LÖSNINGAR

Tentamensskrivning — Nätverksprogrammering (EDA095 - FED) 2004-05-25, kl 8-13

DEL 1 - Frågor av teoretisk, principiell eller utredande karaktär

1. a)

1. UDP = User Datagram Protocol
2. TCP = Transmission Control Protocol

b) UDP är ett okopplat protokoll baserat på sändning av "paket" av data i form av bytes. Varje paket kan ha en separat mottagare. Det finns ingen garanti att ett paket kommer fram till adressaten eller att paketen kommer fram i samma ordning de skickades. Kommer ett paket fram garanteras dock att det är korrekt. Maximal paketstorlek är 65507 bytes (IPv4).

TCP är ett kopplat protokoll, dvs man upprättar en "fast" förbindelse mellan två noder i nätverket. All data som sänds går till den nod som man upprättade förbindelsen till. Data skickas (ur programmerarens synvinkel) i form av en ström utan uppdelning i enskilda paket. Det garanteras att data kommer fram och att data kommer fram i rätt ordning (automatisk omsändning).

2. Common Gateway Interface, CGI, är en teknik för att få en webbserver att dynamiskt skapa HTML-sidor när en webbläsare begär en sida. Det är en av de äldsta teknikerna för detta och kan t.ex. användas för att behandla inmatning i webbformulär.

Tekniskt fungerar det så att när webbservern mottar en begäran av en CGI-baserad webbsida startas ett externt program som läser in parametrarna för begäran, som texten som matades in i ett webbformulär, och baserat på detta genererar HTML-kod. HTML-koden skrivs till standard output, tas emot av servern och skickas vidare till webbläsaren. Därefter terminerar det externa programmet. Behovet av att starta upp en ny operativsystemsprocess gör tekniken ganska tung och har till stor del ersatts av modernare tekniker som t.ex. JSP.

3. RFC står för Request For Comments och är ett dokument som beskriver ett förslag till standard för t.ex. ett protokoll som används på Internet. Trots namnet fungerar en RFC oftast som en accepterad de facto-standard för det som beskrivs. Exempel på tekniker som beskrivs i form av RFCer är TCP, HTML och SMTP (mailkommunikation).

4.

1. Begäran anländer i form av en “GET”-begäran från klienten.
2. Servern upptäcker, vanligen genom att det begärda dokumentet slutar på “.jsp” att det är fråga om ett JSP-dokument.
3. Om det är första gången dokumentet begärdes läser servern in .jsp-filen och startar en JSP-kompilator för att översätta denna fil till ett Java-program i form av en servlet.
4. Java-filen kompileras (om ej redan kompilerad) med Java-kompilatorn till en färdig servlet.
5. Den genererade servleten startas och genererar ett svar till klienten.
6. Svaret skickas till klienten.

5. I RMI-samband är en stubbe (stub) en klass som på klientsidan representerar det objekt på servern som ska anropas över nätet. Stubben ser för klientprogrammet ut precis som objektet på servern och omvandlar metodanrop till meddelanden som skickas över nätet. På serversidan finns det ett s.k. skelett (skeleton) som tar emot meddelanden från klienten och

översätter dem till metodanrop på serverobjektet. Från Java version 1.2 finns det inte behov av en separat skelettklass, utan dess funktionalitet är inbyggd i Javas standrdklasser för RMI.

6.

a) Ström som används för att lagra data på ett skivminne: **FileOutputStream**
b) Superklass för alla klasser som representerar utgående strömmar: **OutputStream**
c) Superklass för flera av de klasser som utökar funktionaliteten hos en existerande ström: **FilterOutputStream**
d) Innehåller metoder för att skicka olika typer av primitiva datatyper, t.ex. heltal, flyttal: **DataOutputStream**

7. a) Ömsesidig uteslutning innebär att bara en tråd i taget kan använda sig av en delad resurs. Om en andra tråd försöker använda sig av samma resurs kommer den att blockeras tills den första tråden är färdig med resursen.

b) Busy-wait

c) Mycket CPU-krävande. I värsta fall kan den helt hindra andra mera lågprioriterade trådar att köra genom att ta upp all tillgänglig CPU-tid.

d)

```
public class BinarySemaphore {
    boolean locked = false;

    public synchronized void take() {
        while(locked) {
            try {
                wait();
            } catch(InterruptedException e) { }
        }
        locked = true;
    }

    public synchronized void give() {
        locked = false;
        notify();
    }
}
```

8.

a) Falskt.
b) Korrekt.
c) Falskt.
d) Korrekt.

