Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
« ____ » _____ 2019г.

# МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Программный комплекс для блочной репликации пользовательских данных в корпоративных сетях
тема

09.04.01 – «Информатика и вычислительная техника»
код - наименование направления

09.04.01.04 — «Технология разработки программного обеспечения»
код и наименование магистерской программы

| | | | |
|---|---|---|---|
| Руководитель | 26.06.19 | канд. техн. наук, доцент | А.И. Постников |
| | подпись, дата | должность, ученая степень | инициалы, фамилия |
| Выпускник | | | М.С. Бриченко |
| | подпись, дата | | инициалы, фамилия |
| Рецензент | | канд. техн. наук | Н.Г. Кузьменко |
| | подпись, дата | должность, ученая степень | инициалы, фамилия |
| Нормоконтролер | 26.06.19 | | В.И. Иванов |
| | подпись, дата | | инициалы, фамилия |

Красноярск 2019

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
« ____ »_____ 2019г.

**ЗАДАНИЕ**
**НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ**

**в форме** магистерской диссертации
бакалаврской работы, дипломного проекта, дипломной работы, магистерской диссертации

Студенту Буриченко Михаилу Сергеевичу

фамилия, имя, отчество

Группа КИ17-01-04М Направление (специальность) 09.04.01.04

номер код

Технология разработки программного обеспечения

наименование

Тема выпускной квалификационной работы Программный комплекс для блочной репликации пользовательских данных в корпоративных сетях

Утверждена приказом по университету № _____ от _____

Руководитель ВКР А.И. Постников, канд. техн. наук, доцент, доцент кафедры вычислительной техники ИКИТ СФУ

инициалы, фамилия, должность, ученое звание и место работы

Исходные данные для ВКР Задание на ВКР, существующие программные средства, технологии и механизмы для репликации данных в сетях связи

Перечень разделов ВКР Задание на ВКР, анализ задания, анализ предметной области, выбор программных средств, структура программного комплекса, протокол локального обнаружения, протокол обмена блоками, централизованное и децентрализованное хранилище данных

Перечень графического материала Презентация, выполненная с помощью ONLYOFFICE Desktop Editors 5.2.83.480

Руководитель ВКР _____ А.И. Постников

подпись инициалы и фамилия

Задание принял к исполнению _____ М.С. Буриченко

подпись, инициалы и фамилия студента

« 16 » февраля 2018 г.

# РЕФЕРАТ

Выпускная квалификационная работа по теме: «Программный комплекс для блочной репликации пользовательских данных в корпоративных сетях». Пояснительная записка содержит 55 страниц текстового документа, 13 иллюстрации, 3 таблицы и 8 использованных источников.

ПРОГРАММНЫЙ КОМПЛЕКС, БЛОЧНАЯ РЕПЛИКАЦИЯ ДАННЫХ, АЛГОРИТМ, БЛОК-СХЕМА.

Цель работы – разработать программный комплекс, позволяющий выполнять блочную репликацию пользовательских данных между несколькими вычислительными машинами в корпоративных сетях связи.

В процессе работы был произведен анализ теоретических сведений по заданному вопросу, была выбрана основа и разработана структура программного комплекса.

Результатом работы является реализация указанного выше программного комплекса с подробным описанием структуры и алгоритмов работы.

**СОДЕРЖАНИЕ**

# ВВЕДЕНИЕ

Мы живём в век стремительно развивающихся технологий. На сегодняшний день большинство частных и государственных компаний заинтересованы в автоматизировании процессов, возможности удалённого и непосредственного контроля производственной деятельности своих работников, а также обеспечения поддержания необходимого стандарта в работе. Вышеуказанные меры позволят увеличить производительность, сократить издержки на производстве, сократить количество производственного брака, а равно и ошибок в соблюдении утверждённых в организации стандартов, а также приведёт к уменьшению нецелевых и вторичных затрат связанных с обеспечением производства. Оптимизации расходов и уменьшение рисков возможных потерь в свою очередь приведёт к увеличению доходов, которые в свою очередь могут быть перенаправлены на инновации, развитие социальной и экономической сферы в организации. Поэтому с каждым годом возрастает количество работников использующих электронно-вычислительные машины для выполнения своих повседневных должностных обязанностей.

