

פיתוח מערכות תוכנה מבוססות Java
הבטי תקשורת במערכות תוכנה

אוהד ברזילי
מרס 2010

בית הספר למדעי המחשב
אוניברסיטת תל אביב

על סדר היום

- תקשורת בין מחשבים
- מיקומה של התקשורת ביישום
- השוואה של 3 ארכיטקטורות תקשורת
 - Sockets
 - RMI - תקשורת ללא תקשורת
 - Web Applications

Internet for dummies

- האינטרנט היא רשת מחשבים (= מחשבים + חוטים)
- לכל מחשב ניתנת כתובת (IP Address) המורכבת מ-4 מספרים בין 0 ל-255
- כל תוכנית הרצה במחשב מקבלת מספר שלוחה (port) בין 0 ל-65,535
- ניתן ליצור קשר עם מחשב המחובר לרשת ע"י ציון כתובתו ופניה אליו בהודעה המתאימה לפי הפרוטוקול
- בדרך כלל ע"י שימוש בתוכנית המממשת את הפרוטוקול

Internet != Web

- **האינטרנט** (תקשורת בין-רשתית) הומצא בשנות ה-60 של המאה הקודמת וחידושו בכך שהוא מאפשר לשני מחשבים לתקשר ביניהם בין אם הם מחוברים ישירות ובין אם הם מחוברים דרך מחשבי ביניים
- יישומים: email, ftp, VoIP, Web
- ה-**Web** הומצא רק ב-1989. זוהי רשת של מסמכים (אתרים) המכילים קישורים (לינקים) זה לזה. המסמכים עשויים להיות מאוחסנים במחשבים שונים המחוברים לאותה אינטרנט, וניתנים לצפייה בעזרת תוכנה מיוחדת (הדפדפן)
- רוב ההתייחסויות בעברית ל**אינטרנט** הם בעצם התייחסויות ל**Web** (מה המקבילה בעברית?)

תקשורת ב Java

■ תוכנית Java מתקשרת עם העולם החיצוני ע"י זרמים (streams)

■ עם המקלדת:

```
InputStreamReader in = new InputStreamReader(System.in);  
BufferedReader bin = new BufferedReader(in);  
String text = bin.readLine();
```

■ עם קובץ:

```
InputStreamReader in = new InputStreamReader(  
    new FileInputStream("foo.txt"));  
BufferedReader bin = new BufferedReader(in);  
String text = bin.readLine();
```

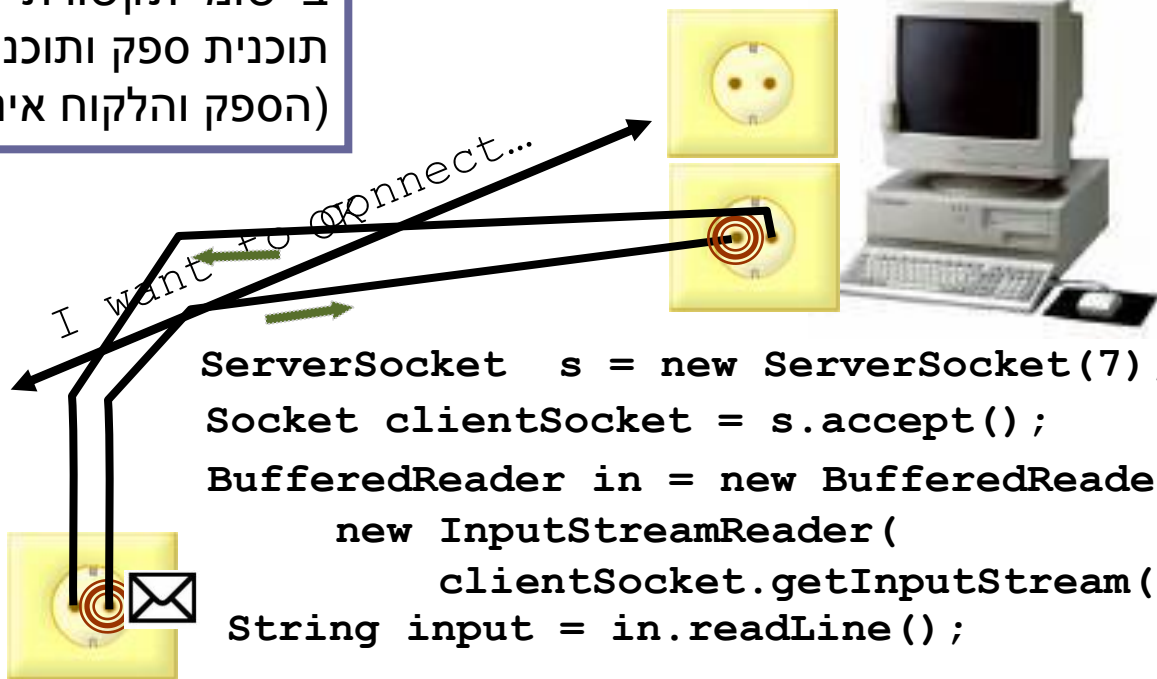
■ ננסה להשתמש באותה גישה בדיוק כדי לתקשר עם **תוכנית מחשב אחרת** (אולי במחשב אחר). לשם כך משתמש במחלקה **Socket** (שקע)

