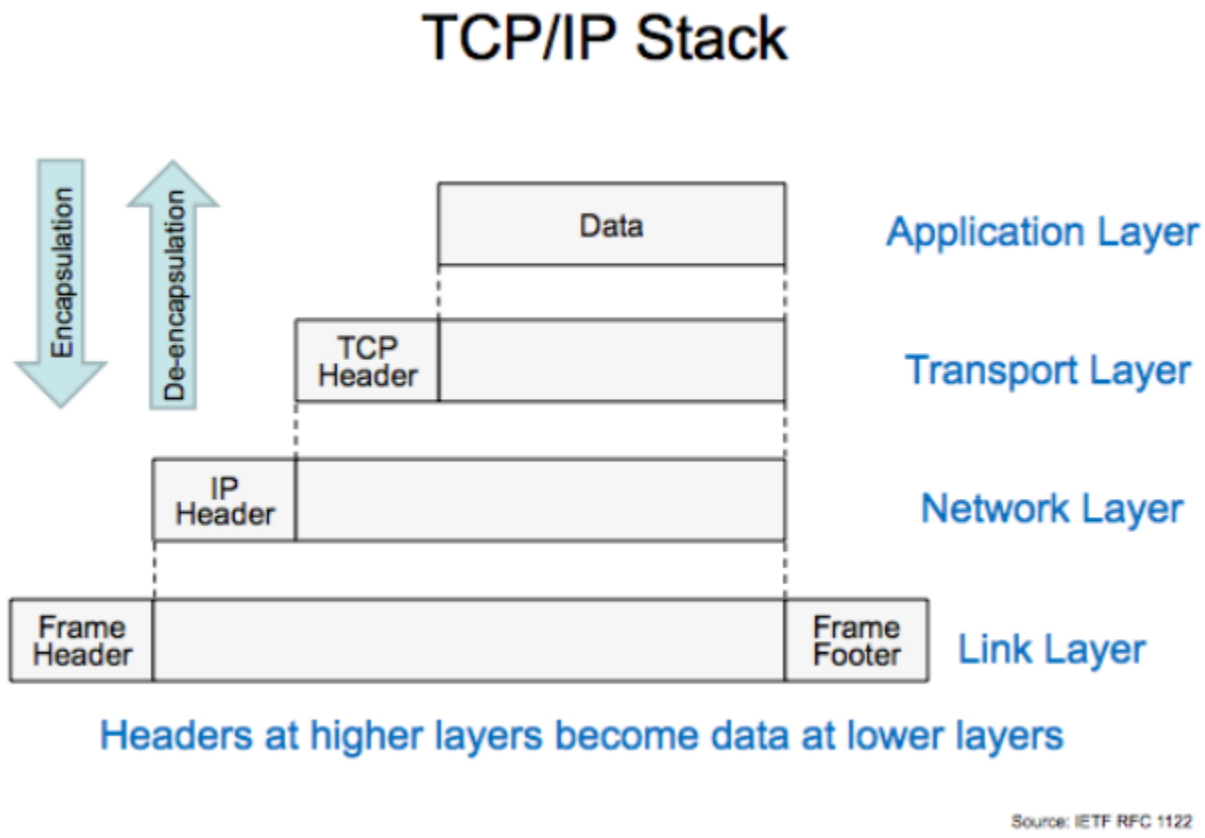
NACKADEMIN



The philosopher-translator-secretary architecture

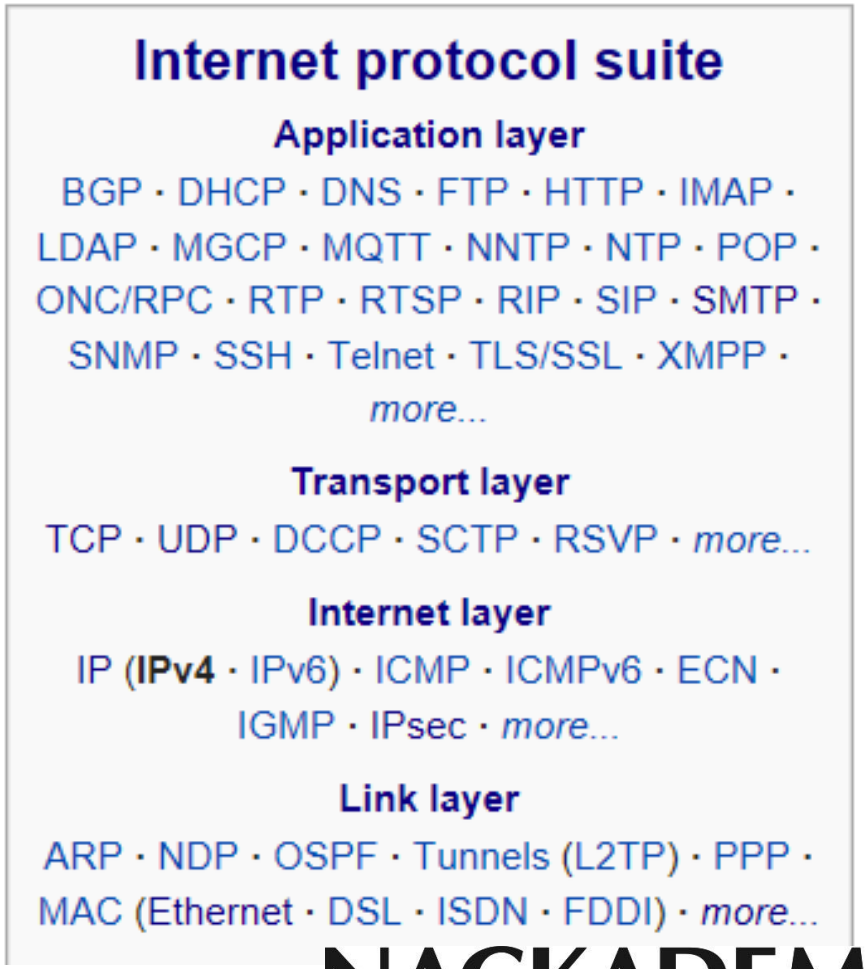
Notera storleken på datapaket

- Varje lager lägger på en header
- De datapaket som faktiskt skickas är större än det ursprungliga datat
- Varje lager strippar sen ”sin” header när paketet är på väg upp i stacken.



Några vanliga protokoll

- Taget från Wikipedia
- Olika protokoll och var i nätverksstacken de hör hemma



NACKADEMIN

UDP

- Funkar som när man skickar brev på posten
- Datagram
  - Självständiga paket med information skickas ut
  - Ingen garanti att de kommer fram i samma ordning som de skickades
  - Ingen garanti att de kommer fram öht
- En sändare skickar ut datagram. Mottagarna lyssnar.
- Mottagarna ”hittar” datagrammet genom att man i sitt program specificerar att det ska lyssna på meddelanden från sändarens IP-adress och ett visst portnummer

NACKADEMIN

UDP, när används det?

- När det viktigaste är att datat kommer fram snabbt
- Vi tar hellre att några paket försvinner på vägen än att förseningar uppstår
- T.ex:
  - Röststreaming
  - Videostreaming

TCP

- Funkar som när man pratar i telefon
- Tvåvägskommunikation
- En ”ledning” upprättas mellan sändare och mottagare
- Ledningen kallas Socket
