Tentamen
Nätverksprogrammering
Del 2

2010–06–01, 08.00–13.00

Tillåtna hjälpmedel för denna del av tentamen:

- Java snabbreferens
- Kursboken: Java Network Programming av Eliotte Rusty Harold.
- Valfri lärobok i Java
- Utskrift av OH-bilder från föreläsningarna.

Denna tentamen i kursen Nätverksprogrammering består av två delar – en del som innehåller frågor av teoretisk/principiell/utredande karaktär och en del som innehåller praktiska programmeringsuppgifter. Detta är del 2. Den ska du ha erhållit tillsammans med ett färgat tentamensomslag när du lämnade in din lösning på del 1 av tentamen.

För godkänt betyg på tentamen krävs sammanlagt minst 20 poäng på tentamen, varav minst 8 poäng på vardera deltentamen. För högre betyg krävs naturligtvis mer, så gör så många uppgifter du kan.

Denna deltentamen innehåller två uppgifter som kan ge 10 poäng vardera. Uppgifternas ordningsföljd avspeglar inte nödvändigtvis deras inbördes svårighetsgrad.

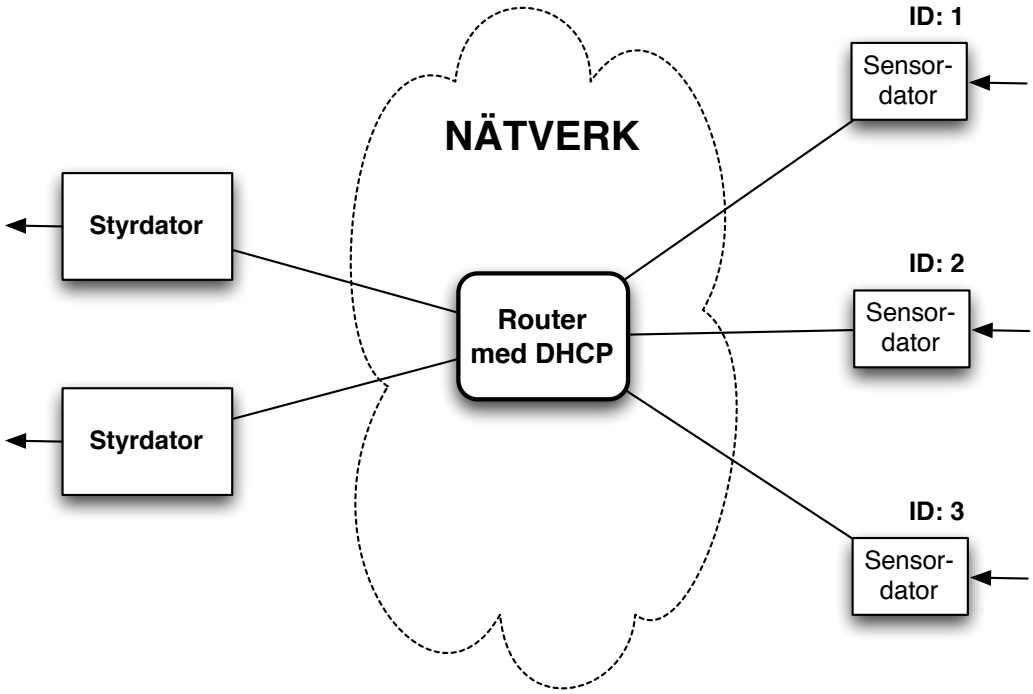
---

1. Överföring av sensordata med UDP

Inom processautomation börjar det bli allt vanligare med nätverk av datorer som tillsammans styr processen. För att få indata till styrningen behöver dessa datorer tillgång till sensorer som känner av processens aktuella tillstånd. Värdena från dessa sensorer används sedan för att styra processen. Sensorerna är ofta distribuerade och kopplas även dessa in i systemet via nätverket.

Uppgiften handlar om ett sådant enkelt nätverk i processmiljö. Nätverket är ett vanligt IP-baserat nätverk. Kärnan i nätverket består av en router med inbyggd DHCP-server. Till denna router ansluts dels en eller flera styrdatorer som styr processen, samt ett antal enklare datorer kopplade till olika sensorer. Styrdatorn/styrdatorerna hämtar regelbundet mätvärden från datorerna som är kopplade till sensorerna och räknar ut nya styrsignaler utgående från dessa. Flera styrdatorer kan hämta sensorvärden från samma sensordator. Se figuren på nästa sida.

En enskild sensordator identifieras genom att den har ett unikt nummer (1,2,3...). Detta nummer anges av operatoren (via kommandoraden) när programmet som styr respektive sensordator startas. Det IP-nummer som sensordatorn får är däremot ej förutsägbart. DHCP-servern som delar ut IP-nummer gör detta i praktiken slumpmässigt så en styrdator som vill kommunicera med sensordatorerna, eller för den delen operatören av systemet, kan ej känna till de enskilda sensordatorernas aktuella adresser vid uppstart (de förändras dock inte under drift).



