Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Кафедра «Компьютерные технологии»

И.А. Вотинов, Б.З. Хасянзянов, А.А. Шалыто

Система сетевого файлообмена
(аналог системы *NetBIOS*)

Проектная документация

Проект создан в рамках
«Движения за открытую проектную документацию»
http://is.ifmo.ru

Санкт-Петербург
2007

Оглавление

Введение ... 3
1. Постановка задачи ... 3
2. Описание команд протокола ... 3
3. Автоматы ... 5
  3.1. Клиентский автомат (*A1*) ... 5
    3.1.1. Описание ... 5
    3.1.2. Схема связей ... 5
    3.1.3. Описание состояний ... 5
    3.1.4. Описание событий ... 6
    3.1.5. Список выходных воздействий ... 6
    3.1.6. Список параметров при переходах ... 7
    3.1.7. Граф переходов ... 7
  3.2. Серверный автомат (*A2*) ... 8
    3.2.1. Описание ... 8
    3.2.2. Схема связей ... 8
    3.2.3. Описание состояний ... 8
    3.2.4. Описание событий ... 9
    3.2.5. Список выходных воздействий ... 9
    3.2.6. Список параметров при переходах ... 9
    3.2.7. Граф переходов ... 10
4. Реализация ... 10
  4.1. Интерпретационный подход. ... 10
  4.2. Компиляционный подход. ... 12
  4.3. Запуск программы ... 14
Заключение ... 14
Литература ... 14
Приложение. Демонстрация проекта ... 15

Введение

Данный проект демонстрирует возможности автоматного программирования в сфере разработки сетевых приложений. В качестве основного инструментального средства был использован *UniMod*, который представляет собой надстройку над открытой средой разработки *Eclipse.*

Данный проект является аналогом систем *NetBIOS* (*Windows*) и *Samba* (*Linux*).

1. Постановка задачи

Требуется создать систему обмена файлами по сети между любым количеством пользователей. Система должна состоять из серверной и клиентской части.

Серверная часть (далее «сервер») должна предоставлять пользователям сети информацию о файлах и папках доступных для копирования, и давать возможность копировать их. Несколько клиентов должны иметь возможность одновременного подключения к одному серверу.

Клиентская часть (далее «клиент») должна позволять получать информацию о файлах и папках сервера, доступных для копирования, и давать возможность копировать их.

Для взаимодействия между сервером и клиентом должен использоваться транспортный протокол *TCP*, что позволит осуществлять взаимодействие между сервером и клиентом не только в локальной сети, но и в сети *Internet*.

2. Описание команд протокола

Команды протокола обмена файлами по сети определяют функции:

- установки связи между клиентом и сервером;
- передачи списка файлов, доступных на сервере для передачи клиенту;
- передачи файла с сервера клиенту.

Команды, посылаемые клиентом, описаны в табл. 1.

Таблица 1

CLIENT_HELLO	Первая команда, посылаемая клиентом для инициации соединения. Клиент отправляет команду сразу после подключения к серверу.
CLIENT_FIRST_LIST_PART	Клиент данной командой запрашивает первую часть списка файлов в доступной папке на сервере. Инициируется передача списка по частям.
CLIENT_NEXT_LIST_PART	Клиент сигнализирует об успешном приеме очередной части списка файлов и запрашивает следующую часть.
CLIENT_FIRST_FILE_PART	Клиент запрашивает конкретный файл из списка доступных на сервере. Инициируется передача файла по частям.

CLIENT_NEXT_FILE_PART	Клиент сигнализирует об успешном приеме очередной части файла и запрашивает следующую часть.
CLIENT_CANCEL_FILE_DOWNLOAD	Клиент отменяет загрузку файла.
CLIENT_BYE	Клиент сообщает о необходимости завершить соединение.

Команды, посылаемые сервером, описаны в табл. 2.

Таблица 2

SERVER_HERE	Сервер посылает данную команду клиенту в ответ на команду «CLIENT_HELLO», сигнализируя о том, что соединение установлено, и сервер готов обрабатывать запросы.
SERVER_LIST_PART	Посылается сервером в ответ на команду клиента «CLIENT_FIRST_LIST_PART» или «CLIENT_NEXT_LIST_PART», если посылаемая часть не является последней. Содержит часть списка файлов в доступной папке на сервере.
SERVER_LAST_LIST_PART	Посылается сервером в ответ на команду клиента «CLIENT_FIRST_LIST_PART» или «CLIENT_NEXT_LIST_PART», если посылаемая часть является последней. Содержит часть списка файлов в доступной папке на сервере.
SERVER_FILE_PART	1) Сервер отвечает на команду «CLIENT_FIRST_FILE_PART»(в случае если файл состоит из нескольких частей) и начинает передачу запрошенного файла. В ответе содержится очередная часть запрошенного файла. 2) Сервер отвечает на команду «CLIENT_NEXT_FILE_PART» и отсылает очередную часть файла.
SERVER_LAST_FILE_PART	1) Сервер отвечает на команду «CLIENT_FIRST_FILE_PART»(в случае если файл состоит всего из одной части). В ответе содержится единственная часть запрошенного файла. 2) Сервер отвечает на команду «CLIENT_NEXT_FILE_PART» и отсылает последнюю часть файла.
SERVER_BYE	Сервер сообщает о необходимости завершить соединение.

3. Автоматы

3.1. Клиентский автомат (*A1*)

3.1.1. Описание

Этот автомат обеспечивает управление клиентской частью приложения.

3.1.2. Схема связей

Схема связей автомата *A1* изображена на рис. 1.

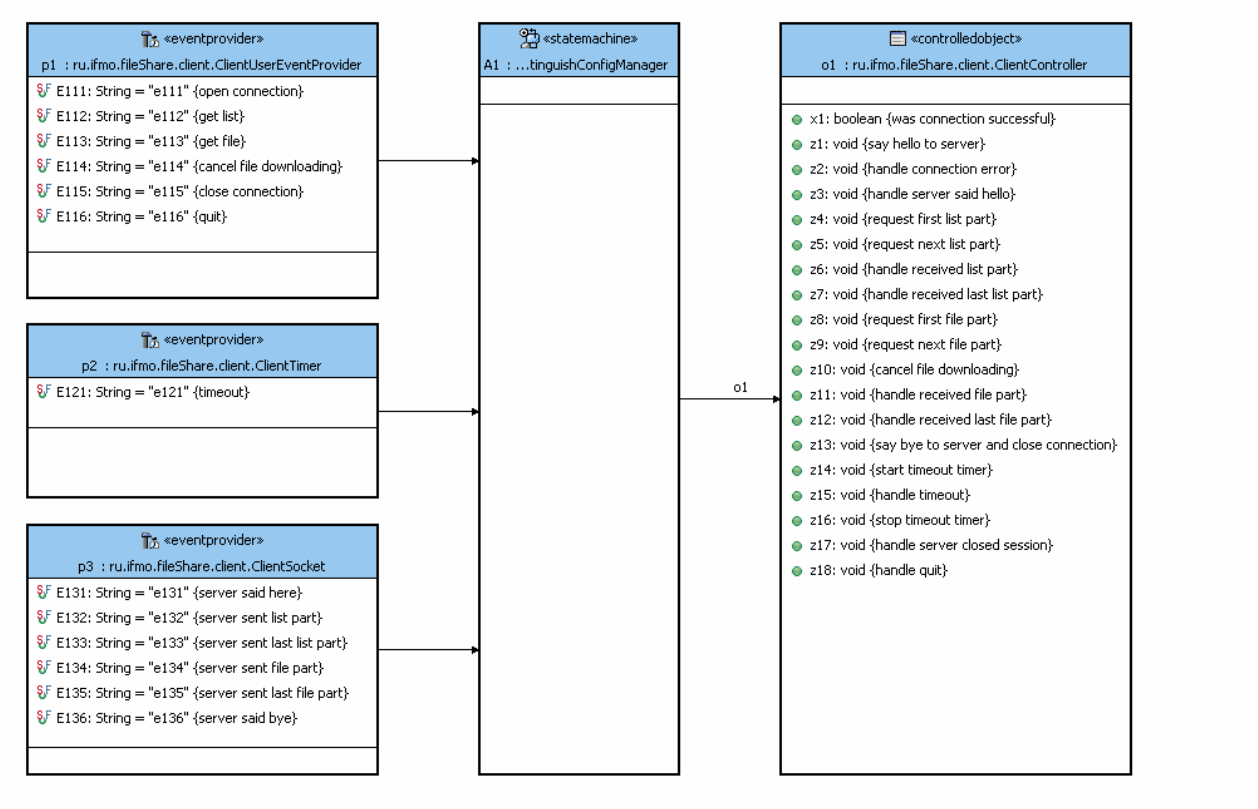


Рис. 1. Схема связей автомата *A1*

3.1.3. Описание состояний

Состояния автомата *A1* приведены в табл. 3.

