УДК 004.38:004.6

ПРОБЛЕМА ПЕРЕДАЧИ ДАННЫХ МЕЖДУ ПЕРСОНАЛЬНЫМ КОМПЬЮТЕРОМ И ПЛАТФОРМОЙ БОЛЬШИХ СЕРВЕРОВ КЛАССА МЭЙНФРЕЙМ

Е.А. МАРГУНОВ
(*Белорусский государственный университет информатики и радиоэлектроники*);
канд. техн. наук, доц. В.И. МИСЮТКИН
(*Гомельский государственный технический университет им. П.О. Сухого*)

Рассмотрены проблемы обмена данными между приложениями, функционирующими на различных по мощности вычислительных устройствах: мэйнфреймах и персональных компьютерах, имеющих разные операционные системы. Проанализированы существующие способы передачи данных и предложено новое решение в виде визуального FTP-клиента с полной поддержкой организации данных в ОС операционных системах Windows и z/OS, обладающего высокой скоростью передачи данных и имеющего удобный, простой в обращении и интуитивно понятный для пользователя интерфейс.

Ключевые слова: *мэйнфреймы, передача данных, FTP-клиент, ОС Windows, ОС z/OS, персональный компьютер.*

Введение. Мэйнфрейм – это большой универсальный высокопроизводительный отказоустойчивый сервер, имеющий значительные ресурсы ввода/вывода, большой объем оперативной и внешней памяти и высокое быстродействие. Мэйнфреймы обладают целым рядом достоинств: высокой степенью надежности и устойчивости, высокой пропускной способностью и степенью загрузки, повышенной системой защиты, вследствие чего их используют в критически важных системах с интенсивной пакетной и оперативной транзакционной обработкой. Именно по этим причинам, по данным печати, около 70% всех важных бизнес-данных обрабатываются на мэйнфреймах.

Основной разработчик мэйнфреймов – корпорация IBM, чьи компьютеры работают под управлением операционной системы (ОС) z/OS. Данная ОС изначально разрабатывалась как многопользовательская система с возможностью удаленного подключения пользователей. Такими пользователями в настоящее время являются персональные компьютеры, работающие в своем большинстве под управлением ОС Windows в разных ее версиях.

Проблемы взаимодействия двух разных платформ возникают при обмене данными между мэйнфреймом и персональным компьютером. Их причины – существенные различия в организации хранения данных в z/OS и Windows и отсутствие удобного пользовательского интерфейса, посредством которого осуществляется такой обмен. Последнее у мейнфреймов нуждается в усовершенствовании.

Рассмотрим один из вариантов решения указанных проблем.

Особенности организации данных в ОС Windows и z/OS. Основной единицей системы организации данных в ОС Windows является файл. Файлы размещаются в директориях, образующих иерархическую структуру. Само содержимое файлов представляет собой поток байтов, не имеющих структуры. Поскольку внутренняя организация у файлов отсутствует, то все их свойства являются внешними, например: дата создания, дата изменения, атрибуты доступа и др. Их можно изменить без какого-либо влияния на данные файла. Для представления данных в Windows используются кодировки семейства ASCII или UNICODE [1].

Наоборот, в z/OS не существует прямого эквивалента понятию директории, а все наборы данных образуют одноуровневую структуру. Но принятая в z/OS схема именования наборов данных превращает одноуровневую структуру в иерархическую. Поэтому, можно считать, что эквивалентом директории в z/OS является квалификатор имени набора данных. Сама структура набора данных в z/OS, которая определяется при его создании, не может быть изменена. Наборы данных, помимо внешних, имеют еще и внутренние свойства, как например: форма организации набора данных, формат записи, длина записи, изменить которые невозможно [2]. В z/OS данные представляются в кодировке семейства EBCDIC, в том числе и в файлах HFS. Все кодировки семейства EBCDIC, также как кодировки ASCII, являются однобайтовыми.

Можно выделить две цели, которые преследуются при обмене данными из ОС Windows в z/OS и наоборот:

1. Последующая обработка данных принимающей стороной.
2. Последующее хранение данных на принимающей стороне.