Uppgift
Din uppgift är att implementera programvaran i sensordatorerna samt den del i styrdatorn som har hand om kommunikationen mellan en styrdator och en av sensordatorerna enligt beskrivningen nedan.

Observera: Designkrav på systemet är att UDP ska användas för kommunikation mellan styr- och sensordator samt att all programvara ska implementeras i Java.

Styrdator
Implementera följande trådklass som används för att kommunicera med en given sensordator. Du får givetvis lägga till egna privata metoder och attribut efter behov.

```
public class SensorCommunication extends Thread {

    /* Create a thread that communicates with sensor id. */
    public SensorCommunication(int id) { ... }

    public void run() {
        ...
    }
}
```

- Använd multicast när tråden startar för att hitta IP-adressen för sensordatorn med det givna id-numret.
- Om inget svar erhålls inom en sekund på multicastförfrågan så gör ett nytt försök att kontakta sensordatorn. Om inget svar erhållits efter 10 misslyckade försök så avbryt programexekveringen med ett felmeddelande.
- När sensordatorns IP-nummer är känt övergår tråden till att ungefär en gång i sekunden skicka en UDP-begäran (unicast) till sensordatorn om aktuellt sensorvärde. Vänta som mest 0,5 sekunder på svar (UDP-paket kan ju försvinna).

- Om inget svar erhålls enligt föregående punkt och vi avbryter försöket att hämta sensordata kan det ändå vara så att ett svar bara var ovanligt mycket försenat och anländer strax efter att vi gav upp försöket. För att inte svaret ska ligga kvar och av misstag betraktas som svar för nästa förfrågan behöver vi kunna sortera bort dessa meddelanden. För full poäng på uppgiften krävs en lösning som med rimlig säkerhet sorterar bort dessa försenade meddelanden.
- Behandlingen av de inlästa sensorvärdena ingår ej i uppgiften. Därför antas följande färdiga klass finnas tillgänglig (ska ej implementeras):

```
public class Control {
    /* A new value from sensor sensorid with value sensorvalue is now available. */
    public static void newValue(int sensorid, int sensorvalue) { ... }

    /* No response was received from sensor sensorid this time interval. */
    public static void noResponse(int sensorid) { ... }
}
```

  Anropa metoden `newValue()` om svar erhölls i tid, annars metoden `noResponse()`.
- *LEDNING 1:* Med hjälp av metoden `void setSoTimeout(int timeout)` i socket-klasserna (Socket/DatagramSocket) kan man ange att ett SocketTimedoutException ska genereras angivet antal millisekunder efter anrop av `read()/receive()` om inget svar erhålls. Metoden `setSoTimeout()` kan generera ett SocketException. Se även kursboken.
- *LEDNING 2:* För att vänta tills nästa gång det är dags att hämta ett sensorvärde kan du använda metoden `sleep(long milliseconds)` som finns i klassen Thread. Metoden kan generera ett InterruptedException som måste fångas. Det angivna tidsintervallet för att hämta sensorvärden är att se som ungefärligt.

Sensordator
Implementera ett komplett javaprogram, komplett med main-metod, för att styra en sensordator som fungerar ihop med programmet för en styrdator ovan.

- Antag att operatören anger sensordatorns id-nummer (1,2,3...) på kommandoraden vid start av Javaprogrammet.
- Följande klass antas finnas tillgänglig för att läsa av aktuellt sensorvärde utan att särskild import behöver göras(för att förenkla uppgiften något utgörs sensorvärdet av ett heltal).

```
public class Sensor {
    public static int readValue() { ... }
}
```

- Tänk på att flera styrdatorer - vid olika tillfällen i tiden - kan vilja börja använda sig av sensordatorn. Därför måste programmet alltid vara beredd på att svara på multicastförfrågningar samt kunna hantera förfrågningar om sensordata från flera håll.

(10p)

2. Första uppdraget som civilingenjör inom IT

Företaget *Den galaktiska mjukvaran* i Lund har utvecklat en http-server i Java för sin serverpark och använde koden för servern JHTTP i kursboken, *Java Network Programming, 3:e upplagan, sidan 356–362 (JNP/3)*, som utgångspunkt för projektet. Strax efter att första versionen av servern var klar bestämde sig projektledaren, Oppfinnar-Jocke Johansson, för att ta tjänstledigt och åka iväg på en världomsegling med sin segelbåt. Som nyanställd på ditt första arbete efter examen blir det därför din uppgift att komplettera servern med den funktionalitet företaget behöver. Du ombedes därför att göra ett antal mindre modifieringar av koden enligt nedan. Du återfinner koden för en något modifierad (felrättad) version av koden i boken som bilaga till tentamen.