Таблица 3

Disconnected	Нет соединения с сервером
Connected	Сеанс работы с сервером
Handshaking	Ожидание приветствия сервера
Idle	Ожидание команды пользователя
Getting List	Получение списка файлов
Getting File	Получение файла

3.1.4. Описание событий

События, относящиеся к поставщику событий окна приложения (*p1*), приведены в табл. 4.

Таблица 4

e111	Запрос подключения
e112	Запрос списка файлов
e113	Запрос файла
e114	Отмена загрузки файла
e115	Завершение соединения
e116	Закрытие приложения

События, относящиеся к поставщику событий таймера (*p2*), приведены в табл. 5.

Таблица 5

e121	Таймаут

События, относящиеся к поставщику событий соединения с сервером (*p3*), приведены в табл. 6.

Таблица 6

e131	Получено приветствие сервера
e132	Получена очередная часть списка файлов
e133	Получена последняя часть списка файлов
e134	Получена очередная часть файла
e135	Получена последняя часть файла
e136	Сервер закрыл соединение

3.1.5. Список выходных воздействий

Выходные воздействия клиентского объекта управления (*o1*) приведены в табл. 7.

Таблица 7

z1	Запрос подключения к серверу
z2	Обработка ошибки подключения
z3	Обработка успешного соединения
z4	Запрос первой части списка файлов
z5	Запрос очередной части списка файлов
z6	Обработка полученной части списка файлов
z7	Обработка полученной последней части списка файлов
z8	Запрос первой части файла
z9	Запрос очередной части файла
z10	Отмена загрузки файла
z11	Обработка полученной части файла
z12	Обработка полученной последней части файла
z13	Закрытие подключения к серверу
z14	Запуск таймера
z15	Обработка сигнала таймера
z16	Сброс таймера

z17	Обработка закрытия соединения сервером
z18	Завершение работы

3.1.6. Список параметров при переходах

Параметры клиентского объекта управления (*o1*) приведены в табл. 8.

Таблица 8

x1	Сетевое соединение успешно установлено

3.1.7. Граф переходов

Граф переходов автомата *A1* изображен на рис. 2.

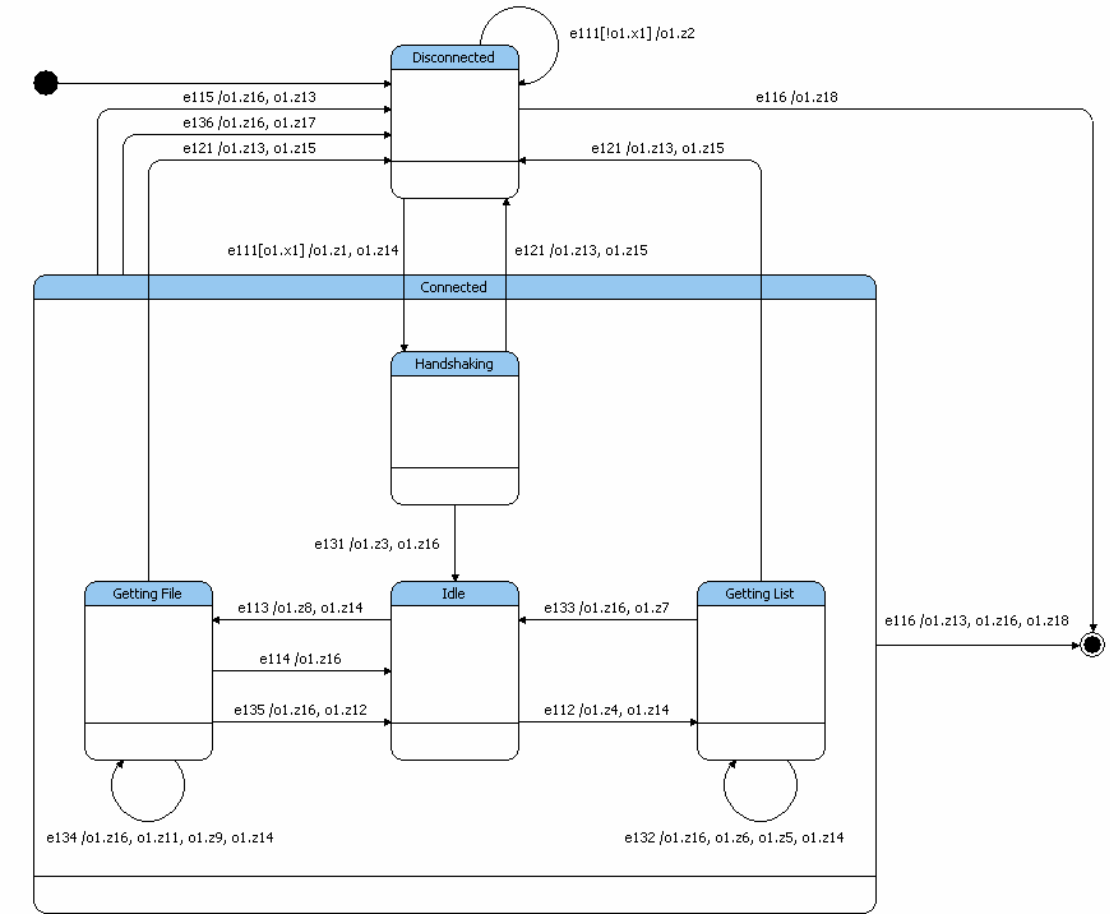


Рис. 2. Граф переходов автомата *A1*

3.2. Серверный автомат (*A2*)

3.2.1. Описание

Этот автомат обеспечивает управление серверной частью приложения.

3.2.2. Схема связей

Схема связей автомата *A2* изображена на рис. 3.

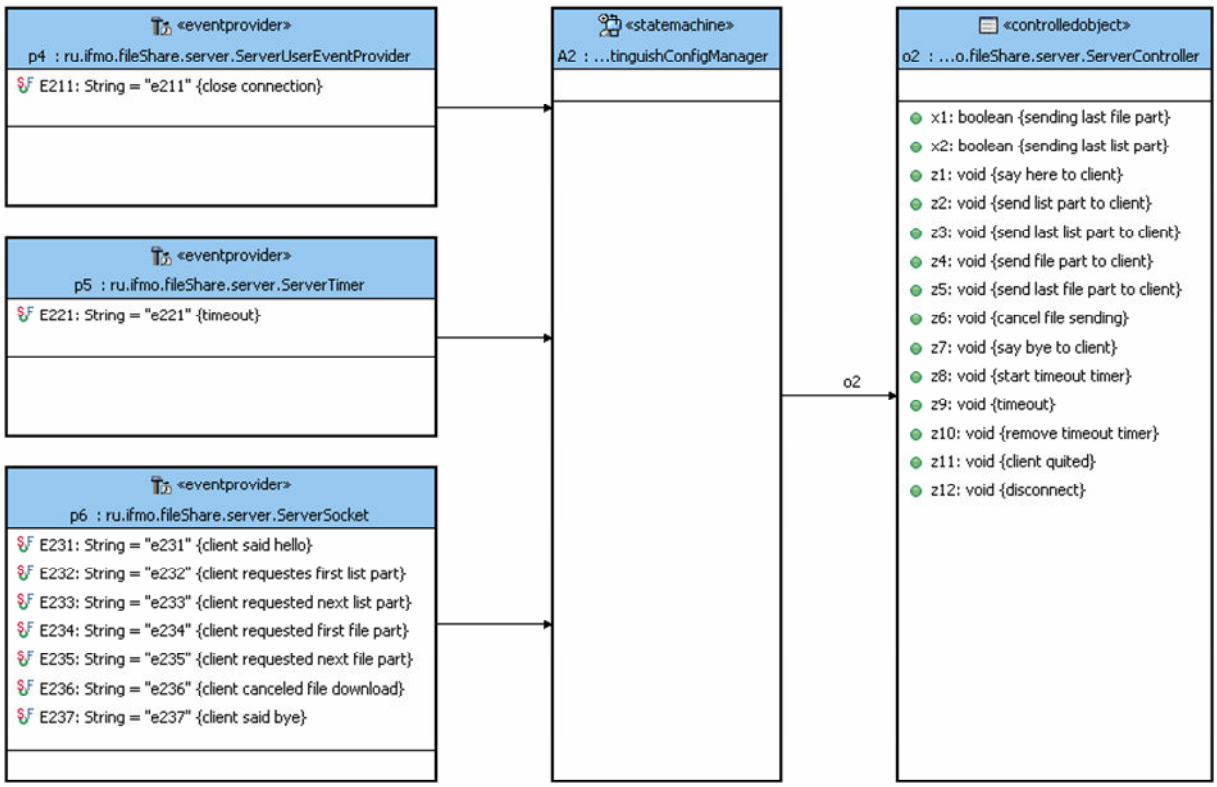


Рис. 3. Схема связей автомата *A2*

3.2.3. Описание состояний

Состояния автомата *A2* приведены в табл. 9.

