CS 360 Spring 2012 PUIM Command Line Client and Library **Assignment Three**
March 21, 2012
DUE: April 16, 2012, 2:15pm **50 points**

For this project, you need to produce an Android chat client that will work with the instructor's server.

There is no C server for this project, only a Java server.

The PUIM (Pacific University Instant Message) has changed!

PUIM Version 3
The protocol messages are specified below.

Each line of text is a maximum of 1024 characters including the \n.
Each line must be terminated by a \n.
Lines marked with C are produced by the client. Lines marked with S are produced by the server. Neither C nor S are transmitted.

For ERR messages, any string after the ERR is optional.

Session Startup.
C HELLO username
S OK | ERR Username already in use

Session Disconnect
C DISCONNECT
S OK | MSG FROM:username

Send a message to another user.
C MSG TO:username
S OK | ERR Username not found | MSG FROM:username
C <TEXT terminated with single . on a line>
C .
S OK

Receive a message from another user.
S MSG FROM:username
C OK
S <TEXT terminated with single . on a line>
S .
C OK

Data messages
C DATA TO:username
S OK | ERR Username not found | ERR unsupported
C CONTENT/TYPE: *mime/type*
C <TEXT terminated with single . on a line> (base64 encoded image)
C .
S OK

S DATA FROM:username
C OK | ERR Username not found | ERR unsupported
S CONTENT/TYPE: *mime/type*
S <TEXT terminated with single . on a line> (base64 encoded image)
S .
C OK

CONFLICTS

If a server and client both initiate the send of a message (text, data, icon) the CLIENT must back off and accept the server's message by sending an OK. The client then attempts to resend its message after the server completes its transaction.

If the client requests to DISCONNECT but receives [MSG|DATA|ICON] FROM:username rather than OK, the client must receive the full message and after the final OK then try again to DISCONNECT. Once a server has received DISCONNECT (even if that DISCONNECT is interrupted by a message from the server) no new messages [MSG|DATA|ICON] can be sent to the client.

Line Length

If any line of data is too long the server will send the message: **ERR too long** and terminate the current transaction (stop receiving the message or data).

Escaping a leading period

Using a single period on a line as a message terminator prevents the user from sending a single period on a line as part of a message. This is unacceptable. To remedy this, we will adapt the procedure outline in the SMTP RFC:
http://tools.ietf.org/html/rfc5321#section-4.5.2

- Before sending a line of **text**, the client checks the first character of the line. If it is a period, one additional period is inserted at the beginning of the line.

- Before displaying a line of **text**, the client checks the first character of the line. If that character is a period, the first character of the line is deleted.

- Note that the terminator (.\n) is not part of the **text** and so does not change.

- The server has no role in this change.

ADDITIONS

Your client must return
ERR unsupported
to any unknown message type. For example:

S DOG:beagle
C ERR unsupported

Send an icon to the server. Icons should be BMP, PNG, GIF, JPG. The server does not enforce icon formats. Clients are not required to display every type of icon. Icons are recommended to be 32x32 or 24x24.

C ICON
S OK | ERR unsupported
C CONTENT-TYPE: *image/png*
C <TEXT terminated with single . on a line> (base64 encoded image)
C .
S OK

Request an icon from the server.
C REQICON:username
S ICON FROM:username | ERR Username not found | ERR unsupported | ERR no icon
C OK
S CONTENT-TYPE: image/png
S <TEXT terminated with single . on a line> (base64 encoded image)
S .
C OK | ERR unsupported mime type

Push an icon to the client
S ICON FROM:username
C OK | ERR unsupported
S CONTENT-TYPE: image/png
S <TEXT terminated with single . on a line> (base64 encoded image)
S .
C OK | ERR unsupported mime type

The server will send icons to the client at random intervals.

You may receive an icon from a user you have never communicated with. You may also receive an icon from yourself.

The Android Client

The Android client must allow the user to specify the server address, server port, and username. The client must allow the user to set the icon to an image file on the device. The client must accept icons from the server and display those icons alongside a message from the appropriate user. If a user that has sent a message to the client does not have an icon, a default icon or no icon may be displayed.

The user must be able to send and receive text and data messages from any user. This can be done in one unified interface similar to the Command Line Client, or you may choose to have a separate Window/View for each distinct conversation and provide some means of switching between conversations.

A TabHost/TabWidget setup *may* be useful here or a ListView.

For an example of the former:
http://developer.android.com/resources/tutorials/views/hello-tabwidget.html

For an examples of the latter:
http://stackoverflow.com/questions/541966/android-how-do-i-do-a-lazy-load-of-images-in-listview/3068012#3068012
http://developer.android.com/resources/tutorials/views/hello-listview.html
http://android.amberfog.com/?p=296

If a data message is received, you must use an implicit Intent to display that content. You may need to write the data message to a temporary file. Be sure to enable SD access and Internet access in your application. Try to delete the temporary files after the Intent finishes.
http://android-developers.blogspot.com/2011/08/horizontal-view-swiping-with-viewpager.html
http://thepseudocoder.wordpress.com/2011/10/05/android-page-swiping-using-viewpager/
The client should use Android 2.3.3.

Base 64 Encoding.

The package android.util.Base64 exists to encode data on Android. You can choose to bundle the Apache libraries with your application, or use the built-in Android package.

A simple server will be provided tonight! You can run this server as follows. The first command line argument is the port, the second argument is the loop delay:

bart$ java -jar PUIMSimpleServer.jar edu.pacificu.cs.cs360.PUIM.PUIMServer 12349 500

Eclipse Project

Name your Eclipse projects **CS360_PUIM_AndroidClient_PUNetID.**

Recommendation
Add the commands ICON and REQICON to the command line client (where REQICON saves the icon to a local file, just like DATA) to allow you to more easily debug the client library, and then build your Android client.