Сохранность годами накопленной информации обычно ложится на плечи сотрудников отдела информационных технологий (ИТ). Чаще всего это — единые информационные системы (ЕИС), базы данных, сервисы и службы. Всё остальное, те данные, которые находятся на рабочих станциях пользователей, обычно на совести самих пользователей. Документы, служебные записки, отчеты и сканы. Потеря этих данных не критична для организации в целом, но это проблема. Проблема которая требует решения.

Для решения подобного рода проблем требуется разработка актуальных программных средств, комплексов и систем. Как с применением уже существующих технологий, так и на основе собственных разработок.

Цель магистерской диссертации — разработать программный комплекс для блочной репликации пользовательских данных в корпоративных сетях,

который позволяет — автоматизировать перенос данных между рабочими станциями и их резервное копирование, что положительно скажется на рабочем процессе.

## 1 Задание на ВКР

Заданием на выпускную квалификационную работу является разработка программного комплекса, позволяющего реплицировать (синхронизировать) пользовательские данные блочным способом в корпоративных сетях.

Исходные данные:

– существующие программные средства, технологии и механизмы для репликации данных в сетях связи;

Программный комплекс должен обеспечивать:

– централизованное управление;

– централизованное и децентрализованное хранение данных;

– автоматическое обнаружение участников обмена в пределах широковещательного домена (одной подсети);

– авторизацию участников обмена;

– ведение журнала операций;

– сбор статистики обмена;

– поддержку уведомлений (информационные сообщения для администратора системы);

– поддержку блочной репликации данных (при изменении больших данных, передаются только те блоки, в которых содержатся изменения);

– поддержку последовательных версий файлов;

– поддержку автономной работы;

– поддержку распределения нагрузки на каналы связи (балансировка);

– поддержку защищенных каналов связи (SSL/TLS).

Системные требования:

– электронно-вычислительная машина (ЭВМ) с архитектурой x86_64;

– операционная система «Microsoft Windows 7 SP1» и выше или «GNU/Linux»;

– наличие подключения к сети связи.

## 2 Анализ задания

Разработка программного комплекса, позволяющего реплицировать (синхронизировать) пользовательские данные блочным способом в корпоративных сетях сводится к:

– анализу предметной области (пониманию проблем, существующих программных средств, технологий и механизмов реплицирования данных);

– выбору программных средств для реализации алгоритмов.

### 2.1 Анализ предметной области

Потеря данных – один из самых серьезных инцидентов, которые могут произойти в ИТ-инфраструктуре. Это данные сотрудников, накопленные годами, данные сервисов и служб, бухгалтерские и кадровые данные – их потеря недопустима.

На современных компьютерах, обладающих достаточными мерами безопасности, относительно редко происходит повреждение или потеря данных. Но иногда это всё же случается.

Чаще всего данные повреждаются во время записи на диск. При сохранении документа могут возникнуть разного рода ошибки. Современное программное обеспечение в большинстве случаев способно обработать такие ошибки и предложить варианты решения проблемы. Но иногда, к сожалению, пользователь не подозревает об этом, пока не попытается через некоторое время получить доступ к данным.

При завершении работы операционной системы, пользователю предлагается должным образом сохранить все файлы, с которыми он работает в данный момент. Когда этого не происходит – например, в случае аварийного отключения питания – у программ нет возможности закрыть все файлы правильно. Это может привести к повреждению данных, которые были

открыты в этот момент, включая не только рабочие данные сотрудника, но и самой операционной системы.

Проблемы с жестким диском также могут привести к повреждению данных. Наличие дефектных физических секторов на диске, механические повреждения также являются частой причиной повреждения информации.

И, конечно, вредоносные программы могут испортить данные. Часто факт заражения компьютера обнаруживается, когда большая часть информации повреждена или уничтожена.

Для того чтобы решить проблему, необходимо периодически создавать резервные копии если не всех, то хотя бы критических данных. Желательно на другое устройство. Желательно на несколько.