■ ב Java קיים חוסר סימטריה בין **יוזם ההודעה הראשונה** (לקוח - Client) ובין הצד השני (שרת - Server)

דוגמא: איך נשלח מחרוזת בין שני מחשבים?

ביישומי תקשורת יש לכתוב שתי תוכניות:
תוכנית ספק ותוכנית לקוח.
(הספק והלקוח אינם מריצים את אותה התוכנית)

www.tau.ac.il (132.67.250.239)



```
ServerSocket s = new ServerSocket(7);  
Socket clientSocket = s.accept();  
BufferedReader in = new BufferedReader(  
    new InputStreamReader(  
        clientSocket.getInputStream()));  
String input = in.readLine();
```

```
Socket s = new Socket("www.soft1.co.il", 7);  
PrintWriter out = new PrintWriter(s.getOutputStream(), true);  
out.println("hello there!");
```

פיתוח מערכות תוכנה מבוססות Java
אוניברסיטת תל אביב

EchoClient

```
package examples.sockets;

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

public class EchoClient {
    public static void main(String[] args) throws IOException {

        Socket echoSocket = null;
        PrintWriter out = null;
        BufferedReader in = null;

        try {
            String hostName = args[0];
            echoSocket = new Socket(hostName, 7);
            out = new PrintWriter(echoSocket.getOutputStream(), true);
            in = new BufferedReader(
                new InputStreamReader(echoSocket.getInputStream()));
        } catch (UnknownHostException e) {
            System.err.println("unkown host");
            System.exit(1);
        } catch (IOException e) {
            System.err.println("Couldn't get I/O for "
                + "the connection to host");
            System.exit(1);
        } catch (ArrayIndexOutOfBoundsException aiobe) {
            System.err.println("wrong usage: enter hostname");
            System.exit(1);
        }
    }
}
```

```
// establish an input stream to read from the standard input
BufferedReader input =
    new BufferedReader(new InputStreamReader(System.in));
```

```
String userInput;
```

```
while ((userInput = input.readLine()) != null) {
    // writer line to output stream
    out.println(userInput);
    out.flush();

    // print received echo result
    System.out.println("echo: " + in.readLine());
}
```

Buisness Logic

```
// close streams and socket
out.close();
in.close();
input.close();
echoSocket.close();
```

```
}
}
```



```
class EchoServer {
```

EchoServer

```
    public static void main(String[] args) {  
        ServerSocket serverSocket = null;  
        try {  
            serverSocket = new ServerSocket(7);  
        } catch (IOException ioe) {  
            System.err.println("Couldn't listen on port 7");  
            System.exit(1);  
        }  
    }
```

```
    Socket clientSocket = null;  
    try {  
        clientSocket = serverSocket.accept();  
    } catch (IOException ioe) {  
        System.out.println("Accept failed: 7");  
        System.exit(-1);  
    }  
}
```

```
    try {  
        PrintWriter out =  
            new PrintWriter(clientSocket.getOutputStream(), true);  
        BufferedReader in = new BufferedReader(  
            new InputStreamReader(clientSocket.getInputStream()));
```

```
        String input = in.readLine();  
        out.println(input);
```