DEL 2 - Praktiska programmeringsuppgifter

1. TicTacToe, eller “tre-i-rad”

```
import java.io.*;
import java.net.*;

class TicTacToe {
    public static void main(String [] args) {
        try {
            ServerSocket ss = new ServerSocket(7878);
            while (true) {
                Socket player1,player2;
                player1 = ss.accept();
                player1.getOutputStream().write(“Please wait for
                                                    opponent...\r\n”.getBytes());
                player2 = ss.accept();
                new TicTacToeGame(player1,player2).start();
            }
        } catch(IOException e) {
            System.out.println(“Network error, shutting down.”);
        }
    }
}

class TicTacToeGame extends Thread {
    private OutputStream[] p;
    private BufferedReader[] r;
    private Socket[] s;

    public TicTacToeGame(Socket p1,Socket p2) {
        s = new Socket[2];
        r = new BufferedReader[2];
        p = new OutputStream[2];
        s[0] = p1;
        s[1] = p2;
    }

    public void run() {
        boolean stop = false;
        TicTacToeBoard board = new TicTacToeBoard();
        int turn = 0;
        int moves;
        try {
            p[0] = s[0].getOutputStream();
            r[0] = new BufferedReader(new InputStreamReader(
                                                        s[0].getInputStream()));
            turn = 1;
            p[1] = s[1].getOutputStream();
            r[1] = new BufferedReader(new InputStreamReader(
                                                        s[1].getInputStream()));
            while (!stop) {
                turn = 0;
                println(p[0],”New game begins! Your mark is ‘X’.\r\n”);
                turn = 1;
                println(p[1],”New game begins! Your mark is ‘O’.\r\n”);
                board.clear();
                turn = (int) (Math.random()*2.0);
                moves = 0;
                while (!board.gameWon() && !stop && moves<9) {
                    turn = 1-turn;
```

```
                println(p[turn],board.toString());
                boolean legal = false;
                while (!legal) {
                    println(p[turn],"Enter your move (or 'q' to quit): ");
                    String response = r[turn].readLine();
                    if (response.equals("q")) {
                        stop = true;
                        legal = true;
                    } else {
                        if (response.length()==2) {
                            int r = ((int)response.charAt(0))-((int)'0');
                            int c = ((int)response.charAt(1))-((int)'0');
                            if (r>0 && r<4 && c>0 && c<4 &&
                                board.isFree(r-1,c-1)) {
                                if (turn==0) {
                                    board.setX(r-1,c-1);
                                } else {
                                    board.setO(r-1,c-1);
                                }
                                moves++;
                                legal = true;
                                println(p[turn],board.toString());
                            }
                        }
                    }
                    if (!legal) {
                        println(p[turn],"Illegal move or command,
                                                    try again!\r\n");
                    }
                }
            }
            if (!stop) {
                if (!board.gameWon()) {
                    println(p[turn],"Game over. There was a tie!\r\n\r\n");
                    turn = 1-turn;
                    println(p[turn],"Game over. There was a tie!\r\n\r\n");
                } else {
                    println(p[turn],"Game over. You won!\r\n\r\n");
                    turn = 1-turn;
                    println(p[turn],"Game over. You lost!\r\n\r\n");
                }
            } else {
                println(p[turn],"You left the game.\r\n");
                turn = 1-turn;
                println(p[turn],"Your opponent left the game!\r\n");
            }
        }
    } catch(IOException e) {
        try {
            println(s[1-turn].getOutputStream(),
                    "Network error occurred, terminating game.\r\n");
        } catch(IOException x) {}
    }
    try { s[0].close(); } catch(IOException e) {}
    try { s[1].close(); } catch(IOException e) {}
}

private void println(OutputStream o,String s) throws IOException {
    o.write(s.getBytes());
    o.flush();
}
}
```

2. Internetradio via multicast

```
import java.net.*;
import java.io.*;

public class NetworkRadioBroadcaster{
    public static void main(String args[]) {
        SoundIO io = new SoundIO();
        try {
            MulticastSocket ms = new MulticastSocket();
            byte[] buf = new byte[2048];
            InetAddress receiver =
                                InetAddress.getByName("all-systems.mcast.net");
            ms.joinGroup(receiver);
            DatagramPacket dp = new DatagramPacket(buf,buf.length,
                                                   receiver,40099);
            while(true) {
                int len = io.read(buf);
                dp.setLength(len);
                ms.send(dp,1); // A TTL of one is enough for the Department
            }
        } catch(IOException e) {
            System.out.println("Exception:"+e);
        }
    }
}
```