В первом случае для принимающей стороны особый интерес представляют сами данные, а не их атрибуты. Наборы данных или файлы передают в такой форме, которая является наиболее приемлемой для принимающей стороны. Самой существенной проблемой для достижения этой цели является использование различных представлений данных. Внутри каждого семейства представление латинских символов, цифр и основных специальных символов совпадает, однако воспроизведение даже этого основного набора символов для разных семейств различно. Поэтому перекодировка данных является неизбежной. Она может выполняться как отдельный процесс, до передачи данных или после нее, но более удобно и просто осуществлять перекодировку в процессе передачи данных, конечно, если это возможно.

Во втором случае интерес вызывают как сами данные, которые в этом случае нежелательно подвергать какой-либо перекодировке, так и их атрибуты, т.е. свойства соответствующих наборов данных или файлов. По причине того, что даже между подобными объектами Windows и z/OS есть лишь приблизительное соответствие, потеря некоторых атрибутов при передаче данных является неизбежной. В некоторых случаях это не является критичным, так как утерянные атрибуты могут быть достаточно простым образом восстановлены. На практике передаваемый объект преобразуют до его передачи в транспортную форму, которая должна обеспечить сохранение и данных, и их атрибутов. Вполне годятся для этого программы архивирования, которые, кроме того, сжимают данные, тем самым существенно уменьшая их объем при передаче, а также обеспечивают проверку целостности переданных данных [3]. Для Windows это распространенные программы архивации: Zip, RAR, 7z и другие, а для z/OS – программа ADRDSSU, которая может обрабатывать наборы данных всех видов организации, включая VSAM и HFS.

Проектирование и реализация визуального FTP-клиента для передачи данных между ОС Windows и z/OS. Для решения указанных проблем был спроектирован и реализован визуальный FTP-клиент с полной поддержкой организации данных в операционных системах Windows и z/OS. Чтобы обеспечить гибкость в сопровождении и возможность дальнейшего расширения функциональных возможностей, он имеет модульную структуру.

С этой целью был использован способ реализации системы, основанный на «прозрачном» функциональном ядре, который позволил осуществить все операции по прорисовке графического интерфейса, организации многопоточной работы и осуществления операции по взаимодействию модулей. Компонентная схема функционального ядра системы представлена на рисунке 1.

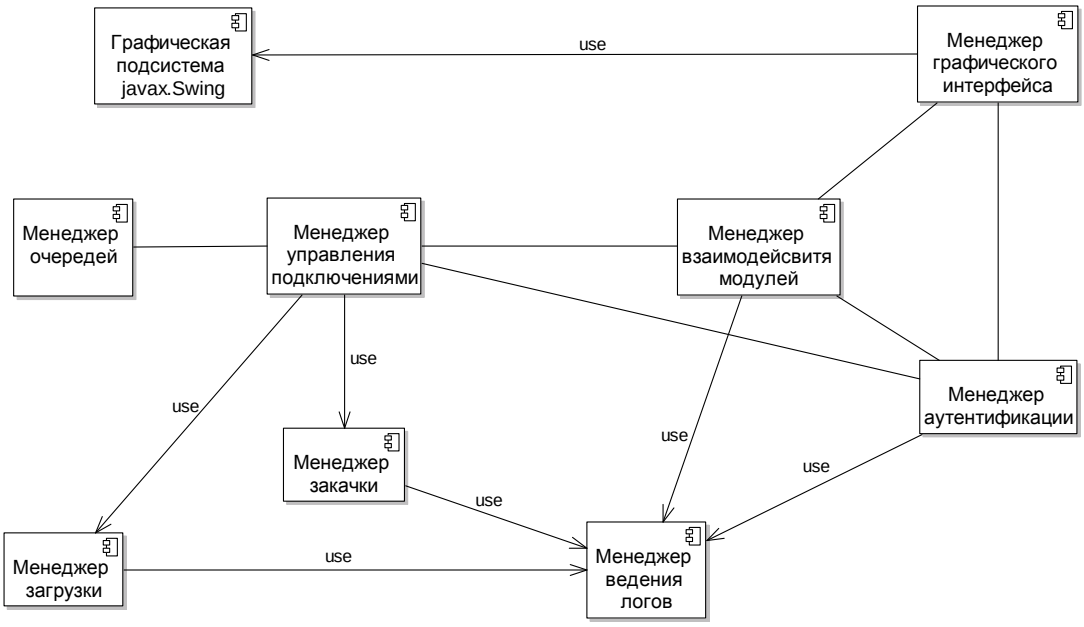

Рисунок 1 – Компонентная схема функционального ядра системы

Рассмотрим более детально некоторые из компонентов функционального ядра.