Buisness Logic

```
        out.close();  
        in.close();  
        clientSocket.close();  
        serverSocket.close();  
    } catch (IOException ioe) {  
        System.err.println("Couldn't communicate with client");  
    }  
}
```

פיתוח מערכות תוכנה מבוססות Java

אוניברסיטת תל אביב

למה לא Sockets? (1)

- **בעיה:** שימוש בפרוטוקול לא סטנדרטי –
- התקשורת אמנם עוברת מעל פרוטוקולים סטנדרטים כגון IP, TCP, Ethernet ואולי אחרים (מאחורי הקלעים)
- אולם היישום עצמו מגדיר פרוטוקול משלו (שלוחה מספר 7, תוכן ההודעה הוא מחרוזת...)
- כדי להשתמש בתוכנית שלנו לקוחות (אנושיים) יצטרכו להריץ בעצמם את תוכנית הג'אווה: EchoClient

■ פרוטוקול תקשורת הוא **תאור** תחלופת הודעות בין מחשבים הכולל בין השאר את מבנה ההודעות השונות, סדר הודעות חוקי, הודעות שגיאה אפשריות ועוד...

■ התאור **אינו** מתייחס לשפת התכנות ששימשה לכתובת תוכניות המדברות בעזרת הפרוטוקול

למה לא Sockets? (2)

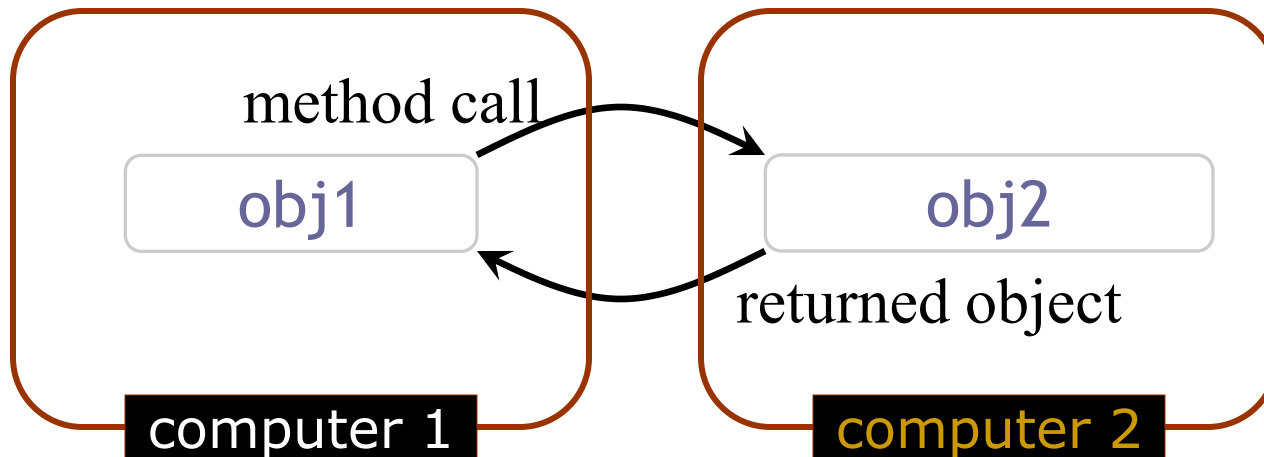
- הטכנולוגיה תופסת חלק נכבד מהקוד, בעוד הלוגיקה העסקית (מה שהתוכנית באמת עושה) זניחה
- הדבר בולט בישומים פשוטים
- התקשורת והמקביליות הן היבטים (aspects) של היישום ולא נרצה לערב אותן עם הלוגיקה העסקית

למה לא Sockets? (3)