Таблица 9

Handshaking	Ожидание приветствия клиента
Connected	Сеанс работы с клиентом
Idle	Ожидание запроса клиента
Sending List	Отправка списка файлов
Sending File	Отправка файла

3.2.4. Описание событий

События, относящиеся к поставщику событий окна приложения (*p4*), приведены в табл. 10.

Таблица 10

e211	Завершение соединения

События, относящиеся к поставщику событий таймера (*p5*), приведены в табл. 11.

Таблица 11

e221	Таймаут

События, относящиеся к поставщику событий соединения с клиентом (*p6*), приведены в табл. 12.

Таблица 12

e231	Получено приветствие клиента
e232	Получен запрос первой части списка файлов
e233	Получен запрос очередной части списка файлов
e234	Получен запрос первой части файла
e235	Получен запрос очередной части файла
e236	Клиент отменил загрузку файла
e237	Клиент закрыл соединение

3.2.5. Список выходных воздействий

Выходные воздействия серверного объекта управления (*o2*) приведены в табл. 13.

Таблица 13

z1	Отправка приветствия
z2	Отправка очередной части списка файлов
z3	Отправка последней части списка файлов
z4	Отправка очередной части файла
z5	Отправка последней части файла
z6	Отмена отправки файла
z7	Отправка сигнала закрытия соединения
z8	Запуск таймера
z9	Обработка сигнала таймера
z10	Сброс таймера
z11	Обработка закрытия соединения клиентом
z12	Закрытие соединения

3.2.6. Список параметров при переходах

Параметры серверного объекта управления (*o2*) приведены в табл. 14.

Таблица 14

x1	Посылается последняя часть файла
x2	Посылается последняя часть списка файлов

3.2.7. Граф переходов

Граф переходов автомата *A2* изображен на рис. 4.

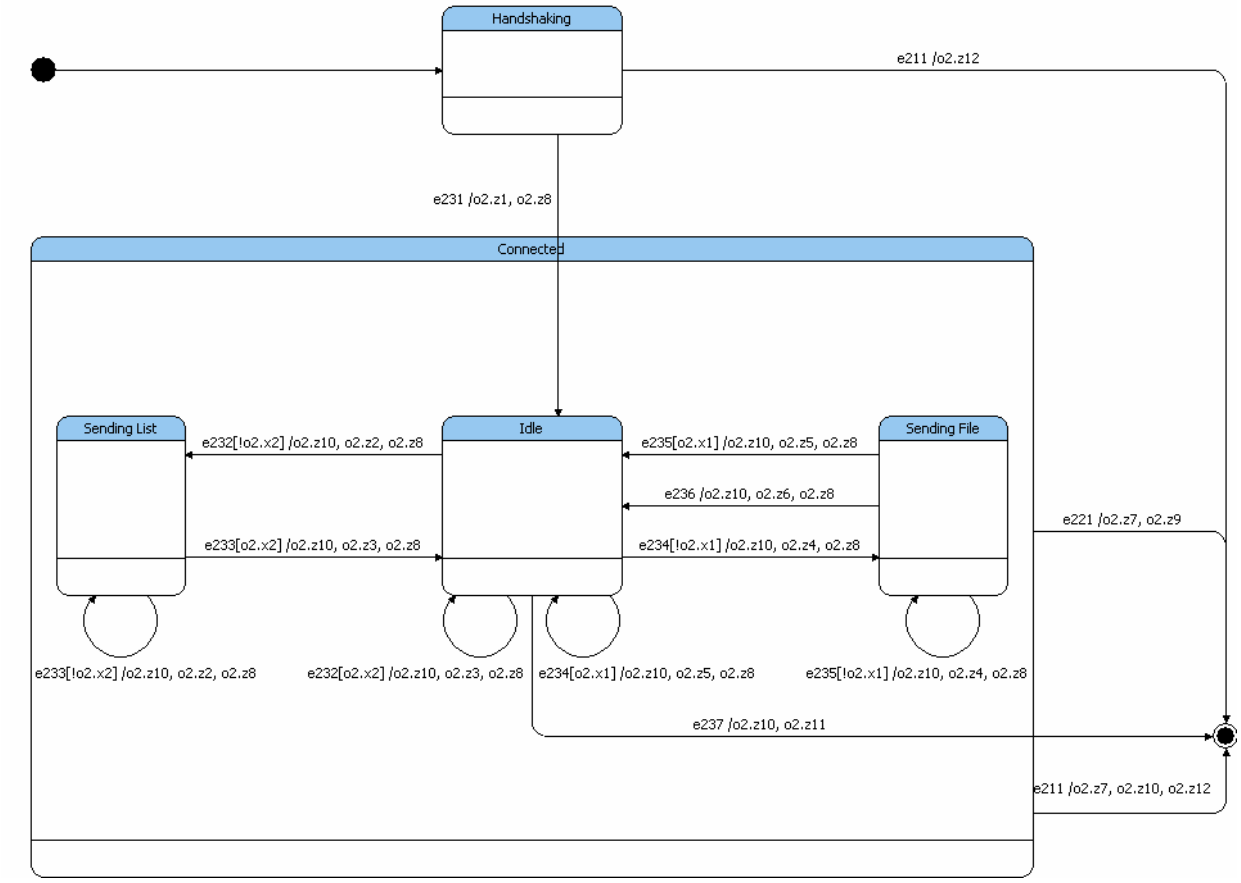


Рис. 4. Граф переходов автомата *A2*

4. Реализация

Одна из сильных сторон пакета *UniMod* – это возможность визуального конструирования программ. В отличие распространенного подхода, когда вспомогательные картинки и *UML*-диаграммы рисуются с надеждой на улучшение документации и продуктивности труда, разработанные с помощью инструментального средства *UniMod* диаграммы и вручную написанные классы в целом формируют работающее приложение.

4.1. Интерпретационный подход

При разработке программы в среде *Eclipse* может применяться интерпретационный подход, позволяющий эффективно выполнять тестирование и отладку. Для получения готового приложения в рамках этого подхода *UML*-диаграммы экспортируются в *XML*-описание.

При интерпретационном подходе (рис. 5) во время выполнения программы в памяти создаются экземпляры *Java*-классов источников событий и объектов управления.

В процессе обработки события интерпретатор по текущему состоянию получает набор переходов в новое состояние и вычисляет логические условия переходов. Если результатом вычисления какого-либо логического условия является истина, то происходит следующее:

- интерпретатор с использованием механизма *Reflection* языка *Java* вызывает выходные воздействия, ассоциированные с выбранным переходом;
- затем производится переход в новое состояние, и выполняются выходные воздействия, предусмотренные для исполнения в момент входа в новое состояние;
- управление передается вложенным автоматам.

Рис 5. Этапы разработки и выполнения приложения на основе интерпретационного подхода

В процессе работы программы ведется протокол, в который заносятся:

- поступающие события;
- состояние, в котором находится каждый автомат;
- значения входных переменных;
- переходы;
- выходные воздействия;

Интерпретационный подход весьма удобен в процессе разработки приложения, однако он обладает следующими недостатками:

- для запуска программы, кроме скомпилированных *Java*-классов, дополнительно необходимы *XML*-описание и интерпретатор *UniMod*.
- по производительности интерпретационный подход выполнения программы проигрывает компиляционному.

Все это привело к созданию еще и компиляционного подхода.

4.2. Компиляционный подход

Данный подход позволяет транслировать *UML*-диаграммы в код на языке *Java* (рис. 6). Трансляция *XML*-описания, полученного по диаграммам, в код на языке *Java* осуществляется с использованием довольно гибкого инструмента *Velocity* (http://jacarta.apache.org/velocity).

