

# NetComputing :: The SocialHosting Project

Samuel Esposito and Jorn van de Beek

2010-04-01

## 1 Context

## 2 Problem statement

## 3 Solution design

The social hosting project means to revolutionize the traditional client-server pattern where there is a server running on one machine (hereafter referred to as the server-side) listening for HTTP-requests and one or more clients on other machines on the same network (referred to as the client-side hereafter) doing HTTP-requests on this server. The architecture of the software build for this project consists of different components, two of them running on the server-side and two of them on the client-side.

### 3.1 Server-Side HTTP-Server

On the server-side there will be a component listening for incoming HTTP-requests, just like in the traditional client-server pattern. This server-side HTTP-server however will not handle all HTTP-requests by itself. If it disposes over a pool of helper HTTP-servers running on other hosts on the network, it will propagate the request to the servers on these social hosts, thus balancing the load. The term we coined for this process is ‘social hosting’, because the pool of hosts is dynamic and consists of social hosts that offer hosting services on their own initiative to the sever-side HTTP-server. To manage these offered services, we need another component, which is discussed in the next subsection.

### 3.2 Sever-Side Registration-Server

As discussed above we need a component that registers and manages all incoming offers of hosting services. This process is handled by a server-side registration server that is continuously listening for offers from social hosts that want to share the load of HTTP-requests with the server-side HTTP-server. When a request is received, it is stored in a data structure that is available to the HTTP-server running on the server-side. This HTTP-server then propagates part of its HTTP-requests to the registered social host. Now the question arises, what are the components running on the social host? This is lined out in the next two subsections.

### 3.3 Client-Side Registration-Client

The social hosts we introduced earlier are actually the clients from the traditional client-server pattern. These clients can take the initiative to help the HTTP-server on the server-side out in handling HTTP-requests by registering themselves as social hosts. This registration is done by the registration client that makes a request to the registration server on the server-side, passing all information necessary for the future propagation of HTTP-requests.

### 3.4 Client-Side HTTP-Server

This leaves us only the handling of the propagated HTTP-requests by the social host. It comes as no surprise that this will be done by an HTTP-server on the client-side. This server disposes over (part of)

the same resources the server-side HTTP-Server uses to answer the HTTP-requests. Also it knows the origin of the requests, so it can send the answer directly to the client that made the request. This client in turn can make future requests directly to the social hosts until the social host again propagates the request to the server-side HTTP-server and we're back to square one.

In the next section the implementation of the different components is discussed.

## **4 Overview of realization**

In the previous section we explained of what components the social hosting software consists and how they interact. In this section we will elaborate each components implementation in the Java programming language.

To ease the communication and cooperation between the two components running on the server-side, we choose to implement them as different threads in the same java process, sharing a portion of memory for communication.

### **4.1 Server-Side HTTP-Server**

### **4.2 Server-Side RMI-Server**

### **4.3 Client-Side RMI-Client**

### **4.4 Client-Side HTTP-Server**

## **5 Evaluation**