- למרות העושר שמספקת Java, ביישומים רבים זה אינו מוקד הפיתוח (פרט אולי לשיקולי ביצועים)
- אנו מעוניינים לכתוב את היישום ללא תלות בפרוטוקול התקשורת שמעליו אנו עובדים
- יתר על כן, אנו מעוניינים לתאר את האינטראקציה בין שני עצמים **באותה צורה** בין אם הם נמצאים על אותה מכונה ובין אם הם מתקשרים דרך האינטרנט
- נזכיר כי אחד מיסודות התכנות מונחה העצמים הוא **העברת הודעות** (message passing), כאשר הכוונה להפעלה של מתודות על עצמים

“The network is the computer”*

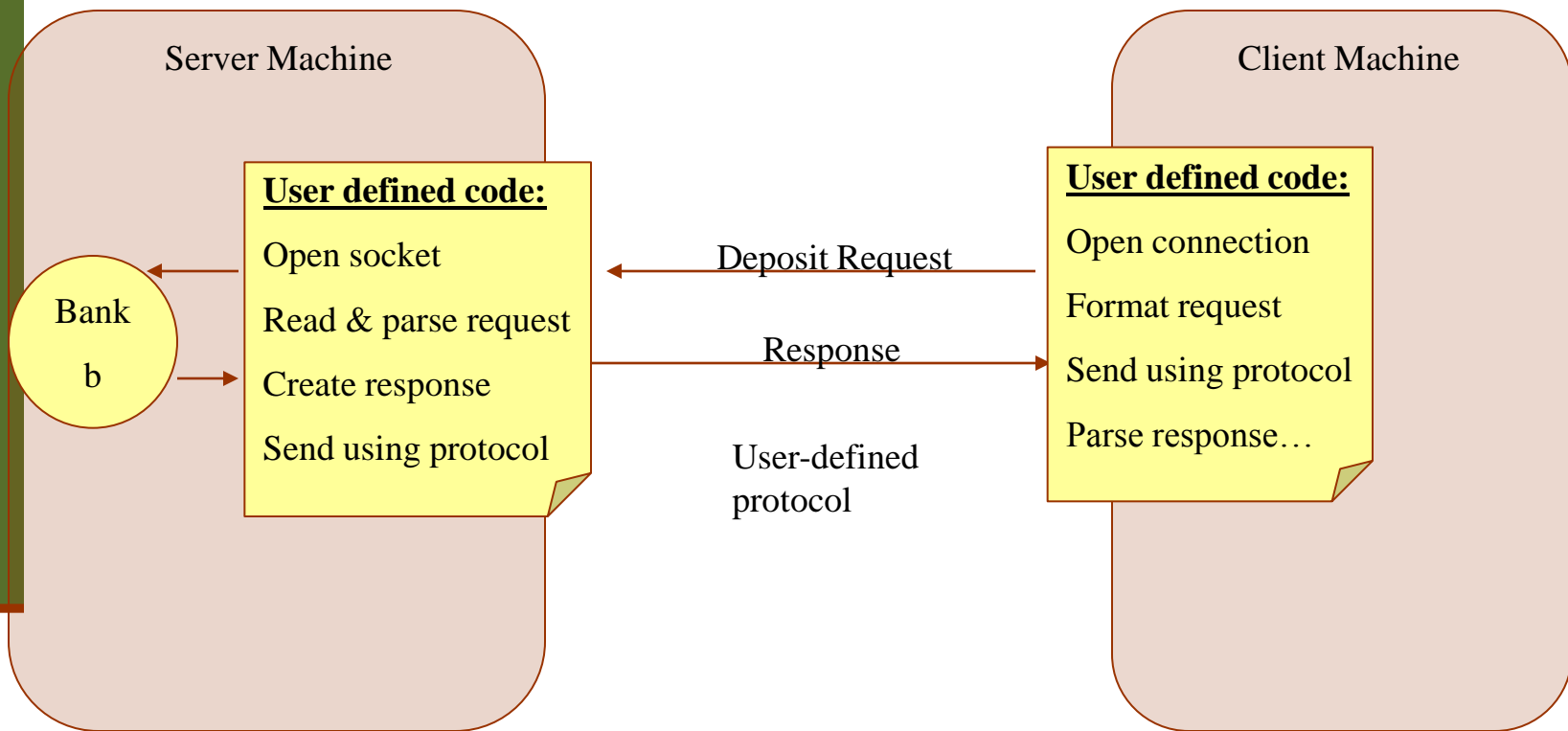
- Consider the following program organization:



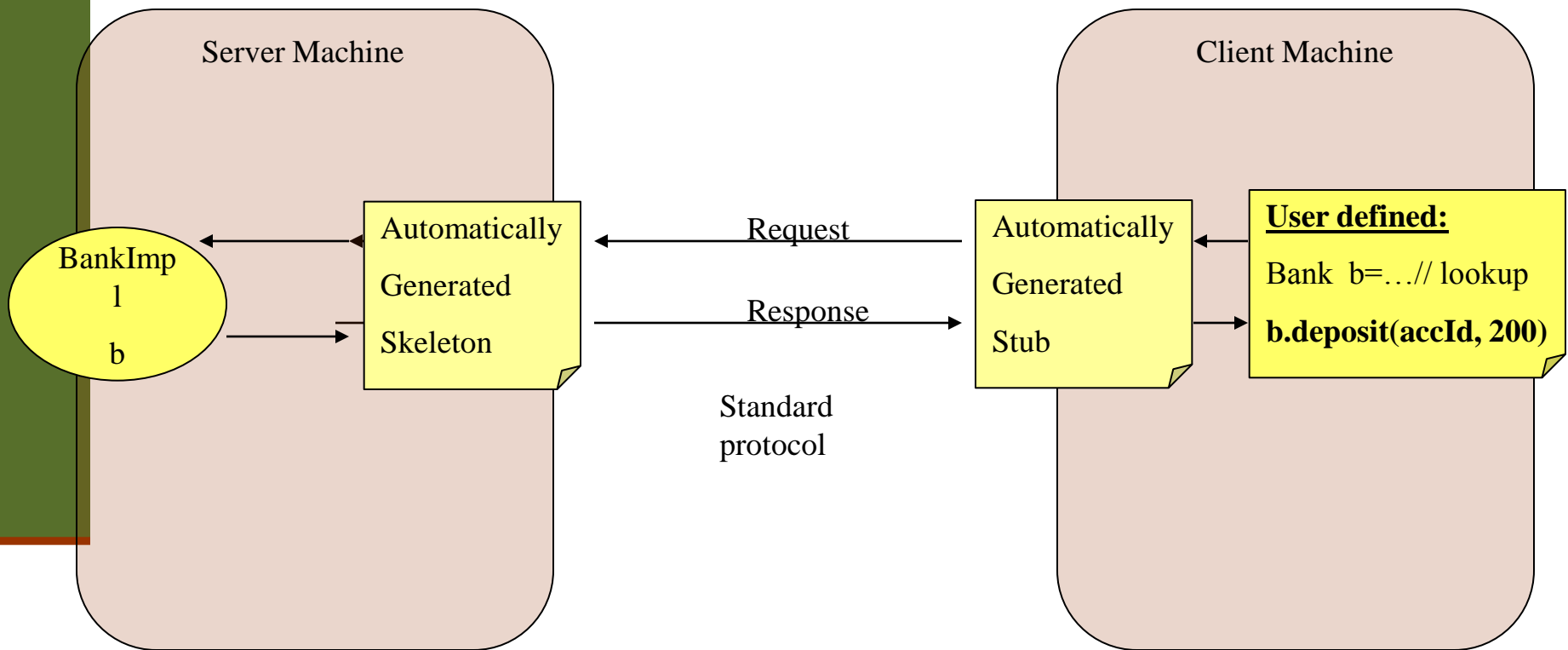
- If the network *is* the computer, we ought to be able to put the two classes on different computers (RPC)
- **RMI** is one technology that makes this possible

* For an opposing viewpoint, see <http://www.bbspot.com/News/2001/04/network.html>