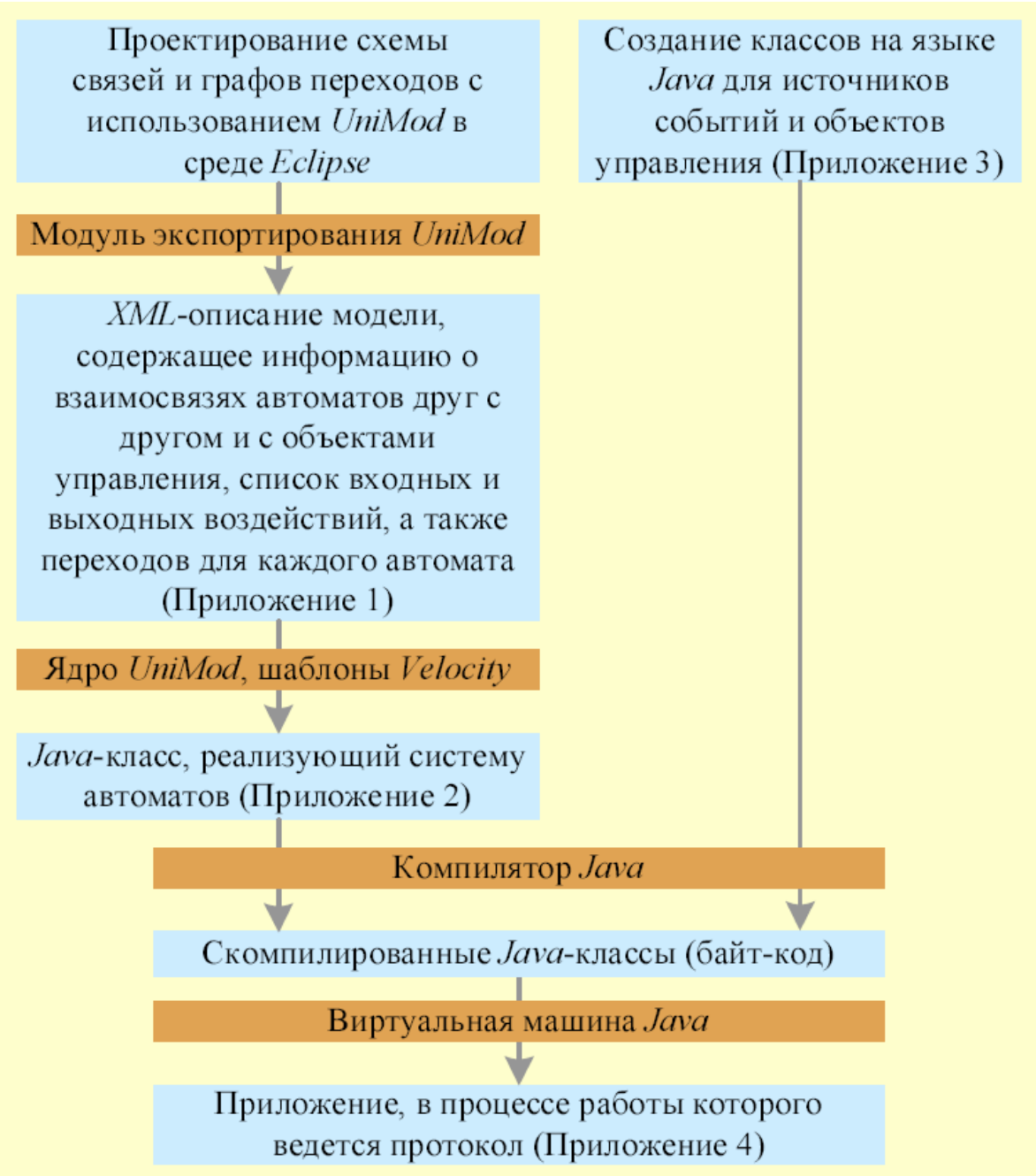


Рис. 6. Этапы разработки и выполнения приложения на основе компиляционного подхода

По *XML*-описанию автоматически генерируется *Java*-класс, который реализует систему автоматов. Затем этот *Java*-класс и классы, реализующие функциональность источников событий и объектов управления, компилируются. Полученное таким образом приложение уже не зависит ни от *XML*-описания, ни от интерпретатора *UniMod*. Для запуска программы требуется только библиотека времени исполнения *UniMod* в скомпилированном виде.

Отметим, что *UML*-диаграммам однозначно соответствует автоматически генерируемый код на языке *Java*. При этом приложение, разработанное в рамках интерпретационного подхода, будет функционально эквивалентно его аналогу, построенному с использованием трансляции диаграмм в код на языке *Java*.

На рис. 7 приведено сравнение объема кода *Java*-классов проекта, сгенерированных по *UML*-модели и написанных вручную.

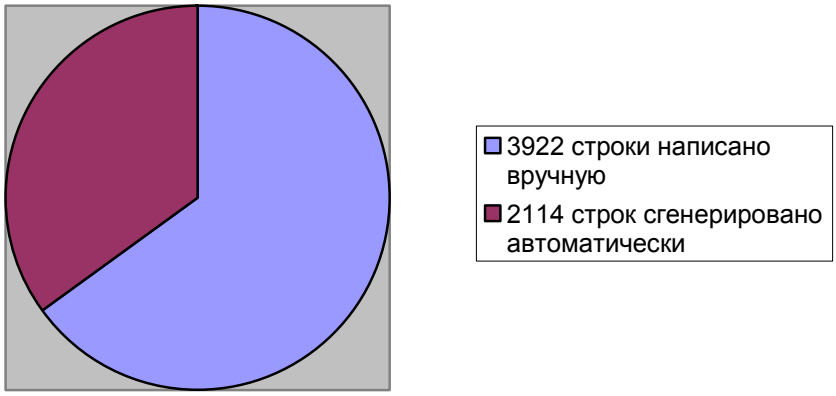


Рис. 7. Сравнение количества строк кода на языке *Java*, написанных вручную и сгенерированных по *UML*-диаграммам.

4.3. Запуск программы

Для запуска скомпилированной программы необходимо с сайта http://is.ifmo.ru (раздел «UniMod-проекты») скачать архив с исполняемым кодом, распаковать его и запустить с помощью файла `start.bat`. Для запуска необходимо, чтобы на компьютере была установлена *Java* версии 1.5 или более поздней.

Заключение

Данный проект показал удобство и целесообразность применения автоматного подхода при проектировании и программировании сетевых приложений. При этом основой для разработки таких приложений является описание протокола взаимодействия между клиентами и сервером. Используя этот протокол, строятся графы переходов рассматриваемых приложений, а также описываются потоки данных. Использование средств визуального построения схемы связей и автоматов позволяет значительно упростить процесс разработки автоматов приложения.

Литература

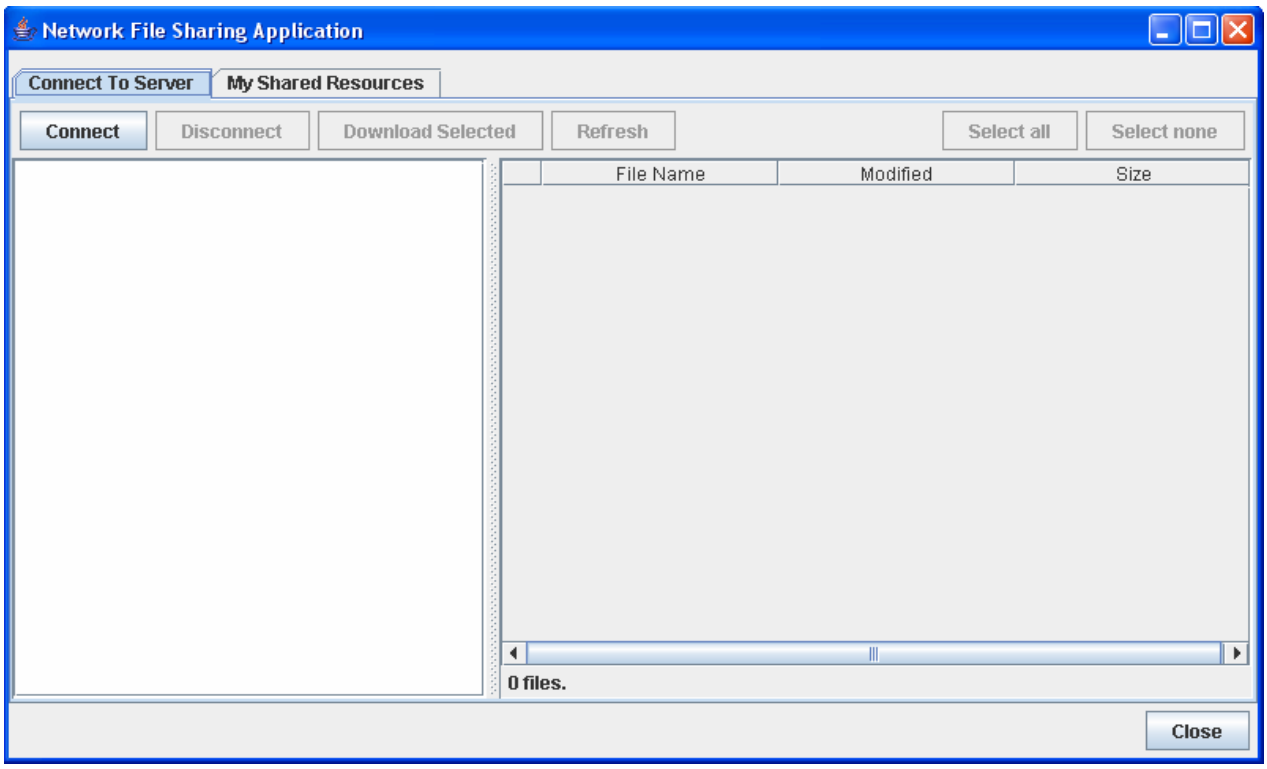
1. *Шалыто А.А.* SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука. 1998. http://is.ifmo.ru/books/switch/1