I uppgiften ingår tre helt oberoende deluppgifter (a, b och c). Gör så många du kan och hinner i vilken ordning du vill.

a) *Den galaktiska mjukvaran* vill att servern förutom att hantera dokument i formaten text (".txt", HTML (".html"), javakällkod (".java"), javaklasser (".class") och bilder i GIF-/JPEG-format (".gif"/".jpg"/".jpeg") även överföra filmer som strömmande media. Utvidga koden så att även filer i formaten MPEG (".mpg"), Quicktime (".qt") och VRML 3D (" .wrl") förses med korrekt MIME-deklarationer när de laddas ner från servern. Skriv nödvändig kod och ange med hjälp av radnumren i kodbilagan var du vill stoppa in den. Listan av MIME-typer finns på sidorna 59–62 i JNP/3 boken samt i bilagan till tentamen.

(2p)

b) Av de många olika HTTP-kommandona som kan sändas till en webbserver implementerar den aktuella servern från JNP/3 enbart GET. Företaget *Den galaktiska mjukvaran* vill utvidga stödet för fler HTTP-kommandon och din uppgift blir därför att implementera stöd även för HEAD. Beskrivningen av HEAD lyder som följer. All relevant information finns i det första stycket ("message body" = själva innehållet i dokumentet).

```
The HEAD method is identical to GET except that the server MUST NOT
   return a message-body in the response. The metainformation contained
   in the HTTP headers in response to a HEAD request SHOULD be identical
   to the information sent in response to a GET request. This method can
   be used for obtaining metainformation about the entity implied by the
   request without transferring the entity-body itself. This method is
   often used for testing hypertext links for validity, accessibility,
   and recent modification.

   The response to a HEAD request MAY be cacheable in the sense that the
   information contained in the response MAY be used to update a
   previously cached entity from that resource. If the new field values
   indicate that the cached entity differs from the current entity (as
   would be indicated by a change in Content-Length, Content-MD5, ETag
   or Last-Modified), then the cache MUST treat the cache entry as
   stale.
```

Källa: http://tools.ietf.org/html/rfc2616#section-9.4

Skriv nödvändig kod och ange med hjälp av radnumren i kodbilagan var du vill stoppa in den och/eller vilka rader du vill ändra på.

(2p)

c) *Den galaktiska mjukvaran* har behov av att samla in information från sina kunder och vill därför ha ett enkelt stöd för att köra CGI-skript via HTTP-kommandot POST. Ett exempel kan vara att man vill att kunder ska kunna registrera sig med namn och e-postadress via ett formulär som kan se ut ungefär som följer:

När formuläret har fyllts i kommer ett POST-kommando att skickas till webbservern. Detta POST-kommando kan se ut som följer:

```
POST /galactic_crunch.pl HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; fr; rv:1.9.2.3) Gecko...
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Referer: http://localhost:8000/test_temp_post_perl.html
Content-Type: application/x-www-form-urlencoded
Content-Length: 50

name=Pierre+Nugues&email=Pierre.Nugues%40cs.lth.se
```

Några kommentarer till POST-kommandot ovan:

- Rader avslutas med teckenkombinationen CR+LF, dvs två tecken med koderna 13 respektive 10.
- URLen efter POST är namnet på det externa program som ska köras för att ta hand om begäran från klienten.
- Efter den inledande raden med POST kommer ett antal header-rader som i de flesta fall kan ignoreras.
- Header-raden som inleds med ”Content-Length” anger hur lång den s.k. query-strängen är.
- Query-strängen följer direkt efter en tomrad som avslutar header-delen av begäran.

I) Skriv en metod `processQuery()` i klassen `RequestProcessor` med signatur enligt nedan som givet sökvägen till ett externt kommando exekverar detta och skriver resultatet på en given ström.

```
/* Execute the external command given by path. Submits the query string
   given by query to the external command as a command line argument.
   All output to standard out from the command is written to the stream
   given by out. */
public void processQuery(String path, String query, Writer out)
                                                    throws IOException {
    ...
}
```

Ledningar:

- Du kan starta ett externt kommando med hjälp av metoden `exec()` i klassen `Runtime` enligt följande exempel:

```
String path = "/home/Pierre/cgicommand";
String query = "name=Pierre+Nugues&email=Pierre.Nugues%40cs.lth.se";
Process process = Runtime.getRuntime().exec(path+" "+query );
InputStream inp = process.getInputStream();
OutputStream outp = process.getOutputStream();
```

  Det som det externa kommandot skriver till standard output kan du läsa från `inp` ovan och, omvänt, det som skrivs till `outp` går till kommandots standard input. När kommandot exekverat färdigt stängs strömmarna.
- Hantering av input-/outputströmmarna ovan kan ge upphov till ett IOException. Därav `throws`-deklarationen.
- Skriv allt som går att läsa från det som det externa kommandot skriver på standard output till strömmen som anges av argumentet `out`.

(3p)

II) Modifiera den givna webbservern så att den klarar av att hantera POST-kommandon enligt ovan. Skriv nödvändig kod och ange med hjälp av radnumren i kodbilagan var du vill stoppa in den och/eller vilka rader du vill ändra på.
Ledningar:

- Använd metoden `processQuery()` du skrev i förra deluppgiften för att exekvera CGI-skriptet.
- Path-delen av URLen i POST-kommandot förväntas utgöra en korrekt sökväg till ett externt kommandot.

(3p)

Slut – Glad sommar!