Устройства для резервного копирования данных могут быть самыми разными. Компакт-диски, флеш-накопители и накопители на жестких магнитных дисках. Даже магнитные ленты до сих пор используются. Это то, что доступно каждому.

У любого из этих вариантов есть свои преимущества и недостатки. Идеального решения, которое подойдет каждому и в любой ситуации — нет.

Флеш-накопители и накопители на жестких магнитных дисках достаточно компактны и высокопроизводительны, но не отличаются хорошей отказоустойчивостью. Одни имеют ограниченное количество циклов записи, другие очень чувствительны к физическому воздействию.

Компакт-диски – достаточно надежный и дешевый способ хранить резервные копии данных при условии, что эти данные не будут изменяться со временем. Фото и видео архивы, например. Или комплект документов за закрытый отчетный период.

Еще одним способом предотвратить потерю информации являются облачные хранилища. В наше время существует огромное количество сервисов предоставляющих услуги по хранению данных на виртуальных серверах. Они доступны 24 часа в сутки, 7 дней в неделю, при условии наличия достаточно

стабильного подключения к глобальной сети интернет.

Такие сервисы работают на основе сертифицированных отказоустойчивых центров обработки данных. Многие из них предоставляют дополнительные услуги, например, хранение нескольких последних версий определенных пользователем файлов.

Так как подключение к глобальной сети может быть нестабильным или не быть вовсе, многие облачные хранилища поддерживают определенные механизмы репликации данных.

Термин «репликация» часто применяется в сфере информационных технологий. Под ним понимается автоматический или полуавтоматический процесс копирования данных из одного хранилища на другое (или на множество других) и наоборот. Иными словами, когда между несколькими копиями данных ликвидируются различия. Некий механизм синхронизации.

Различные механизмы репликации данных в облачные хранилища позволяют избавиться от необходимости постоянного доступа к глобальной сети, оставив нерешенным лишь вопрос доверия. Доверия хранения своих данных третьим лицам.

Далее будут рассмотрены несколько технологий и программных средств для репликации данных в сетях связи.

### 2.1.1 Перемещаемый профиль (Roaming Profile)

«Roaming Profile» (перемещаемый профиль пользователя) является частью операционных систем «Microsoft Windows», семейства «Microsoft Windows NT» (»Microsoft Windows 2000» и выше). Данная технология позволяет пользователям получить доступ к своему профилю на компьютерах, подключенных к домену «Active Directory» в пределах корпоративной сети. В профиль пользователя включаются не только документы, но и настройки окружения и установленных программ, а также ассоциации файлов.

Преимущества использования данной технологии в следующем:

– имеет ограниченное централизованное управление через оснастку «групповые политики» в домене «Active Directory» (где хранить эталонную копию данных, какие директории игнорировать при перемещении);

– перемещаемый профиль позволяет пользователю автоматически получать доступ к своим данным с любого компьютера с операционной системой семейства «Windows NT», подключенного к корпоративной сети и входящего в домен домен «Active Directory»;

– использование перемещаемых профилей упрощает резервное копирование данных пользователей, предотвращая потерю данных при выходе из строя носителей информации на рабочих станциях;

– при замене компьютера нет необходимости вручную переносить документы и настройки пользователя на другую рабочую станцию, т.к. при входе в домен они автоматически загружаются с домена.

При всех преимуществах, недостатки данной технологии очень значительные и заключаются в следующем:

– загрузка данных на рабочую станцию происходит только во время входа пользователя в систему, а выгрузка только во время выхода, что делает невозможным полноценно взаимодействовать одновременно с несколькими устройствами;

– пользователь не может взаимодействовать с системой, пока профиль полностью не загрузится с домена (в случае с беспроводным или ограниченным подключением к корпоративной сети это может занимать десятки минут);

– для выгрузки профиля в случае разрыва соединения с доменом (отключение электропитания или активного оборудования), пользователю требуется выполнить повторный вход и выход из системы;

– проблемы с ассоциациями файлов на перенесенном профиле, когда на рабочих станциях установленно разное программное обеспечение закрепленное за одним и тем же расширением файлов («Microsoft Office» на одной и

«LibreOffice» на другой). Например вы работали с файлами «*.docx» в «Microsoft Word». При перемещении профиля на другую рабочую станцию операционная система будет искать именно эту программу при попытке открыть файл «*.docx». Пользователю придется вызывать «контекстное меню», вызывать метод «Открыть с помощью...» и сохранять выбранную программу для определенных типов файлов. При работе на предыдущей рабочей станции придется повторять процесс заново, так как изменения перенесутся и на неё.