- En sändare ”skriver” data till socketen
- Mottagaren ”läser” data från socketen
- Funkar på samma sätt som när vi läste och skrev data till filer

UDP vs TCP

- Fördelar UDP
  - Enklare att kommunicera med många parter samtidigt
  - Enklare i situationer när vi inte behöver feedback från mottagaren
  - Snabbare än TCP
- Fördelar TCP
  - Garanterar att all information faktiskt kommer fram till mottagaren
    - Om den tappas bort får man felmeddelande
  - Tvåvägskommunikation möjlig
  - Funkar när det är viktigt att vi tar emot data i en viss sekvens
    - T.ex när vi laddar kollar en sida på nätet, om datat kommer i fel ordning kommer browsern inte att veta hur den ska visa upp sidan om HTML:en är i oordning

UDP

NACKADEMIN

InetAddress

- Representerar en IP-adress på nätet (obs, inte samma sak som url)
- Saknar konstruktor
- Skapas genom att någon av följande statiska metoder anropas
  - InetAddress.getLocalHost() – ger namnet på den egna datorn
  - InetAddress.getByName(”namn”) - där namn är en IP-adress eller namn på en dator
- Användbara metoder:
  - getHostName() – ger datornamnet för denna InetAddress
  - getHostAddress() – ger IP-adressen för denna InetAddress
  - isMultiCastAddress – kan denna adress användas för att skicka meddelanden till många samtidigt?