Менеджер графического интерфейса для прорисовки интерфейса напрямую использует графическую подсистему javax.Swing. Ее компоненты поддерживают специфические, динамически подключае-

мые виды и поведения, благодаря чему становится возможной адаптация системы к интерфейсу платформы. Приложения, использующие Swing, могут выглядеть как «родные» для любой операционной системы, что делает их интерфейс универсальным для различных платформ.

На основании данных о зарегистрированных в системе модулях (используя *менеджер взаимодействия модулей*) и в соответствии с правами доступа (используя *менеджер аутентификации)* менеджер графического интерфейса изменяет состояние компонентов меню и форм.

Менеджер аутентификации и авторизации отвечает за хранение учетных данных и доступ пользователей к серверу мэйнфрейма, а также за использование функциональных возможностей того или иного зарегистрированного модуля.

Менеджер ведения логов отвечает за отображение всех существенных изменений в системе в виде сообщений, которые появляются в одном из окон менеджера графического интерфейса и в системном файле.

Менеджер управления подключениями отвечает за установку соединения с сервером. Для управления многопоточностью он взаимодействует с *менеджером очередей*. Кроме того, в своей работе он использует *менеджеры загрузки и закачки* для передачи и удаления файлов, наборов данных или директорий между сервером и клиентом.

Этап проектирования был реализован с применением рационального процесса унификации. В качестве общей нотации использован унифицированный язык моделирования UML, инструментом для работы с которым стал продукт Enterprise Architect компании Sparx Systems.

Функциональные возможности системы отражены на рисунке 2.

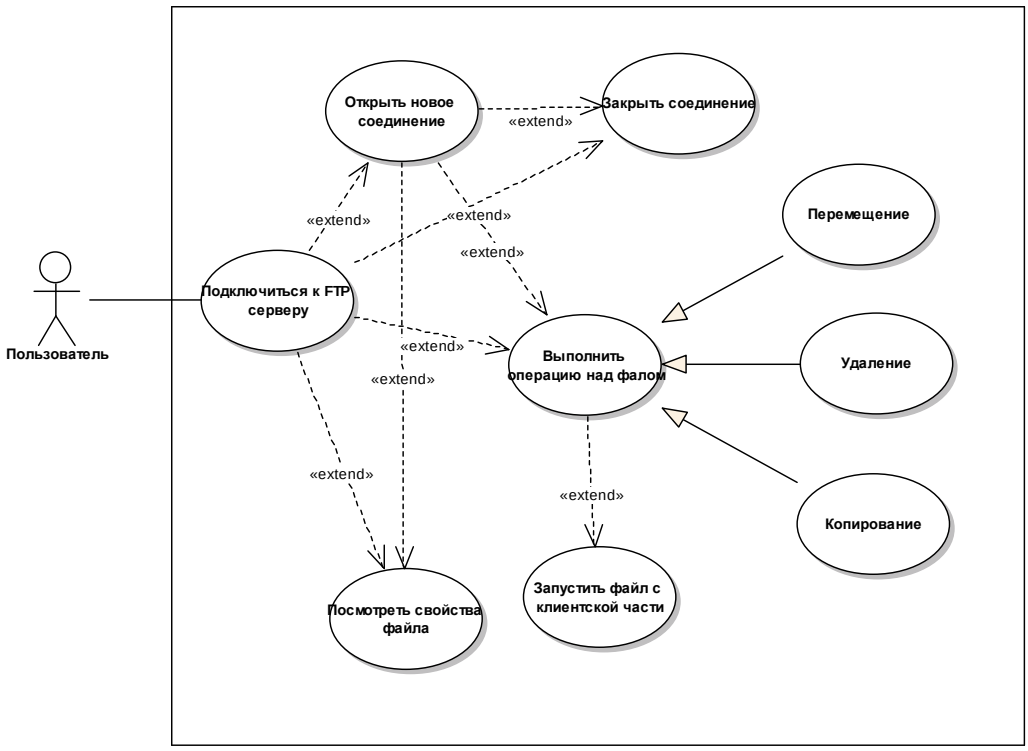

Рисунок 2 – Диаграмма вариантов использования

Отметим функциональные возможности проектируемого программного приложения:

– копирование, удаление, перемещение объектов из локальных папок в каталоги, расположенные в системе z/OS и наоборот;
– поддержка нескольких соединений;
– корректное восстановление передачи данных в случае разрыва соединения;
– возможность редактирования документа прямо на FTP-узле;
– возможность просмотра свойств интересующего объекта;
– отображение атрибутов файлов и каталогов;
– сохранение предыдущих сессий;

- возможность управления загрузками;
- управление очередями;
- возможность хранения учетных данных для соединений в зашифрованном виде;
- сортировка файлов и каталогов по различным параметрам;
- просмотр содержимого файлов;
- сохранение протокола передачи в log-файлах.

Проект реализован в виде набора различных алгоритмов. Рассмотрим основные из них.

1. **Алгоритм для хранения учетных данных пользователя, обеспечивающих вход в систему мэйнфрейма.**

Входные данные: конфигурационный файл, имя пользователя, пароль.

Работа алгоритма. Поскольку доступ к мэйнфрейму строго регламентируется, то категорически запрещается держать пароль в открытом виде. Поэтому учетные данные пользователя, необходимые для установки соединения, сохраняются в конфигурационном файле. Пользователь может выбрать вариант подключения из числа ему предложенных. По умолчанию пароль для доступа к серверу не сохраняется, но пользователь может установить настройки, позволяющие сохранять пароли. Если выбрана опция, позволяющая пользователю сохранять пароль, то пароль зашифровывается.

2. **Алгоритм многопоточной передачи данных.**

Входные данные: адрес сервера, порт, путь к файлу данных на персональном компьютере, путь набору данных на мэйнфрейме, имя файла/набора данных, имя пользователя, пароль.

Работа алгоритма. Выясняем, включена ли поддержка многопоточной передачи данных. Проверяем потоки передачи данных, если передача уже инициирована, то ставим поток в очередь и устанавливаем ему новый статус и наименьший приоритет.

Возможные статусы потока передачи данных: остановлен, в очереди, удален, загрузка, выгрузка.

Периодически контролируем возможность начать передачу данных. Устанавливаем соединение с сервером по алгоритму 3.

3. **Алгоритм установки соединения с сервером.**

Входные данные: адрес сервера, порт, имя пользователя, пароль.

Работа алгоритма. Создаем подключение на базе сокетов. Определяем тип файловой системы на сервере. Загружаем в окно, соответствующее удаленной машине, список файлов. Если тип системы MVS или Windows, проверяем типы наборов данных, файлов и проставляем соответствующие флаги, свойства, размеры, разрешения на доступ.

4. **Алгоритм закачки файла или набора данных.**

Входные данные: соединение с сервером, порт, адрес сервера, имя файла, тип файла, флаг продолжения записи файла, место окончания предыдущей записи, входной поток.

Работа алгоритма. Определяем, что мы будем передавать: файл или набор данных. Получаем имя и путь для копирования на клиентскую часть. Если такой файл уже существует, то возможна ситуация с возобновлением передачи данных. Используя подключение, созданное по алгоритму 3, пытаемся создать новый поток для передачи данных. Если после пяти попыток соединение для передачи данных не установлено, прекращаем попытки. Передаем серверу команду на передачу файла. Если установлен флаг, указывающий, что файл требуется дописать, используем стандартный класс произвольного доступа к файлу. Определяем кодировки принимающей и передающей системы. Если требуется, выполняем перекодировку. Если передача файла не началась, удаляем создавшийся пустой файл. Вызываем обновление окна принимающей машины.

Для программной реализации проекта применяем кросс-платформенный объектно-ориентированный язык программирования Java SE, язык управления заданиями JCL, а в качестве основной среды разработки используем интеллектуальную интегрированную среду разработки Java – IntelliJ IDEA.