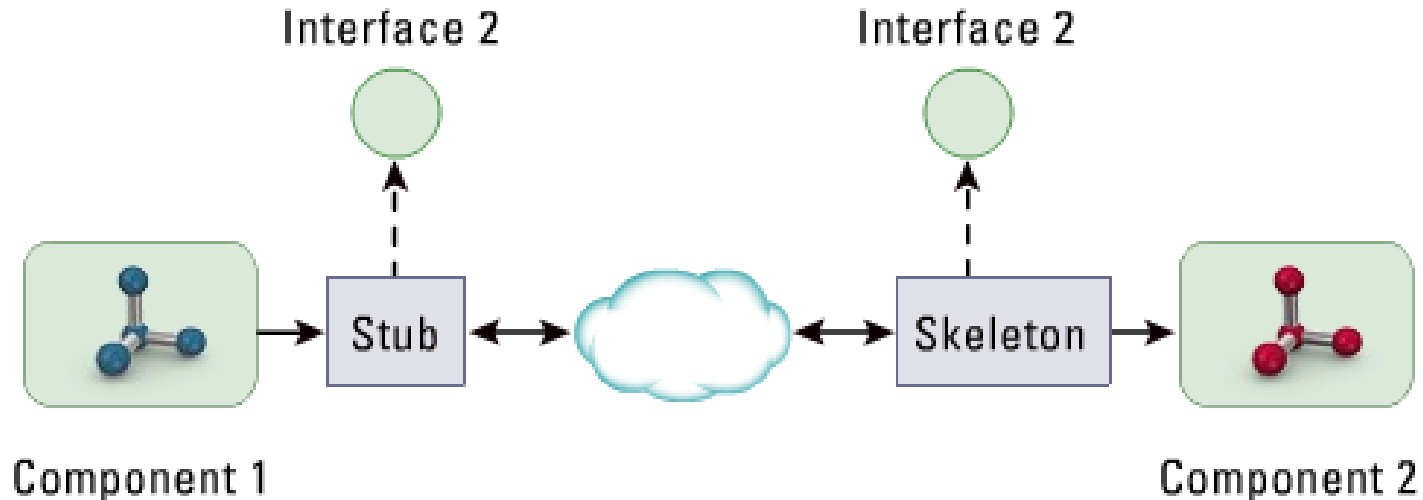
Non-RMI Approach



RMI Approach



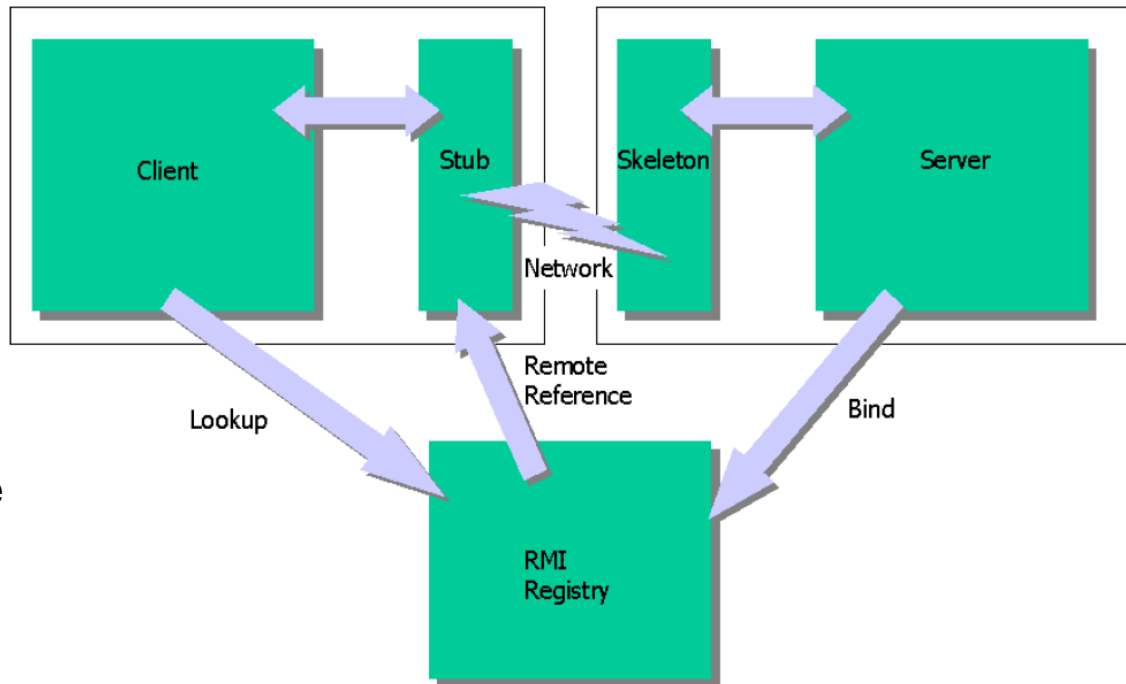
RMI Architecture



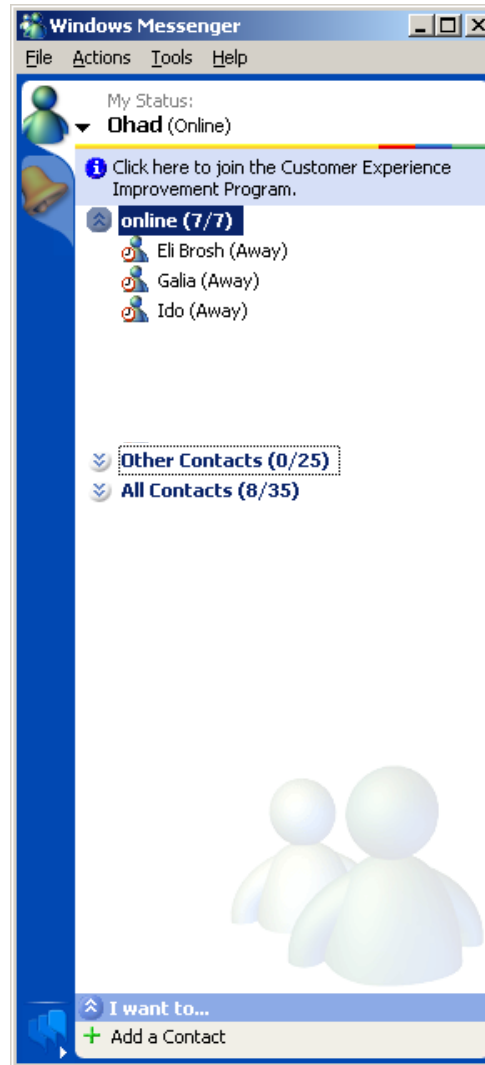
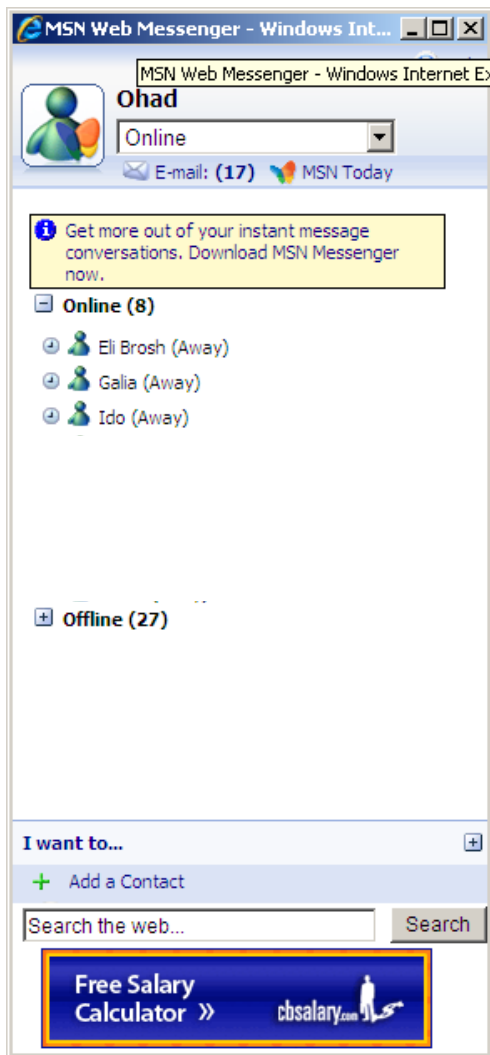
- If an interaction between two components is distributable, the application server must provide an RMI infrastructure by which the two components communicate
- Marshalling and unmarshalling of arguments and return values
- Passing distributed exceptions
- Passing security context and transaction context between the caller and the target