Dags för InetAddress-demo

- https://github.com/sigrunolafsdottir/JAVASprint4/tree/master/src/InetAddressDemo

Övningsuppgift 1

- Titta på koden till IPtest.java och InetAddressDemo.java (de ligger på https://github.com/sigrunolafsdottir/JavaSprint4/tree/master/src)
- Skriv ett eget litet program som skriver ut din IP-adress.
- Notera din IP-adress (den ska vi använda sen)
- Testa att köra ipconfig/ifconfig i kommandotolken/terminalen
- Testa att köra netstat –a i kommandotolken.
  - Fundera över output:en
- Kolla även in er brandvägg, notera hur vissa program har specifika portnummer specificerade där, och även vilka protokoll de använder

Datagram

- Klassen DatagramPacket
- Innehåller:
  - Mottagarens adress (Klassen InetAddress)
  - Mottagarens portnummer (int)
  - Meddelandet som ska skickas (en bytearray)
  - Längden på meddelandearrayen
- Skapa meddelandearrayen (String till byteArray):

```
String meddelande = ”Hej”;
byte[] data = meddelande.getBytes();
```

- Packa upp meddelande (DatagramPacket till String) :
  - `String s = new String(packet.getData(), 0, packet.getLength())`

NACKADEMIN

Datagram, metoder

- `DatagramPacket d = new DatagramPacket(data, data.length, tillAdr, tillPort);`
- getData() – ger datat i datagrammet
- getLength() - ger längden på datat (på bytearrayen)
- getAddress() – ger sändarens InetAddress (vid mottagning)
- getPort() – ger sändarens portnummer (vid mottagning)
- Notera att det är mottagarens adress som anges i konstruktorn men sändarens adress som fås av getAddress()

NACKADEMIN

DatagramSocket

- För att kunna skicka ett datagram måste vi skapa en DatagramSocket
- DatagramSocket socket = new DatagramSocket();
- Ev kan portnummer anges (om inget portnummer anges tar socketen själv en ledig port)
- För att skicka datagram:
  - Socket.send(packet);
- Ger IOException om sändningen misslyckas
  - Notera att detta inte säger någon om paketet faktiskt kom fram eller inte

DatagramSocket, användbara metoder

- Send(datagram) – skickar ett datagram
- Receive(datagram) – tar emot et datagram
- setSoTimeout(ms) – anger hur många millisekunder receive ska vänta innan den ger InterruptedIOException
- Close() – stänger förbildelsen
- getLocalPort – ger portnumret som socketen är bunden till

Exempel datagram, sändare

```
public class DatagramSender {
    public static void main(String[] args) throws UnknownHostException, SocketException, IOException{
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

        InetAddress toAdr = InetAddress.getLocalHost();
        int toPort = 55555;
        DatagramSocket socket = new DatagramSocket();
        String message;
        System.out.println("?: ");
        while((message = in.readLine()) != null){
            if (message.equals("bye"))  System.exit(0);
            byte[] data = message.getBytes();
            DatagramPacket packet = new DatagramPacket(data, data.length, toAdr, toPort);
            socket.send(packet);
            System.out.println("?: ");
        } System.exit(0);
    }
}
```

NACKADEMIN

Exempel datagram, mottagare

```
public class DatagramReceiver  {

    public static void main(String[] args) throws SocketException, IOException{
        int minPort = 55555;
        DatagramSocket socket = new DatagramSocket(minPort);
        byte[] data = new byte[256];
        while(true){
            DatagramPacket packet = new DatagramPacket(data, data.length);
            socket.receive(packet);
            System.out.println("Meddelande från "+packet.getAddress().getHostAddress());
            String message = new String(packet.getData(), 0, packet.getLength());
            System.out.println(message);
        }
    }
}
```