### 2.1.2 Яндекс.Диск

Яндекс.Диск — сервис, который позволяет хранить файлы на удаленных серверах Яндекса и обмениваться ими [8]. Данные доступны авторизованному пользователю с любого устройства, подключенного к интернету.

В бесплатной версии предоставляется 10ГиБ для хранения пользовательских данных без возможности управления несколькими последними версиями файлов.

Официальные клиенты для платформ Windows, Linux, MacOS, Android и iOS. Управление файлами в облачном хранилище так же возможно через WEB портал сервиса.

После установки клиента указывается папка для репликации данных. Файлы загружаются в облачное хранилище и дальнейшем распространяются на все подключенные устройства. Загрузка производится автоматически при подключении устройств к глобальной сети. При обновлении существующих файлов берется самый новый по дате изменения. Следовательно время на всех подключенных устройствах должно быть точным, во избежание непредвиденных ситуаций. При одновременном редактировании и попытке загрузки файла с двух и более устройств, в зависимости от программы клиента, пользователю будет предоставлен выбор — выбрать необходимую версию или сохранить все, добавив порядковый индекс к имени. Если на устройстве

недостаточно свободного места (смартфон, например), клиент позволяет выбрать данные, которые будут загружены только во время обращения к ним.

Таким образом, в зависимости от настроек, данные будут храниться как в облаке, так и на подключенных устройствах.

Если на определенном устройстве велик риск повреждения данных вредоносным программным обеспечением, целесообразно изменить поведение клиента с автоматической загрузки данных в облако, на загрузку по требованию.

В случае, когда для определенной операционной системы нет официального клиента, а работа с файлами через WEB портал неудобна, то сервис поддерживает управление файлами через технологию Web Distributed Authoring and Versioning (WebDAV).

WebDAV (или просто DAV) – набор расширений и дополнений к протоколу HyperText Transfer Protocol (HTTP), поддерживающих совместную работу пользователей над редактированием файлов и управление файлами на удаленных веб-серверах. Сегодня DAV применяется в качестве сетевой файловой системы, эффективной для работы в глобальной сети и способной обрабатывать файлы целиком, поддерживая хорошую производительность работы в условиях окружения с высокой временной задержкой передачи информации. Кроме того, DAV широко применяется в качестве протокола для манипулирования содержимым систем документооборота (Document Management System). В качестве резюме задачу DAV можно указать так: использование HTTP в качестве стандартного уровня доступа к широкому кругу хранилищ информации и расширить его возможности средствами записи информации (HTTP – доступ на чтение, DAV – доступ на запись).

WebDAV расширяет HTTP следующими методами запроса:

– PROPFIND – Получение свойств объекта на сервере в формате XML. Также можно получать структуру репозитория (дерево каталогов);

– PROPPATCH – Изменение свойств за одну транзакцию;

– MKCOL – Создать коллекцию объектов (каталог в случае доступа к файлам);

– COPY – Копирование из одного URI в другой;

– MOVE – Перемещение из одного URI в другой;

– LOCK – Поставить блокировку на объекте. WebDAV поддерживает эксклюзивные и общие (shared) блокировки;

– UNLOCK – Снять блокировку с ресурса.

