Project Report
Group Tortoise

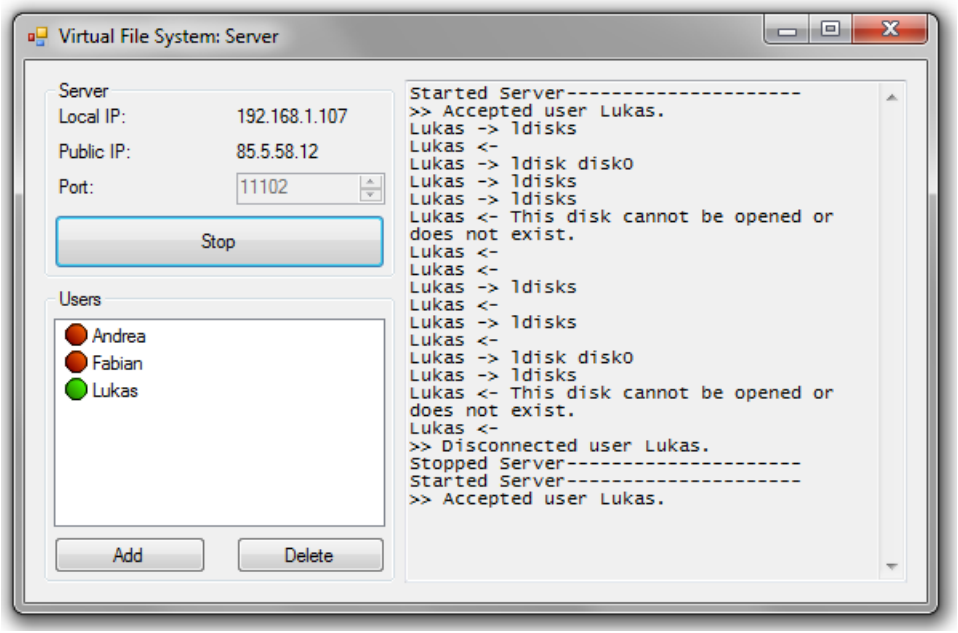
Java and C# in depth, Spring 2014

Lukas Häfliger
Fabian Meier
Andrea Canonica

May 19, 2014

1 Client and Server

For this part of the project we made two new user interfaces, a client interface and a server interface. On startup, the user can choose in which mode the application should run. If he chooses "Client", a simple dialog will open where he can specify the IP and port of the server, as well as his login. If the connection is successful he can open an Explorer Window (which is the same as in part 2) or he can choose to work offline.
If the application is started as a server, a new server window opens. Here the administrator can start and stop the server and register new users. All registered users are displayed on a list which also indicates the online state of a user. On the right side a log is displayed which lists all transactions.



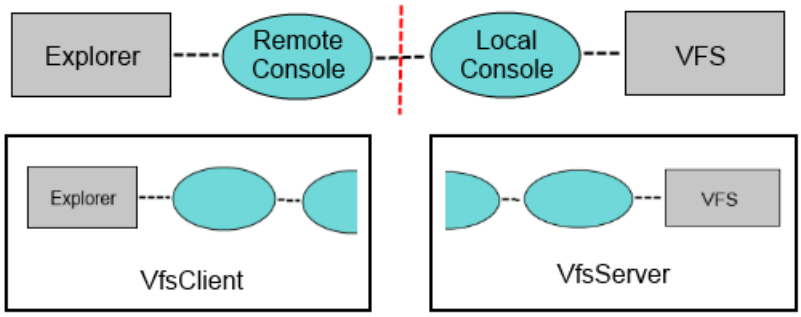
1.1 Design

This part of the project builds directly on the two previous parts (which we separated into their own assemblies, thus they both still run independantly). Since in part 2, the interface between the *VfsExplorer* and the *VfsManager* (part 1) was already designed very narrow, the step into a distributed environment was reasonably easy. In the graphic below you see that the manager and explorer were only connected by the classes *LocalConsole* and *RemoteConsole*, which only communicate command line like strings. To introduce networking we added the two classes *LocalConsoleAdapther* and *RemoteConsoleAdapter*, which are subclasses of the corresponding Console-classes. These two classes mainly contain networking code to send messages, which originally were console in- and outputs, to each other. To control this new setup we also added a *VfsClient* and a *VfsServer* GUI.
The distribution of the VFS is organized as follows:

- The server has a copy of all virtual disks locally. when an user commits a command it will be queued into the VFS and then executed. If it was successfull, the command is sent back to all machines of the same user.
- Each machine of a user has its own local copy of a disk. An issued command is first sent to the server, and it is only locally applied if the

server sends it back. This prevents race conditions between simultaneous commands.

- Import and export commands are treated separately, since they require more communication than just the basic string messages.



1.1.1 VfsServer

This class is a GUI which allows the administrator to set up a server. It initiallizes a new manager instance and connects it's LocalConsole to a new RemoteConsoleAdapter instance. The manager was not modified for this environment and still works the same way as before.

1.1.2 RemoteConsoleAdapter

This class represents the server-side network interface. When the adiministrator starts the server, it opens a new TCP listener and accepts incoming client connections. All of the serverside encoding and decoding of messages is handled by this class.

1.1.3 VfsClient

This class is the client GUI which users use to connect to the server. When the user starts a connection atempt, this class starts a new LocalConsoleAdapter instance. If the connection is succesful and the user clicks the "Explorer" button this class creates and shows a new VfsExplorer and connects it's RemoteConsole to the new LocalConsole Adapter. The explorer required almost no modification to work in this mode, only one Boolean indicator to handle the adding of new disks.

1.1.4 LocalConsoleAdapter

This class handles the client-side of the network interface. All of the clientside encoding and decoding of messages is handled by this class.

1.1.5 Networking Messages

We defined 10 network message types used for this project. The Message type is encoded as the first Byte of the sent data. Most messages contain only an encoded string after that, except the File Transfer messages, which also contain a whole file. To reduce bandwith usage we restrict import and export of files larger than 4MB in this mode.

- Wrong User or Password
- Accept User
- Command
- Success Message
- Error Message
- Query
- File Transfer (import)
- File Transfer (export)
- End

1.2 Requirements

Our VFS Network provides the following features:

1.2.1 Server

- **Execute core commands without writing to a console.**
  We simply provided a GUI with buttons which got rid of the necessity of having the user write the commands himself.
- **Single and multiple selections of files and folders.**
  The *Explorer* contains a field called *selectedNames* in which the names of the selected entries are stored (as strings). To select an entry, one can click on it with the mouse and by holding the Ctrl-Key multiple entries can be selected.

- **Support for keyboard navigation.**
  We allow keyboard navigation by typing an address in the *addressbar*, when a directory is selected or when an item in the treeview is selected. After each of these cases, *Enter* has to be pressed. Navigation into the parent folder is achieved with the Backspace-Key.

- **Support for mouse navigation.**
  Mouse navigation is done by either clicking on an item in the *treeview* or by double clicking a directory in the *listview*.

- **Responsive UI. (Bonus feature)** As shortly mentioned before, the *LocalConsole* has a queue of tasks and a thread which constantly dequeues the tasks and executes them. Therefore the user can keep sending commands which then simply will be added to the queue. We implemented two more methods methods in the *VfsManager* to keep the navigation and listing of the entries correct. In simple words, whenever we should list the entries we check if the entry is loaded and only then it is displayed. Otherwise we enqueue the task. The procedure executed by the Worker-Thread can be found in listing 3.

- **Drag and Drop. (Partial) (Bonus feature)** So far, we allow copying and importing when dragging files from the Host-System into the *listview* or from the *listview* to the *treeview*.