NACKADEMIN

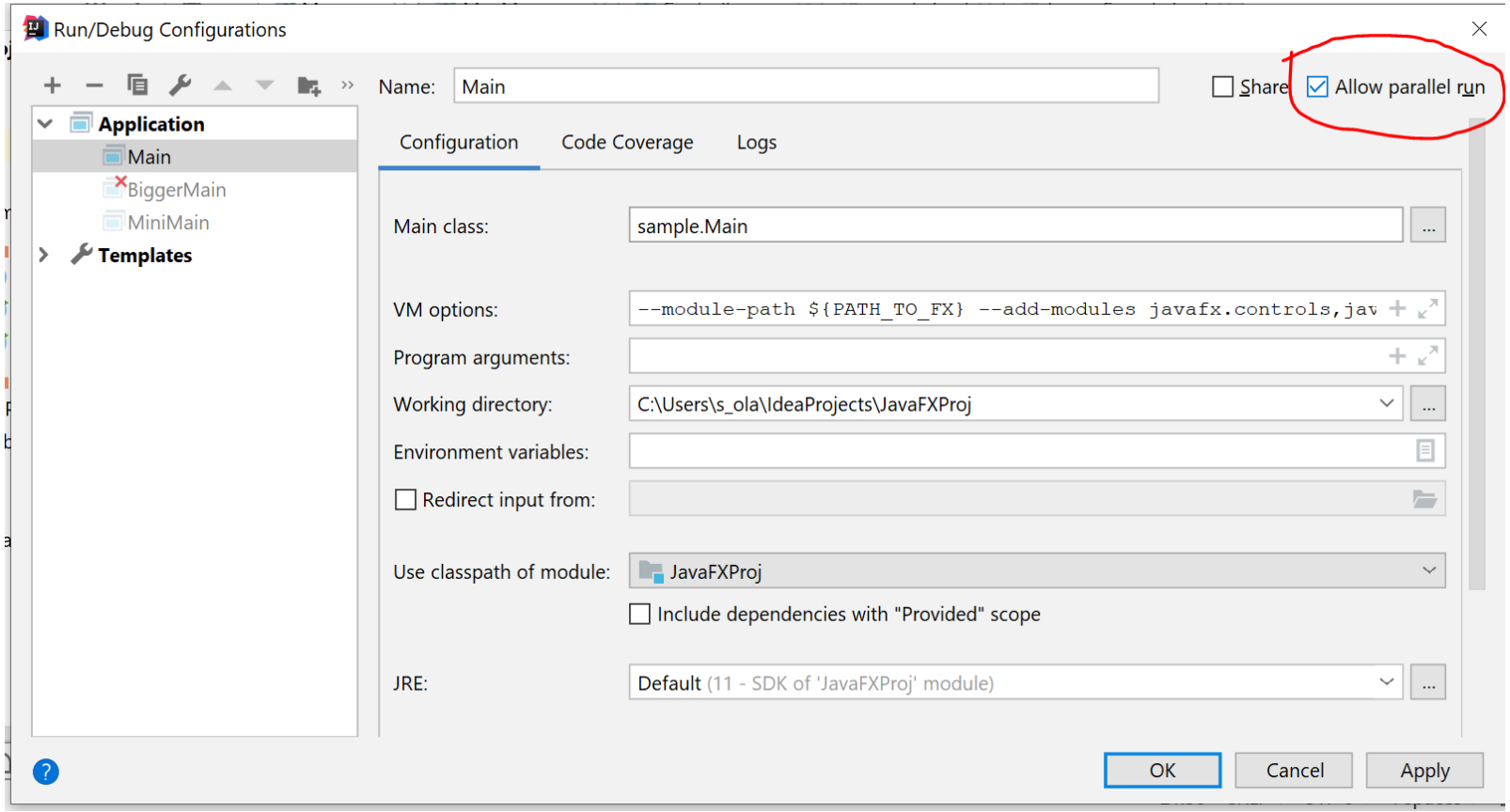
Dags för datagram-demo

- https://github.com/sigrunolafsdottir/JAVASprint4/tree/master/src/DatagramDemo

NACKADEMIN

För att kunna köra parallella program i IntelliJ

- Run -> Edit Configuration -> Allow parallell run



Övningsuppgift 2a

- Skapa ett sändar-program som sänder ut ett tänkvärt citat var 5:e sekund via datagram (skriv ner ca 3 citat och lägg utsändningen i en loop där den egna tråden sover 5 sek/iteration).
- Skapa ett mottagarprogram som tar emot citaten och skriver ut dem på consolen.

NACKADEMIN

Övningsuppgift 2b

- Para ihop er två och två.
- Ändra nu ditt sändarprogram så att det sänder till din kompis dator genom att ange den personens dators ip och det portnummer som den personens lyssnar-app använder.
- Det kan dyka upp ett varningsmeddelande, klicka på att det är ok att ta emot data.
- Om du inte lyckas ta emot data kan du behöva:
  - Kolla att du använder dig av rätt nätverksinterface
  - Kolla att du har IP4 inställt som default (och inte IP6)
  - Kolla att brandväggen låter dig ta emot data

NACKADEMIN

LUNCH

Multicast

- Ibland vill man skicka meddelande till många mottagare samtidigt.
- Multicast låter oss skicka meddelanden till en grupp
- Meddelanden skickas mha MulticastSocket

NACKADEMIN

MulticastSocket

- Subklass till DatagramSocket
- `MulticastSocket m = new MulticastSocket(port);`
- Ingen specifik IP-adress anges
- Istället används en multcastadress
  - IP i intervallet 224.0.0.1-239.255.255.255
- Uppkoppling:

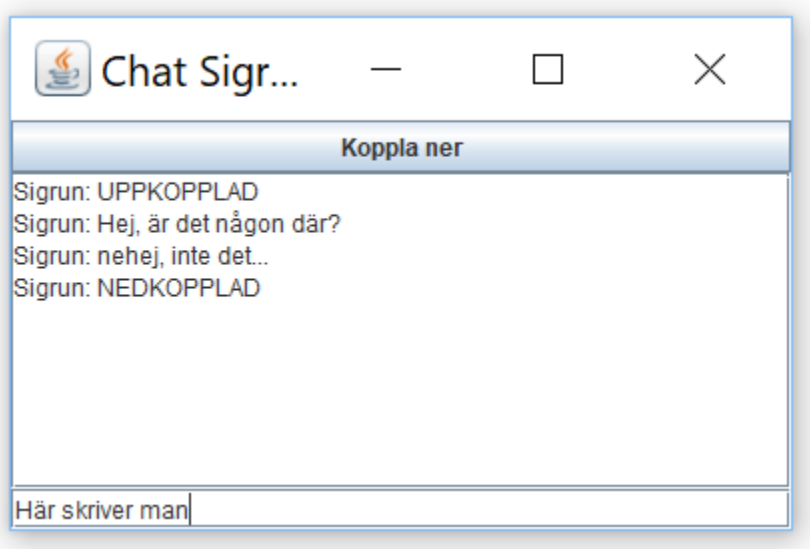
```
Iadr = InetAddress.getByName(”234.235.236.237”);
m.joinGroup(Iadr);
```

NACKADEMIN

MulticastSocket, användbara metoder

- send(packet) - skickar ett datagram
- receive(packet) – tar emot ett datagram
- joinGroup(iadr) – ansluter sig till en multicastgrupp
- leaveGroup(iadr) – lämnar en multicastgrupp
- close() – stänger ner förbindelsen

Övningsuppgift 3, klasschatt



- Skriv en klass-chatt som på bilden
- Följande grafiska komponenter behövs
  - En knapp för nerkoppling från multicast-grupp
  - En JTextArea där din + alla andras texter visas
  - En JTextField där du skriver dina texter. När du trycker ”Enter” ska textfältet tömmans och texten visas i textArean, tillsammans med ditt namn.
- Chatten ska använda sig av multicast
  - Du behöver skicka ut dina meddelanden (tänk på att appenda ditt namn först)
  - Du behöver fånga andras meddelanden (i samma applikation)
- **Använd multicastIP 234.235.236.237 och portnummer 12540.**

Övningsuppgift 3, klasschatt, tips

- Det går bra att använda samma multicastSocket för att både lyssna och sända samtidigt
- Eventuellt kan du vilka dela upp lyssnandet och mottagandet i olika trådar
  - Man kan ändå använda sig av samma socket, om man vill

NACKADEMIN

Övningsuppgift 4a, väderrapporteringssystem

- Skriv ett system för att hantera väderrapportering från olika städer via UDP. Du behöver två olika program, ett mottagarprogram som tar emot väderdata och ett sensorprogram som skickar data.
- Låt sensorprogrammet först fråga användaren efter vilken stad hen befinner sig i. Läs sedan in den temperatur användaren skriver från kommandoraden och skicka stadens namn och temperaturen till mottagaren.
- Låt mottagaren lyssna efter datagram och ta emot skickat data och skriva ut stad och temp i consolen, tillsammans med en tidsstämpel.
- Du ska kunna ha flera sensorprogram igång samtidigt för att kunna simulera inskick från flera städer. Mottagarprogrammet ska kunna lyssna på flera sensorprogram samtidigt.

Övningsuppgift 4b, multicast

- Gör en multicast-variant av vädersystemet I uppgift 4a.
- Låt varje sändarprogram bestå av ett litet Swingprogram där användaren skriver in stad och temp i två JTextFields
- När användaren trycker enter skickas stad och temp till mottagarprogrammet
- Låt mottagaren också vara ett litet Swingprogram, som skriver ut sina mottagna meddelanden i en JTextArea
- Mottagaren ska lyssna till en multicastadress och kunna ta emot data från hela klassens sändare.

NACKADEMIN

Sammanfattning

- UDP är bra när man vill skicka meddelanden snabbt men när det inte är jätteviktigt att de kommer fram i rätt ordning (eller att de kommer fram alls)
- Datagram är informationspaket som skickas över nätet till en viss IP-adress (eller multicast-gruppadress)
- Socketar är ”dataledningar” som används för att skicka och ta emot data över nätet
- Multicast låter oss skicka meddelanden till grupper av mottagare samtidigt

NACKADEMIN

Framåtblick inför nästa lektion

- TCP
- Client-Server
- Protokoll

NACKADEMIN