Основные возможности:

– блокировка – долгосрочные блокировки на запись документа предотвращают потерю информации при одновременном редактировании документа несколькими пользователями. Учитывая характер Интернет-соединений пользователей, длительность блокировки в DAV не зависит от индивидуального сетевого соединения;

– свойства – произвольные метаданные могут храниться в качестве свойств, описанных на языке eXtensible Markup Language (XML). В качестве примера можно привести список авторов документа или его краткую аннотацию. Протокол DAV предоставляет средства для создания, редактирования и удаления свойств. Протокол DAV Searching and Locating (DASL) предоставляет средства поиска и локализации веб-ресурсов на основе значения их свойств;

– именованные области (Namespace) – Раздел DAV, поддерживающий переименование и перемещение веб-ресурсов. Механизм реализован в протоколе с помощью коллекций, выполняющих функцию, аналогичную папкам в файловой системе.

Работа WebDAV регулируется следующими стандартами:

– RFC2291 «Requirements for a Distributed Authoring and Versioning Protocol for the World Wide Web»;

– RFC4918 «HTTP Extensions for Distributed Authoring – WEBDAV»;

– RFC3648 «Web Distributed Authoring and Versioning (WebDAV)

Ordered Collections Protocol»;

– RFC3744 «Web Distributed Authoring and Versioning (WebDAV) Access Control Protocol».

Поддержка WebDAV имеется в большинстве операционных систем.

Для подключения облачного хранилища на платформе Windows необходимо:

1. В «проводнике» (стандартный менеджер файлов) в меню вызвать метод «подключить сетевой диск»;
2. Указать индекс (букву) для подключения диска;
3. Ввести адрес: https://webdav.yandex.ru
4. Нажать «Подключить»;
5. Ввести логин/пароль.

Для «Portable Operating System Interface» (POSIX) совместимых платформ подключение хранилища выполняется командой (необходим пакет davfs2):

# mount -t davfs https://webdav.yandex.ru dest, где dest — директория в которую подключается удалённый ресурс (например «/home/username/cloud»).

Загружаться и выгружаться данные будут во время доступа к ним.

При всех имеющихся преимуществах, сервис имеет несколько недостатков:

– данные хранятся на серверах сторонней организации (неизвестно кто еще может получить доступ к вашим данным);

– отсутствует поддержка версий файлов (в платной версии обещали добавить в будущем);

– при изменении данных, файлы перезагружаются целиком. Если файлы достаточно большие, то процедура репликации доставляет очень большие неудобства. При разрыве соединения, не полностью загруженные файлы, загружаются сначала.

### 2.1.3 ownCloud (NextCloud)

ownCloud — это свободное и открытое веб-приложение для репликации данных, общего доступа к файлам и удалённого хранения документов в «облаке» [7].

Приложение написано на языках программирования PHP и JavaScript. Поддерживает базы данных SQLite, MariaDB, MySQL, Oracle Database и PostgreSQL. Работает на серверах под управлением операционных систем Linux. В ресурсах не прихотлив.

Разработку ownCloud начал один из разработчиков KDE, Франк Карличек, в январе 2010 г. Он стремился создать бесплатную альтернативу коммерческим облачным сервисам хранения данных. В отличие от них, ownCloud можно установить на собственный сервер без дополнительных затрат.

ownCloud и Яндекс.Диск схожи по функционалу, за исключением того, что в данном случае серверная часть облака располагается на собственном персональном компьютере (например на домашнем).

Хоть серверная часть ownCloud разрабатывается для операционных систем Linux, доступна в стандартных репозиториях приложений и устанавливается в несколько команд, её можно запустить и на Windows. Для этого требуется:

1. включить в «программы и компоненты» службу Internet Information Services (IIS) с консолью управления сервером и поддержкой Common Gateway Interface (CGI);
2. загрузить с официального сайта и распаковать PHP требуемой версии (в зависимости от версии ownCloud);
3. добавить «php-cgi.exe» в консоли управления IIS как FastCGI модуль;
4. загрузить с официального сайта и распаковать дистрибутив ownCloud в директорию: «C:\inetpub\wwwroot\»;

5. открыть WEB-браузер и перейти по адресу «https://localhost» для первоначальной настройки.

Если такой способ установки кажется сложным или неудобным, на официальном сайте разработчика ownCloud можно загрузить подготовленные образы виртуальных машин для VirtualBox, VMWare и Hyper-V.