RMI Architecture

- Objects can be accessed remotely because they extend the Remote object
- RMI servers must be designed as an RMI server from scratch because it is not easy to take non-remote classes and make them remote
- A frequently used technique for creating sets of remote objects of the same type is to create a remote object factory that manages those objects and passes back references to those objects to clients



הדפדפן הוא הלקוח



בשנים האחרונות
מתפתחת מגמה של
פרוטוקולי תקשורת
מעל HTTP גם
ביישומים שאינם
כוללים גלישה
באינטרנט

מצאו את ההבדלים
בין לקוחות של MSN
Messenger

הדפדפן הוא הלקוח

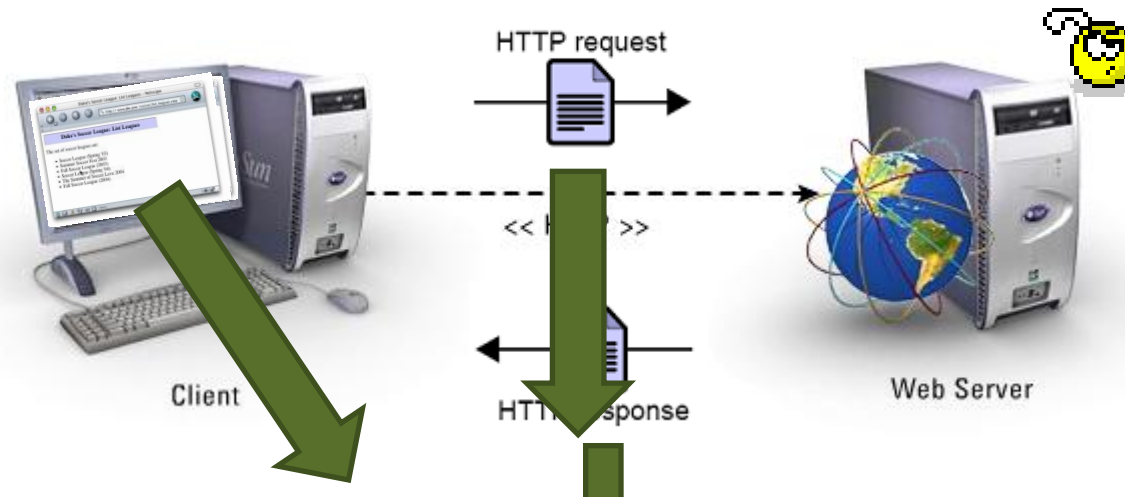
יתרונות הגישה:

- סטנדרטי
- זמינות
- אין צורך בהתקנה או הרשאות
- ריבוי כלים, סביבות ושירותים למפתח (למשל שרתים!)

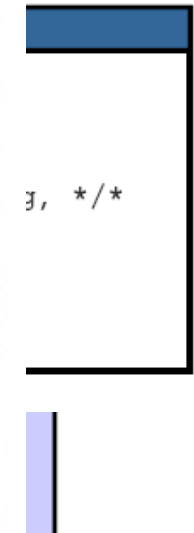
חסרונות:

- הדפדפן מיועד לעבודה מעל HTTP
- חווית משתמש (פרוטוקול STATELESS, GUI)
- קידוד ב HTML, CSS, JavaScript

פרוטוקול HTTP בגלישה בדפדפן



Request line
Request headers
Body
M



לתמרן את HTTP

■ תמרון צד השרת

- המטרה: שימוש בשרתי אינטרנט קיימים
- השרת מקבל הודעת GET אבל במקום להחזיר קובץ שהוכן מראש הוא מריץ פונקציה ומחזיר את התשובה שלה בתור "עמוד התשובה"
- זה בדיוק מה שעושה הספרייה Servlets ב-Java

■ תמרון צד הלקוח

- אסטרטגיה א': שימוש במחלקות Java המיועדות לעבודה עם אתרים: `URL`, `URLConnection`
- אסטרטגיה ב': כתיבת מנשק משתמש ב `html/javascript` ושימוש בדפדפן סטנדרטי