Internet and Intranet Applications and Protocols

Assignment 4: A Peer-to-peer System
Prof. Arthur P. Goldberg
Spring, 2004
Version: April 13, 2004
Due: May 4, 2004
This assignment is at

Introduction
For this assignment write a node for a peer-to-peer query processing system. You will learn about writing a distributed algorithm and a concurrent NIO client/server, and using UDP multicast.

Changes
As I’ve said previously, the specification of this assignment may change a little while you’re working on it. This is especially true of this assignment, since I’ve not given one like it previously. Be prepared to accommodate such ‘specification creep’.

Architecture
This peer-to-peer query processing system will involve multiple query processing nodes and a single centralized coordination site. We will implement and run some of the query processing nodes and the coordination site. You will implement and run a set of query processing nodes. All the query processing nodes will be connected into a network. We will use our components to grade yours.
Like Gnutella this peer-to-peer system will process queries by using query flooding. A query can originate at any node, and may be answered by any node.

Query Flooding
Your nodes should implement query flooding, which we discussed in class. The breadth of a flood is regulated by a hop count.
A node should employ NIO to handle concurrently receiving TCP connections, and sending and receiving datagrams and TCP messages. Use DatagramChannels, which are selectable channels for datagram-oriented sockets, a ServerSocketChannel, which is a selectable channel for TCP listening sockets, and a SocketChannel, which is a selectable channel for TCP sockets. Thus, a node will run just one thread. I passed out examples of NIO in the 3rd week. You may also look at the NIO code in Sun’s NIO Examples at http://java.sun.com/j2se/1.4.2/docs/guide/nio/example/. I think that TimeQuery.java and TimeServer.java would be most relevant.
The nodes adjacent to each node will be the members of a multicast group. A node forwards a query by sending the query to the multicast group for its adjacent nodes.
Message Format

Messages follow the format for headers in RFC 2822. All messages are terminated by a blank line (the string “CRLF CRLF”).

This section describes all the messages used this peer-to-peer system, including the fields for each message. Tokens used are defined in RFC 2822, or in the Section ‘Syntax’ below.

Configuration

When a node starts up it sends one CONFIGURE_REQUEST message via TCP to the coordination site. This TCP connection must be kept open to receive CONFIGURE_RESPONSE and JOIN_GROUP messages.

CONFIGURE_REQUEST:
Type: ‘CONFIGURE_REQUEST’
UniqueSID: atom

The unique system ID for the node sending this CONFIGURE_REQUEST.

A CONFIGURE_RESPONSE message is sent via TCP by the coordination site back to the node that sent the CONFIGURE_REQUEST. A node does not process QUERY_REQUEST messages until it has received a CONFIGURE_RESPONSE message.

CONFIGURE_RESPONSE:
Type: ‘CONFIGURE_RESPONSE’
Status: 3DIGIT

The status of this CONFIGURE_RESPONSE.
200 indicates OK; no other fields are included if the status is not OK.
401 indicates “No such unique system ID”
AdjacentNodes: IPaddress

The multicast IP address for the nodes adjacent to the node that sent the CONFIGURE_REQUEST.
MaxHops: number

The maximum number of hops for a QUERY_REQUEST message.

JOIN_GROUP messages are sent via TCP by the coordination site back to a node that has sent a CONFIGURE_REQUEST. A node can receive a JOIN_GROUP message at any time.

JOIN_GROUP:
Type: ‘JOIN_GROUP’
Status: 3DIGIT

The status of this JOIN_GROUP.
200 indicates OK; no other fields are included if the status is not OK.
401 indicates “No such unique system ID”

---

1 Except you do not need to support ‘folding’ (section 2.2.3)
AnAdjacentGroup: IPaddress
    A multicast IP address to which one of this node’s adjacent nodes will send.

Query Processing
A QUERY_REQUEST message is sent as a UDP multicast to a node’s adjacent nodes.

QUERY_REQUEST:
Type: ‘QUERY_REQUEST’
Hops: 3DIGIT
    The number of additional hops allowed for this QUERY_REQUEST.
    Decremented when a QUERY_REQUEST is forwarded.
UID: atom
    A unique ID for this QUERY_REQUEST. Each time a new QUERY_REQUEST
    is created it must have a unique UID.
Sender: IPaddress:number
    The IPaddress:port on which the sender of this QUERY_REQUEST will listen for
    a QUERY_RESPONSE.

A QUERY_RESPONSE message is sent via TCP by the node with the answer to the
sender of the QUERY_REQUEST.

QUERY_RESPONSE:
Type: ‘QUERY_RESPONSE’
Hops: number
    The number of additional hops that existed on the QUERY_REQUEST.
UID: atom
    The unique ID on the QUERY_REQUEST.

Syntax
IPaddress = 3DIGIT. 3DIGIT. 3DIGIT. 3DIGIT
number = 1*DIGIT

Libraries
Your code must be written in Java. The networking code must be written java.net. I do
not recommend it, but if you wish you may use java string processing libraries or parsers
to parse the messages.

Grading
You will run some nodes of your server on some hosts that we can access.
To run your system
1. Go to our Web site (URL to be provided later), to obtain your unique system number
   (UniqueSID), and the host:port of our coordination site.
2. Run your nodes. Each node should send a CONFIGURE_REQUEST message to the
   coordination site, and handle the CONFIGURE_RESPONSE message it receives.
   The coordination site will configure a connected network of nodes.
3. Issue and handle some queries—the details are to be determined later.
4. Return to our Web site to receive feedback on your system.

Please do not wait for our automated system to begin your development and testing. You should be able to do extensive work without it.
Your server should follow the code quality guidelines we discussed. We will evaluate this by reading your code. Grades will be allocated as follows:

<table>
<thead>
<tr>
<th></th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality</td>
<td>85</td>
</tr>
<tr>
<td>Code quality guidelines</td>
<td>15</td>
</tr>
</tbody>
</table>

**Handing In Your Peer-to-peer System**

When you’re satisfied with your Peer-to-peer system please hand in the following:
- Your source code (with your name in all java files) in a jar or zip file named with your name, “Firstname Lastname”.
- A verbose log of your Peer-to-peer system’s best run from the Peer-to-peer system Grading/Testing program on the web site.
- A non-verbose log of your Peer-to-peer system’s best run.

Please package these three files in a zip file, name the file with your name, “Firstname Lastname”, and email it to me.