Анализ работы спроектированного программного продукта. Результатом работы стал визуальный FTP-клиент, позволяющий передавать файлы и наборы данных между платформами Windows и z/OS, правильно интерпретировать структуру наборов данных в системе z/OS, корректно отображать структуру данных мэйнфрейма, переводить данные в «родные» кодировки принимающих систем и поддерживать многопоточность. Пример работы интерфейса при передаче данных показан на рисунке 3. Интерфейс использует английский язык, поскольку система разработана для нужд иностранного предприятия.

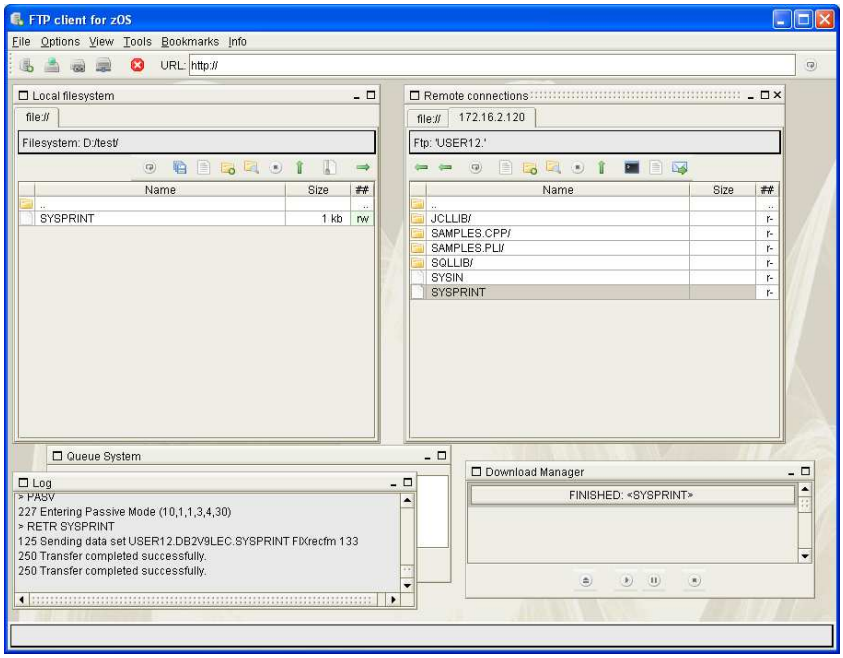

Рисунок 3 – Окно с результатами передачи набора данных

Тестируя разработанную систему, мы сравнили скорости передачи данных между операционной системой семейства Windows (персональный компьютер) и операционной системой z/OS (мэйнфрейм) при передаче посредством разработанного программного средства и через протокол TN3270 (встроенную функцию клиента Personal Communication) [6]. На рисунке 4 приведены графики скоростей передачи данных с использованием разработанной программы и TN3270:

– FTP binary и FTP text – передача данных посредством FTP-протокола в бинарном режиме и текстовом режиме соответственно;

– TN3270 binary и TN3270 text – передача данных с использованием Personal Communication через протокол TN3270 в бинарном режиме и текстовом режиме соответственно.

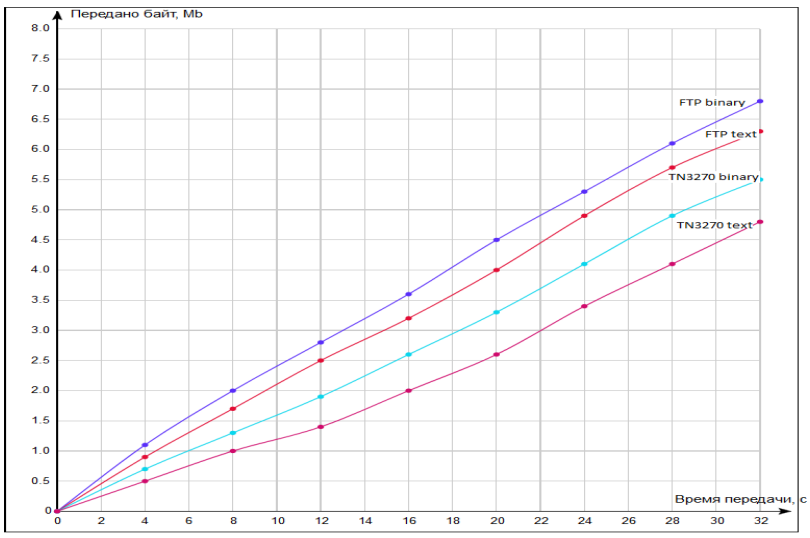

Рисунок 4 – Графики скорости передачи данных с использованием протоколов FTP и TN3270

Из графиков видно, что скорость передачи по протоколу FTP существенно выше, чем скорость передачи данных с помощью протокола TN3270.