2. *Шалыто А.А.* Новая инициатива в программировании. Движение за открытую проектную документацию // Мир ПК. 2003. № 9, с.52–56. http://is.ifmo.ru/works/open_doc

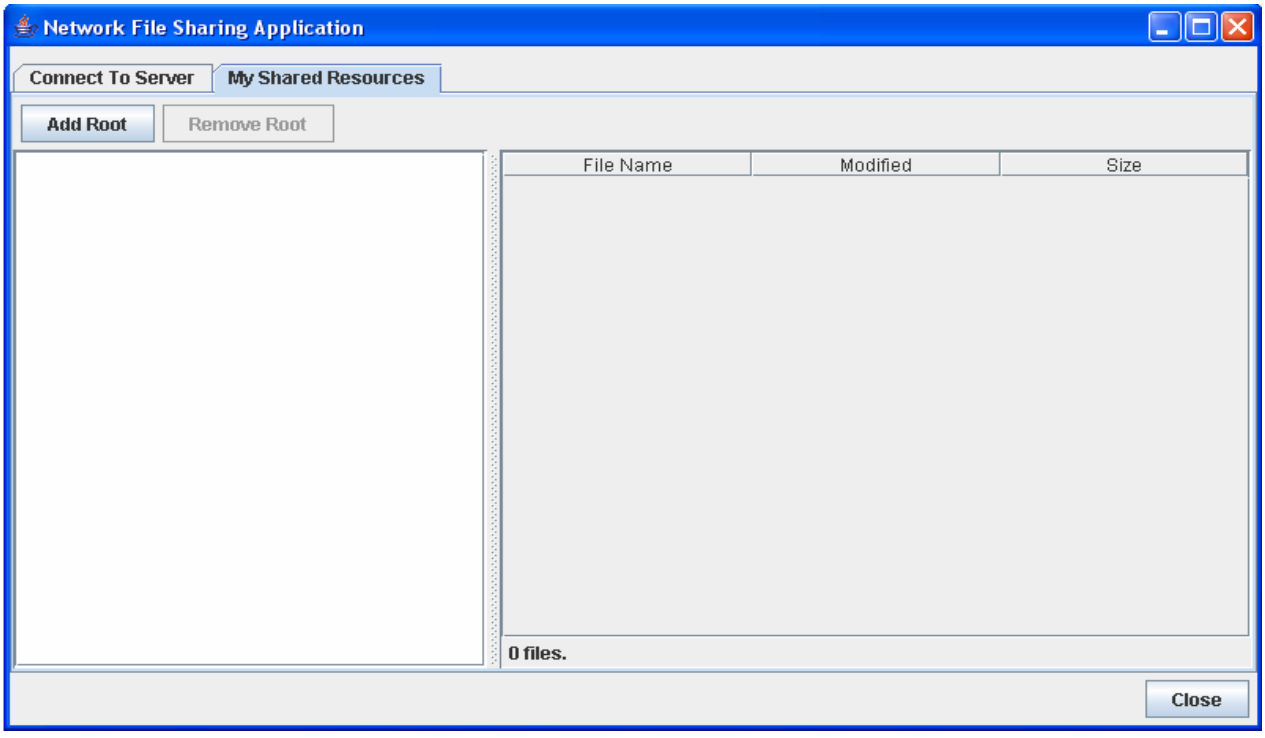
3. *Гуров В.С., Мазин М.А, Нарвский А.С, Шалыто А.А.* UML. SWITCH-технология. Eclipse // Информационно-управляющие системы. 2004. № 6, с.12–17. http://is.ifmo.ru/works/uml-switch-eclipse

Приложение. Демонстрация проекта

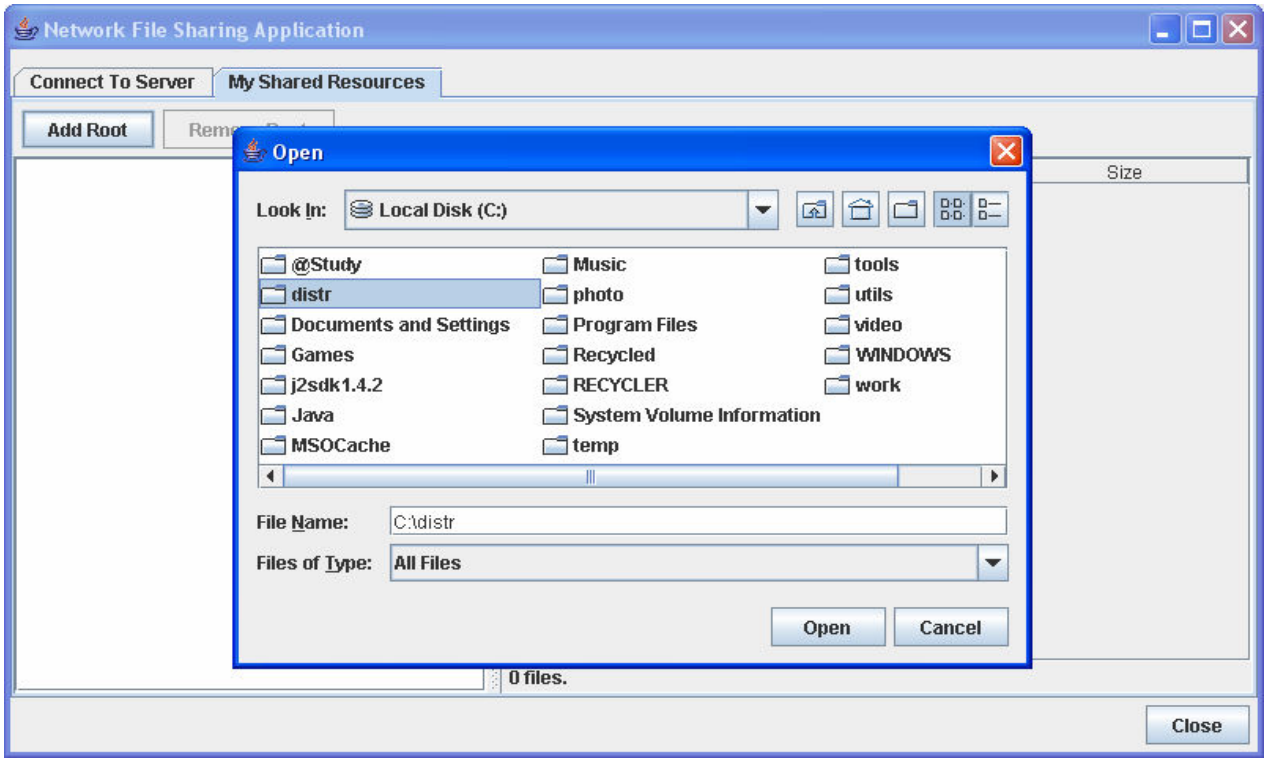
1. Запустите командный файл `start.bat`. Появится окно приложения:



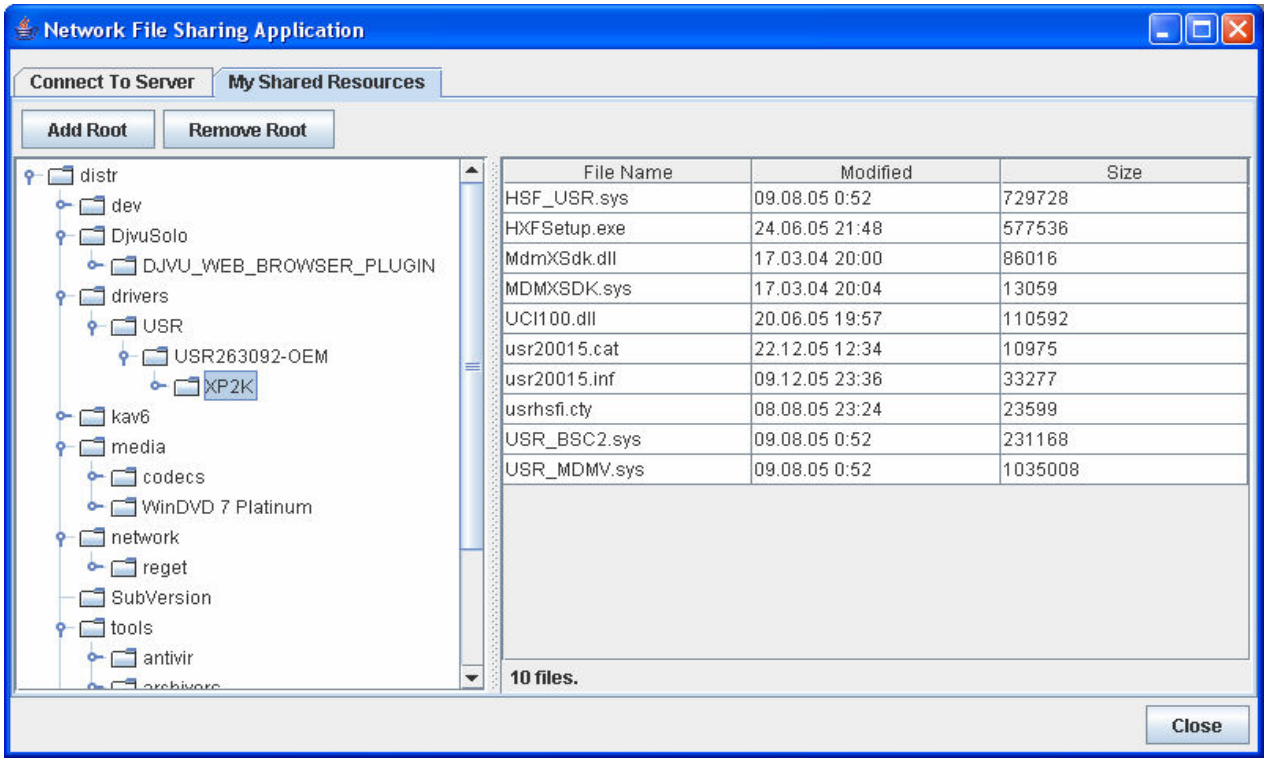
2. Перейдите на вкладку «My Shared Resources»:



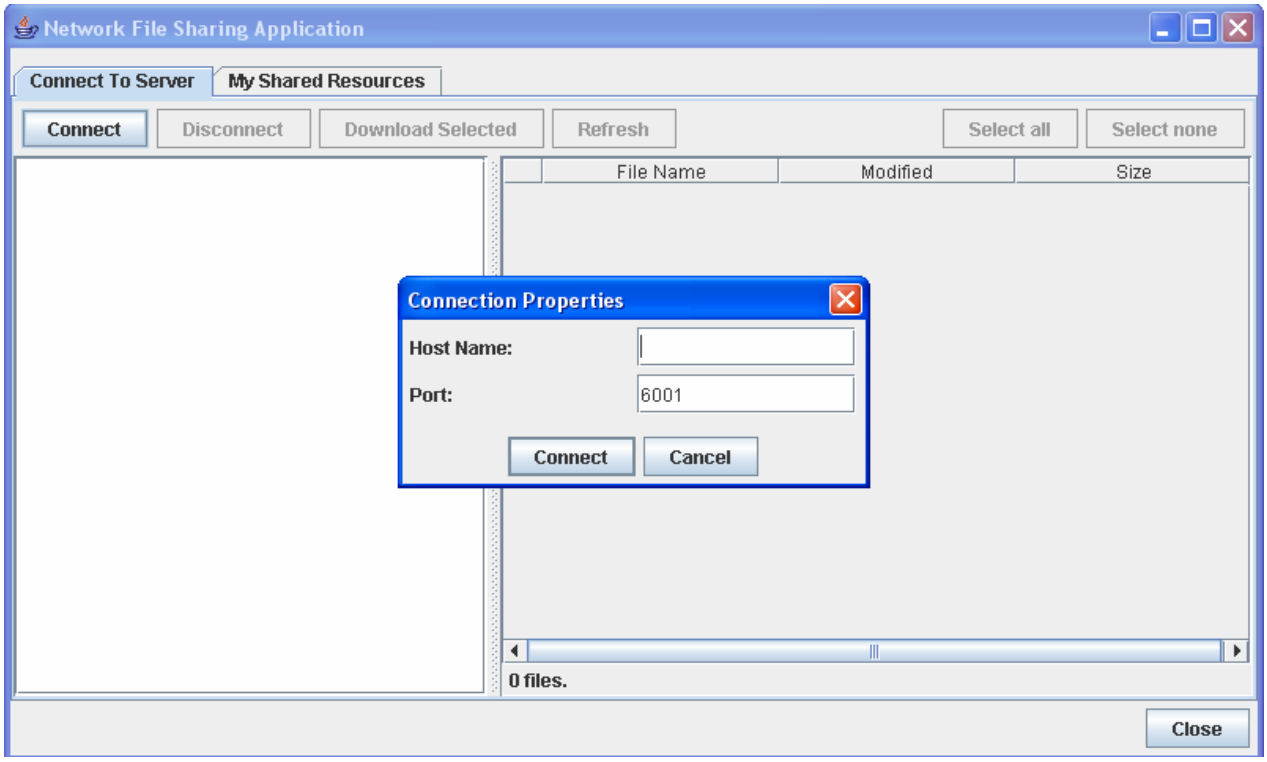
3. Нажмите «Add Root» и выберите папку, к содержимому которой вы хотите предоставить доступ:



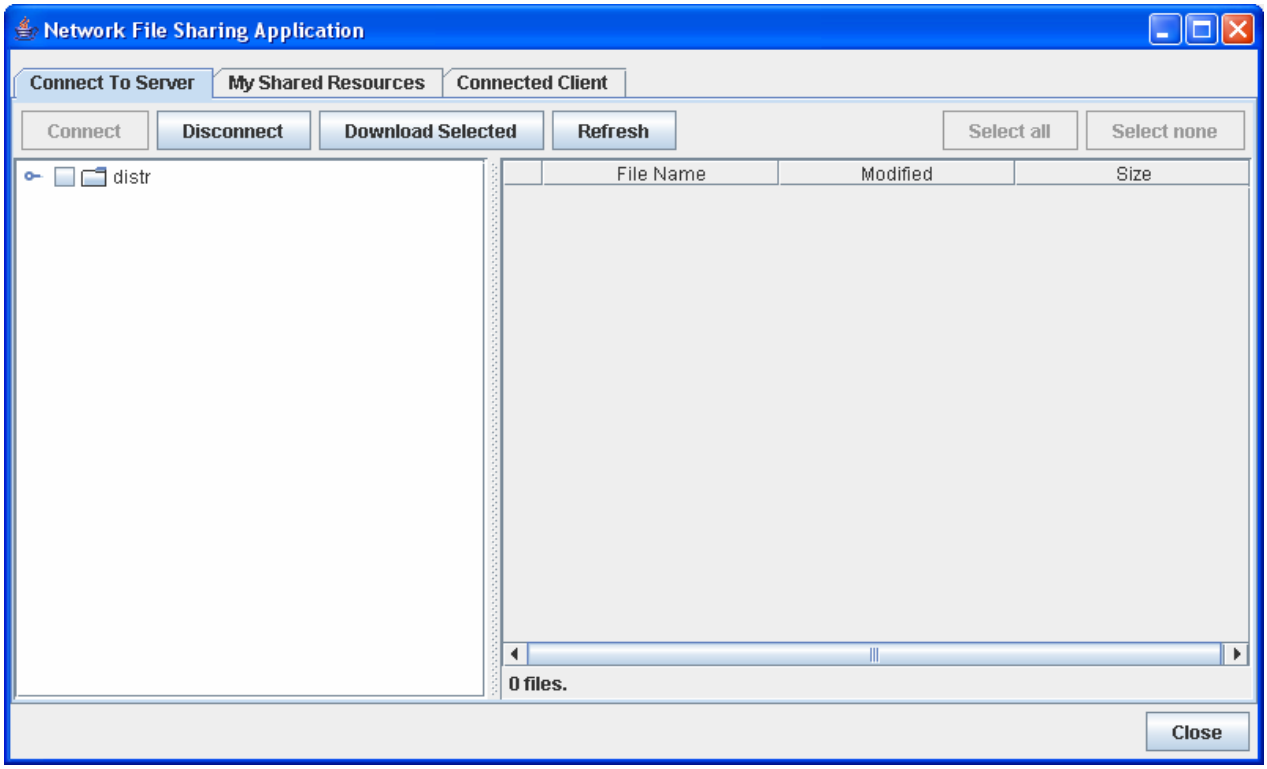
После нажатия кнопки «Open» на вкладке «My Shared Resources» отобразится содержимое выбранной папки:



