An Implementation of a Peer-to-Peer Search System for Android

Overview

The aim of this project is to develop a simple Smartphone P2P Search system for Android. Android is an Operating System for Smartphones. The high level idea is that a **Query Peer** (i.e., some smartphone user denoted as **QP**) wants to know whether its “neighboring” smartphone users have an answer to **Q** (e.g., hold a file that has a filename with keywords posted in **Q**). To solve this problem, **QP** first connects to a server **S,** in order to retrieve a list of “neighboring” nodes. The neighbor list has the form of a communication tree **T** (you do not need to worry about how the tree is constructed, just assume that it is serialized somehow). Once obtained, **QP** connects to P0=**root(T)** posting **Q** and the remaining tree **T. P0** then forwards the **Q** and **T** to the remaining tree. The above procedure continuous recursively for **N** levels. Any peer receiving **Q,** conducts a local search and informs **QP** directly if an answer exists.

First, the query peer **QP** downloads the constructed tree **T** from server **S**:

```
-- CLIENT Connects to S (e.g., 192.168.1.10:65000)

SERVER: +OK READY –- welcoming message

QP CLIENT: GET T -- (note that QP listens on
192.168.1.10:65001 for subsequent steps)

SERVER: 192.168.1.10:65002, 192.168.1.10:65003, …]
-- a detailed explanation of the tree structure follows

-- QP CLIENT Now Closes the connection when EOF received
```

Now the **QP** Client performs a Neighborhood search, using T, starting out from the root of **T** that was denoted as **P0**.

```
P0 CLIENT: +OK READY  -- welcoming message

-- now QP sends P0 a) the level (initially 0 and
incremented every time it is received), b) the search
keywords Q={k1,k2,…,kn}, c) the IP:PORT of QN and d) the
remaining Tree T. The reason for including the IP:PORT of
QN (in c) is to allow peers to respond to QP if they have
an answer to Q.

QP CLIENT: SEARCH 0 | k1,k2,...,kn | 192.168.1.10:65001 |
192.168.1.10:65003, 192.168.1.10:65004, ...
```

Now P0 forwards recursively to P1

```
P1 CLIENT: +OK READY  -- welcoming message
P0 CLIENT: SEARCH 1 | k1,k2,...,kn | 192.168.1.10:65001 |
192.168.1.10:65004, ...
```

The above continuous recursively until N levels (according to the level parameter) have been traversed. Every node **Px** receiving **Q** checks whether its local data repository (e.g., files stored in a certain folder match all **|Q|** keywords). If yes, **Px** notifies **QP** directly.

```
-- For instance, in the following example P1 connects to
QP in order to deliver its result.

QP CLIENT: +OK READY  -- welcoming message

--  P1  now  delivers  its  result(s),  i.e.,  strings
r1,r2,...,rm to QN.

P1 CLIENT: RESULT r1,r2,...,rm -- welcoming message
```

If a peer in T is not responding for whatever reason you can disregard that branch of the tree.

Serialized Format of a Tree

The Following shows below each peer its designated parent peer. In reality, you should substitute the symbols with the respective IP:PORT pairs.

```
# List P2P Topology rooted at QP
# P0, P1, P2, P3
  QP, P0, P1, P2

# Star P2P Topology rooted at QP
# P0, P1, P2, P3
  QP, P0, P0, P0

# Arbitrary P2P Topology
# P0, P1, P2, P3
  QP, QP, P0, P2
```

Hardware/Software

For this project you will be provided **three (initially 2) real** HTC Desire Smartphones. The devices can be programmed with eClipse and JAVA (Android SDK provides an emulator for eClipse that can be used for debugging). After compiling your program with eClipse an APK (executable) file will emerge. The file can be copied and installed on the device (all programming info is here: http://developer.android.com/sdk/index.html).

User Interface Features:

- When your android program opens it displays 4 textboxes (Server Address, Own Address (disabled field), Levels-To-Search and a search box. A user can type a keyword list (seperated by commas, e.g., “red, audi, tt”) in search box. The UI also has one Submit and one Cancel Button. Hitting Submit should initiate the search process as explained above.
- Any arriving result (note that each peer must run a local socket server) must show up in a textarea of the user interface.

You might utilize TCP/IP for the above protocol

Please create an SVN repository for your project (see helpdesk) coined “AndroidP2P” (see http://helpdesk.cs.ucy.ac.cy/subjectview4.php?which=6627) Add yourselves, “dzeina” and “akonstan” to the requested list and ask your instructor to acknowledge the request.

Requested Demo

The requested demo must show how a keyword search can be conducted.