Chat.app
by Andrew Loewenstern
Cube Technologies, Inc.
andrew@cubetech.com

Chat.app is a really simple program I wrote at a Houston Area NeXT User Group meeting to show our user group how to write apps using Distributed Objects. It is a multiuser chat program that lets clients connect to a "server" and share messages with eachother.

The server app is the same as the client app - if you try to connect to a non-existant server on your machine, your app will be the server. The system is minorly "fault-tolerant" in that clients can connect and disconnect at will without disturbing other clients. If the server terminates, all of the clients will terminate as well.

The code is very simple and to the point. I think I counted 50 lines of real code. Every line is commented. There are probably more lines in this RTF document.

This app is totally free. Use it and learn from it. Distributed Objects are way cool, and I hope every NeXT developer starts putting them into their apps. Even non-client/server apps can benefit by providing it's services through a distributed object. If you modify this app and make cool enhancements to it, I'd appreciate it if you would drop me a line via e-mail...

How it works:

The app starts up and asks the user for a hostname to look for the server on. Then the app tries to connect to the server on that host. If it does not find the server, then the app assumes you want to be the server and registers the application delegate with the root name server. By doing this, other apps can query the name server to find your registered object.

When a client starts up, it will immediately find the server on the host given by the user if a server is really running. It will then send a checkIn: message to the server. The server will add the id of the client to a list object.

When a client wants to post a message, it sends an acceptMessage:from: obj-c message to the server with a string containing the text, and a string containing the user-name of the client. The server then prints the incoming message to it's conversation window, and sends an identical acceptMessage:from: obj-c message to each client. Each client will then print the message to it's conversation window. There is an "if" statement to prevent the clients from sending messages to the other clients.

If a client quits, it sends a checkOut: message to the server. The server will then remove the client from the client list. This is unneccessary as the server will recieve a senderIsInvalid: message from the Distributed Objects system if a client goes down. This is just for fun... ;-) If the server suddenly

goes down, each client will recieve a senderIsInvalid message from the Distributed Objects system and will proceed to terminate itself.

Note, the text field where you type in your message is on a different window from the conversation text object so incoming messages don't interrupt your typing...

Goodies:

You could modify this for use on a local subnet only and get rid of the annoying "hostname" panel by commenting out the [self getHostName] and substituting the line:

```
server = [NXConnection connectToName:"ChatServer"
            onHost:[hostName stringValue]];
```

with

```
server = [NXConnection connectToName:"ChatServer" onHost:"*"];
```

This will search the entire subnet looking for the server. If it doesn't find the server, the app will become the server... With this in place, you can make the app even more fault tolerant by having a client become the new server if the current server is about to exit. Then the new server would register

itself, and inform each client of the new server...

This app works over Internet!! You can have a multiuser chat conference with all of your NeXT friends around the world for far less than an AT&T conference call would cost!

enjoy!!
andrew