Полученные результаты являются вполне объяснимыми: передача данных является основной функцией протокола FTP, в то время как основная функция протокола TN3270 – это эмуляция термина-

ла, а вовсе не передача данных. Тот факт, что скорость передачи данных бинарного режима выше скорости передачи соответствующего ему текстового режима в рамках одного протокола, объясняется тем, что для бинарного режима данные передаются без затрат времени на перекодировку.

Заключение:

1. Исследование показало, что для передачи больших объемов данных и увеличения производительности вычислительной системы предпочтительнее использовать тип передачи данных, основанный на протоколе FTP.
2. Основываясь на проведенных исследованиях, была спроектирована и реализована система передачи данных между персональным компьютером и мэйнфреймом. Она обладает высокой скоростью передачи данных, имеет удобный, простой в обращении и интуитивно понятный для пользователя интерфейс.
3. Разработанное программное обеспечение используется в аутсорсинговом проекте IBM Contract Handling International Solutions, что позволило значительно сократить время на передачу генерируемых в операционной системе z/OS документов на персональные компьютеры для их последующего анализа и обработки.

В дальнейшем, перспектива развития разработанного программного продукта может быть связана с расширением его функциональных возможностей, оптимизацией алгоритмов передачи данных, а также с разработкой новых модулей и интеграцией их в систему.

ЛИТЕРАТУРА

1. Меженный, О.А. Microsoft Windows XP. Самоучитель / О.А. Меженный. – М. : Диалектика, 2005. – 304 с.
2. Эбберс, М. Введение в современные мэйнфреймы: основы z/OS : учеб. пособие / М. Эбберс, У. О'Брайен, Б. Огден. – М. : Redbooks, 2007. – 642 с.
3. Интернет-библиотека z/OS [Электронный ресурс]. – Режим доступа: http://www-03.ibm.com/systems/z/os/zos/index.html.
4. z/OS Communication Server. IP User's Guide and Commands : Manual [Электронный ресурс] / IBM Corporation. – 3rd Ed. – Режим доступа: f1a1b920.boo.
5. Маргунов, Е.А. Проблемы передачи данных между персональным компьютером и платформой больших серверов класса мэйнфрейм / Е.А. Маргунов, В.И. Мисюткин // Новые математические методы и компьютерные технологии в проектировании, производстве и научных исследованиях : тез. докл. XV Республиканской науч. конф. студентов и аспирантов, Гомель, 26–28 марта 2012 г. / Гомел. гос. ун-т им. Ф. Скорины. – Гомель, 2012. – С. 87–88.
6. Маргунов, Е.А. Сравнение скорости передачи данных между персональным компьютером и мэйнфреймом через протоколы FTP и TN3270 / Е.А. Маргунов, В.И. Мисюткин // Тез. докл. XIII Межвузовской студенческой науч. конф., Минск, 26 апреля 2012 г. / Белорус. гос. экон. ун-т. – Минск, 2012. – С. 133–134.

Поступила 20.02.2016

PROBLEMS WITH DATA TRANSMISSION BETWEEN A PERSONAL COMPUTER AND A LARGE MAINFRAME-CLASS SERVER PLATFORM

E. MARHUNOU, V. MISIUTKIN

In the article we have considered the problems of communication between applications running on devices with different operating systems and with different computational capability: mainframes and personal computers. All the existing data transfer methods have been analyzed. New solution in the form of visual FTP-client which fully supports organizing data in Windows and z / OS has been proposed. The client has high data transfer rate and handy, easy to use and intuitive user interface.

Keywords: *mainframes, data transfer, FTP-client, OS Windows, OS z/OS, personal computers.*