УДК 004.772

А. Г. Глумов, А. В. Чернышова
Донецкий национальный технический университет, г. Донецк
кафедра прикладной математики и информатики

ПРОЕКТИРОВАНИЕ СИСТЕМЫ РЕПЛИКАЦИИ ПОЛЬЗОВАТЕЛЬСКИХ ФАЙЛОВ

Аннотация

Глумов А. Г., Чернышова А. В. Проектирование системы репликации пользовательских файлов. *Выполнено проектирование системы для репликации пользовательских файлов. Система репликации была реализована на языке программирования C# с использованием библиотеки .Net Framework. Актуальность проекта высока на современном этапе развития вычислительной техники, так как многие пользователи сталкиваются с проблемами, которые возникают при хранении файлов одновременно на нескольких компьютерах.*

Ключевые слова: *репликация файлов, локальная сеть, проектирование.*

Постановка проблемы. В современном мире большинство людей, использующих цифровые данные, сталкиваются с проблемой их репликации и своевременного обновления на различных устройствах. Кроме того, подобная проблема возникает в организациях, в случаях, когда множество людей должны иметь доступ к некоторым данным, и должны иметь возможность их изменять. Для решения данной проблемы была спроектирована и разработана система репликации пользовательских файлов, которая может быть использована как в локальной сети, так и в рамках сети интернет.

Цель статьи – описать идеи создания системы репликации пользовательских файлов.

Постановка задачи исследования. Необходимо спроектировать систему, которая бы обеспечивала репликацию пользовательских данных. Копии этих данных расположены на различных узлах, которые соединены в сеть древовидной формы.

Решение задач и результаты исследований. В ходе разработки программного продукта по репликации пользовательских данных возникли две основные проблемы:

1) проблема авторизации клиентов,
2) проблема обработки событий, поступающих в программу хаотическим образом.

Проблема авторизации состоит в том, что стандартный алгоритм авторизации посредством связки логин-пароль создает накладные расходы. Во

первых, идея авторизации логином и паролем предусматривает одностороннюю авторизацию (то есть клиент авторизуется на сервере, но не наоборот). Для нужд программы репликации необходима двусторонняя авторизация (оба узла должны быть равноправны с точки зрения возможности установки соединения). Во вторых, так как к одному узлу может быть подключено несколько других узлов, приходится хранить учетные записи для каждого из узлов. Третья проблема характерна любым парольным системам – это потенциальная слабость пароля, придуманного пользователем.

В системе репликации был использован механизм авторизации, который решает все три проблемы.

Авторизация разбивается на два этапа: знакомство узлов и собственно авторизацию. Знакомство производится только на этапе создания сети и не доступно после завершения. Для создания сети на центральном узле создается учетная запись. Все узлы, которые должны быть подключены к центральному, получают пару логин-пароль этой учетной записи и устанавливают соединение. После того, как узлы связаны, учетная запись деактивируется.

Во время установки соединения, узлы-клиенты авторизуются на сервере с помощью предоставленной им пары логин-пароль, и вместе с сервером на основе случайных чисел создают ключ для дальнейшей авторизации. Ключ создается случайным образом, за основу берется пароль учетной записи. Этот ключ в силу своей случайности является более стойким, чем пароль, придуманный пользователем. Так как он не передается по сети, а генерируется независимо узлами, злоумышленник не сможет его получить.

Так как эта учетная запись активна только во время создания сети, связывающей синхронизируемые компьютеры, злоумышленник будет сильно ограничен во временных ресурсах, так как пароль очень скоро перестанет быть актуальным.

Общая схема этапа «знакомство» представлена ниже:

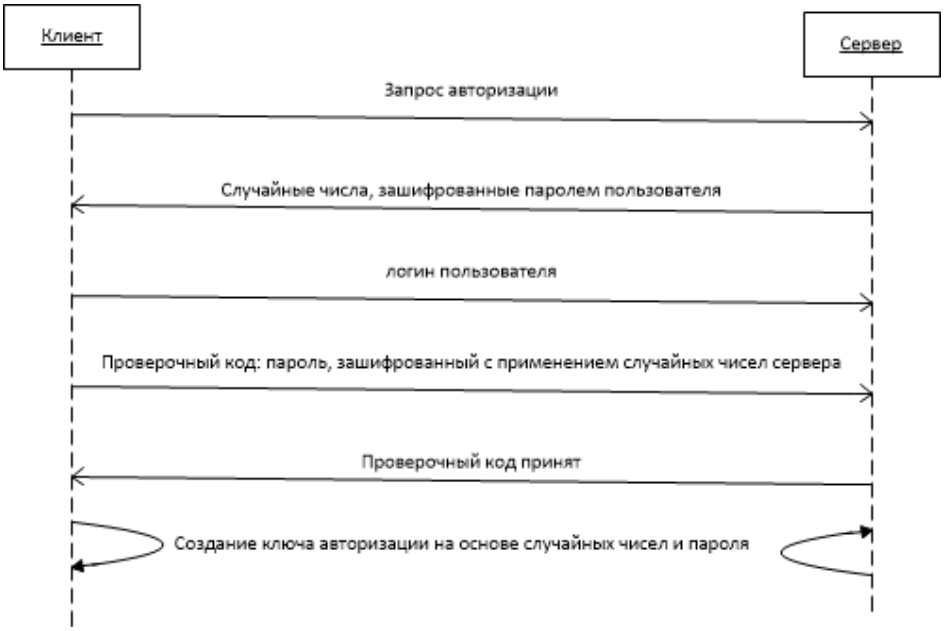

Рисунок 1 – схема этапа «знакомства» узлов сети.

Этап авторизации похож на этап знакомства, за исключением того, что клиент доказывает серверу наличие у него кода авторизации. После осуществления авторизации сервер и клиент создают ключ, которым шифруются передаваемые данные.

Вторая проблема, которая возникает при проектировании системы репликации пользовательских данных – это хаотичность поступаемых данных об изменении файлов в контролируемом каталоге.

Операционная система Windows предоставляет события, которые возникают в момент изменения какого-либо файла или каталога[2]. С помощью этих событий программа будет информироваться о произошедших изменениях и может запустить процесс обновления. Это гораздо экономичнее с точки зрения расхода ресурсов компьютера, чем по таймеру проверять на изменения все файлы в синхронизируемом каталоге.

Так как изменения могут приходить непредсказуемо, а обрабатывать их нужно строго последовательно, в программе используется специальный конвейер, построенный на основе блокирующей очереди, с использованием паттерна проектирования «Команда».

Суть этого механизма состоит в том, что при возникновении изменения, создается объект, который инкапсулирует в себе сведения о произошедших изменениях. Этот объект передается конвейеру, который в данный момент может обрабатывать другое событие. Объект помещается в очередь. Когда

конвейер освобождается, он извлекает объект из очереди и запускает его на выполнение.

Таким образом, обеспечивается строго последовательная обработка событий об изменениях файлов.

Существует два вида репликации – синхронная и асинхронная[1]. В данном случае наиболее целесообразной является асинхронная синхронизация.

Синхронная репликация может ускорить процесс распространения изменений, по тому, что изменение отправляется получателям сразу после его возникновения. Этот вариант подходит для систем, которые обновляются единичными транзакциями, например, базы данных. Однако такой подход будет не так эффективен при репликации файлов. В этом случае одно обновление может происходить за несколько транзакций. К примеру, копирование каталога создает по одной транзакции на каждый файл каталога и еще одну транзакцию на сам каталог.

При асинхронной репликации данные обновляются не мгновенно, а спустя некоторое время, которое задается в параметрах системы репликации. Из-за этого данные могут некоторое время находиться в устаревшем состоянии, однако такой проход существенно снижает затраты на установку-разрыв соединения. В случае репликации файлов такой подход дает существенный выигрыш по сравнению с синхронной репликацией. При копировании каталога, будет скопировано много файлов, однако информация об этом будет передана в рамках одной транзакции, а не множества, как в случае с синхронной репликацией.

Выводы. Были рассмотрены проблемы, которые возникли в процессе проектирования системы репликации пользовательских файлов. Были описаны алгоритмы авторизации и обработки поступающих данных о локальных изменениях файлов.

Список литературы

1. Репликация - википедия Интернет-ресурс. - Режим доступа: www/ URL: http://ru.wikipedia.org/Репликация_(вычислительная_техника)

2. Класс FileSystemWatcher – документация MSDN. Интернет-ресурс. - Режим доступа: www/ URL: http://msdn.microsoft.com/ru-ru/library/system.io.filesystemwatcher.aspx