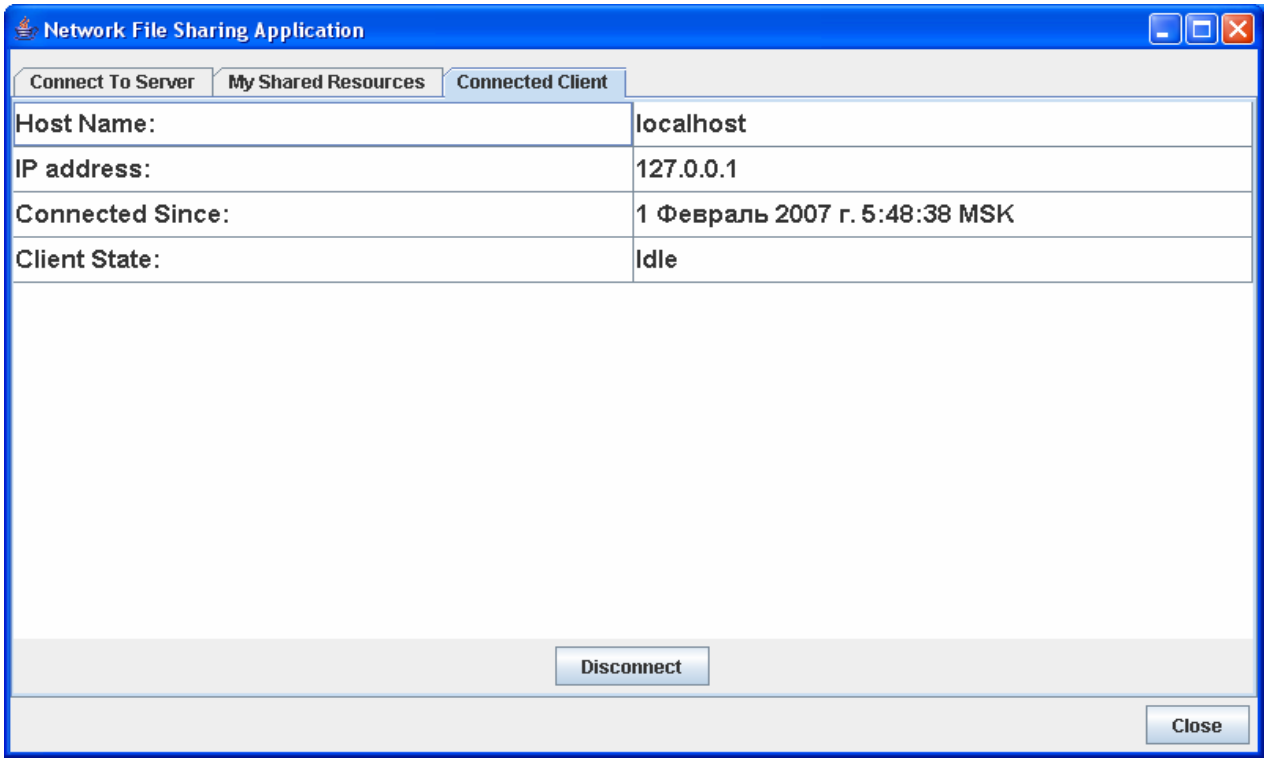
4. Перейдите на вкладку «Connect To Server» и нажмите кнопку «Connect», появится окно, запрашивающее имя сервера и номер порта для подключения:



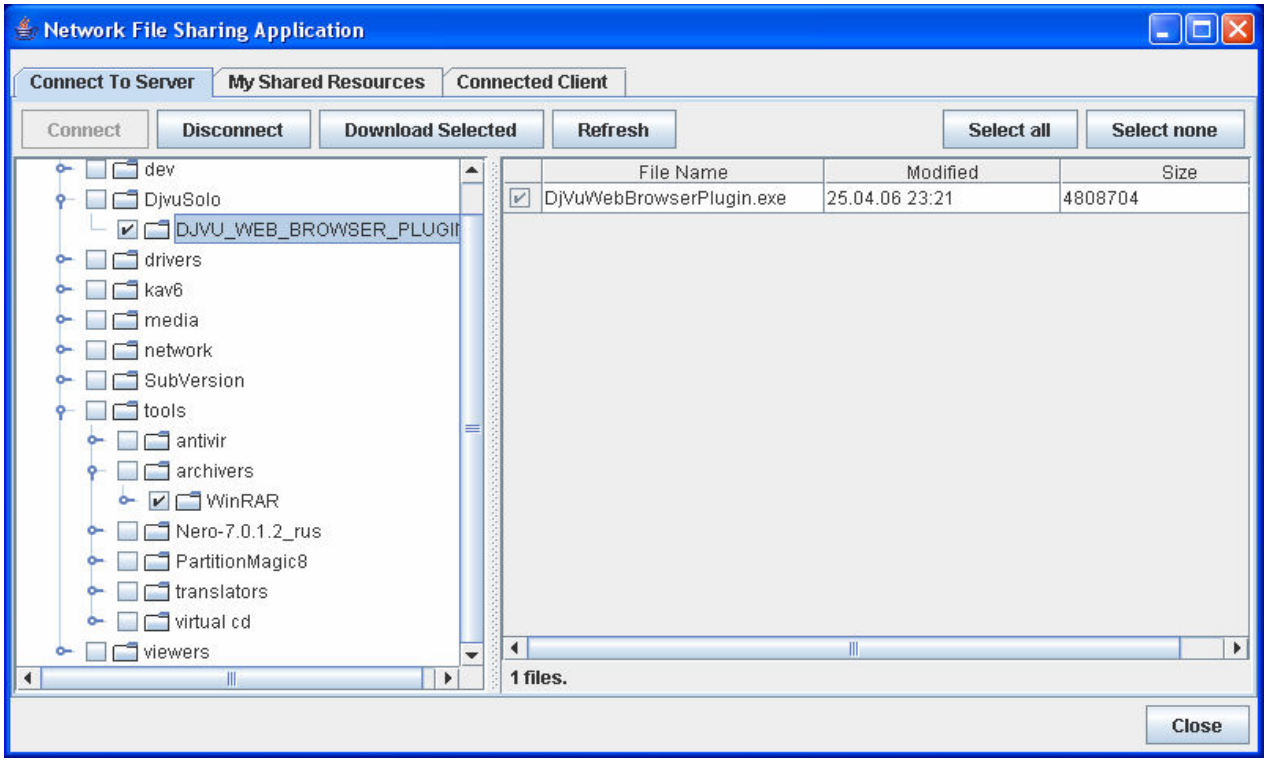
5. Введите «localhost» в поле «Host Name» и нажмите «Connect», на вкладке «Connect To Server» отобразится папка, которую вы выбрали на вкладке «My Shared Resources»:



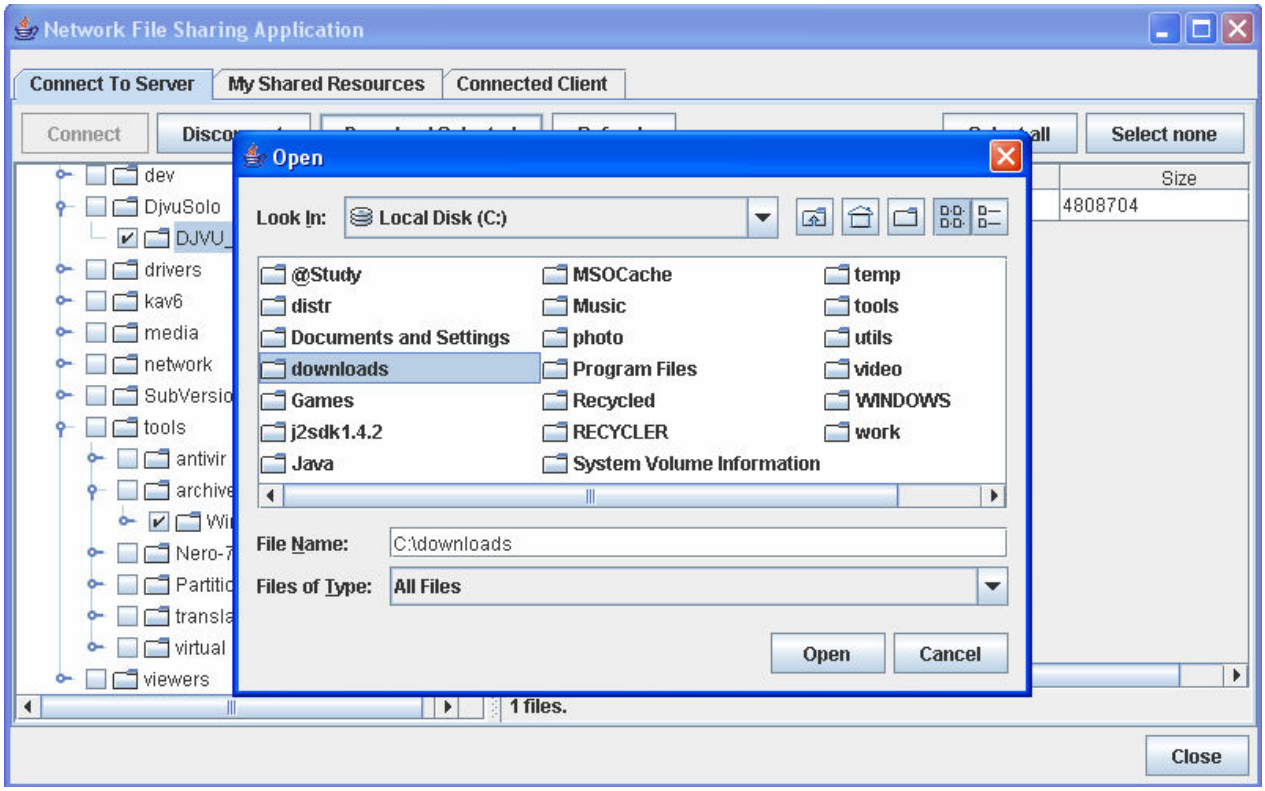
О том, что к серверу подключился клиент, сервер сигнализирует появлением вкладки «Connected Client» с информацией о подключившемся клиенте:



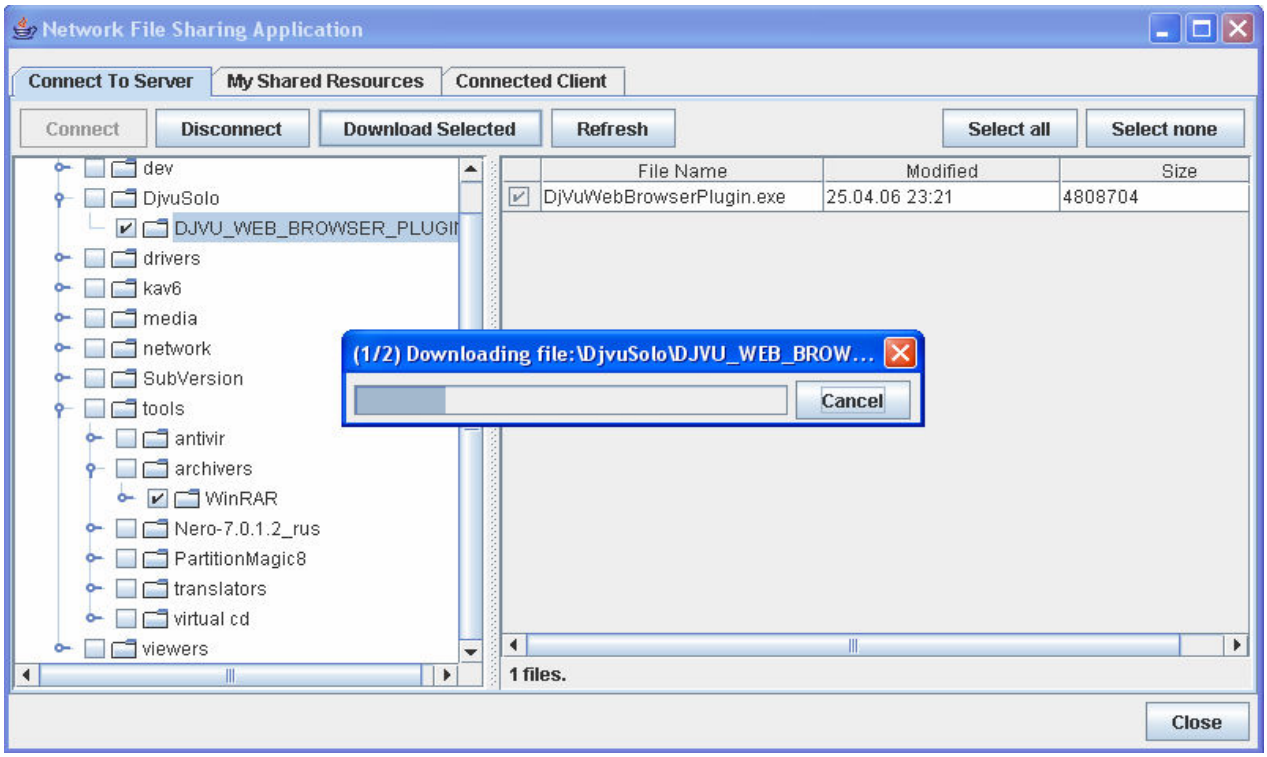
6. Выберите файлы и папки, которые вы хотите скопировать, поставив перед ними галочки:



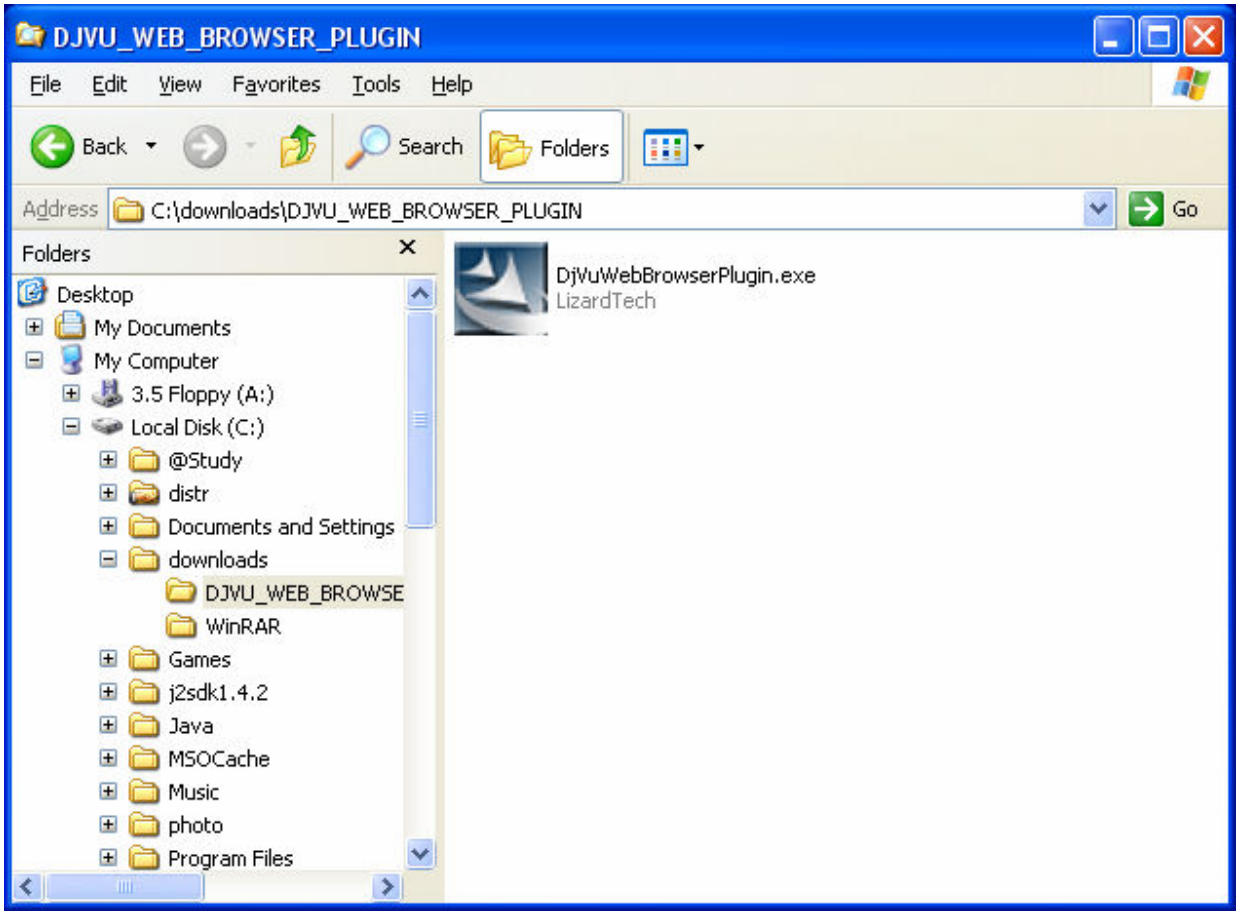
7. Нажмите «Download Selected» и выберите папку, в которую вы хотите сохранить скопированные файлы:



После нажатия кнопки «Open» начнется копирование выбранных файлов и папок:



8. Откройте папку, выбранную для сохранения файлов, чтобы убедиться, что файлы успешно скопированы:



9. Закройте подключение к серверу, нажав на кнопку «Disconnect» на вкладке «Connect To Server».
10. Завершите приложение, закрыв его окно (например, используя комбинацию клавиш *Alt+F4*).