Для подключения к серверу из внешней сети, на компьютере или маршрутизаторе домашней сети должен быть белый (реальный) IP адрес и открыт для входящих подключений 443 порт (HTTPS).

Доступны клиенты для репликации данных с ПК под управлением операционных систем Windows, Mac OS X, Linux и с мобильными устройствами на iOS, Android. Кроме того, все сохраненные данные доступны через веб-интерфейс ownCloud в любом браузере.

Также поддерживается шифрование и работа с данными через WebDAV.

За исключением места хранения всех данных, ownCloud имеет тот же значительный недостаток что и Яндекс.Диск:

– отсутствует поддержка версий файлов;

– при изменении данных, файлы перезагружаются целиком. Если файлы достаточно большие, то процедура репликации доставляет очень большие неудобства. При разрыве соединения, не полностью загруженные файлы, загружаются сначала.

Использование своего «облака» требует от владельца определенные трудовые (временные) и финансовые затраты, да и проблема репликации больших данных в условия «бездорожья» на линиях связи актуальная как никогда. Поэтому далее будут рассмотрено способы репликации данных по частям (блочная репликация) без использования «облаков».

### 2.1.4 Resilio Sync (ранее «BitTorrent Sync»)

Resilio Sync — сервис для репликации файлов и резервного копирования

по протоколу BitTorrent между произвольными устройствами [6]. Поддерживается компанией «Resilio, Inc.», выделенной из «BitTorrent, Inc.».

Сервис был разработан для того, чтобы решить фундаментальные проблемы репликации данных:

– ограничения на скорость, размер и пространство;

– ограничения на безопасность файла и зависимость от облачной инфраструктуры.

Трансферы происходят в зашифрованном виде, а информация не сохраняется на каком-либо сервере или в облаке. Ваше содержимое принадлежит вам и остается на устройствах по вашему выбору.

Принцип работы протокола «BitTorrent» достаточно прост. Перед началом скачивания клиент подсоединяется к специальному трекеру, сообщает ему свой адрес и хеш-сумму файлов, на что в ответ получает адреса других клиентов, скачивающих или раздающих эти же файлы. Далее клиент периодически информирует трекер о ходе процесса и получает обновлённый список адресов. Этот процесс называется объявлением (announce).

Клиенты соединяются друг с другом и обмениваются сегментами файлов без непосредственного участия трекера, который лишь хранит информацию, полученную от подключенных к обмену клиентов и список самих клиентов. Для эффективной работы сети «BitTorrent» необходимо, чтобы как можно больше клиентов были способны принимать входящие соединения.

При подключении к глобальной сети через NAT (Network Address Translation), требуется дополнительная настройка (проброс портов для входящих подключений). Если входящие подключения на клиенте будут запрещены или ограничены, то этот клиент сможет только получать обновленные данные, но не сможет их раздавать (подходит для устройств, на которых будут храниться резервные копии).

При соединении клиенты сразу обмениваются информацией об имеющихся у них сегментах. Клиент, желающий скачать сегмент (личер),

посылает запрос и, если второй клиент готов отдавать, получает этот сегмент. После этого клиент проверяет контрольную сумму сегмента. Если она совпала, то сегмент считается успешно скачанным, и клиент оповещает всех присоединённых пиров о наличии у него этого сегмента. Если же контрольные суммы различаются, то сегмент начинает скачиваться заново. Некоторые клиенты банят тех пиров, которые слишком часто отдают некорректные сегменты.

Таким образом, объём служебной информации напрямую зависит от количества, а значит, и размера сегментов. Поэтому при выборе сегмента необходимо соблюдать баланс: с одной стороны, при большом размере сегмента объём служебной информации будет меньше, но в случае ошибки проверки контрольной суммы придётся заново скачивать больше информации. С другой стороны, при малом размере ошибки не так критичны, так как необходимо заново скачать меньший объём.

Обмен сегментами ведётся по принципу «ты — мне, я — тебе» симметрично в двух направлениях. Клиенты сообщают друг другу об имеющихся у них сегментах при подключении и затем при получении новых сегментов, и поэтому каждый клиент может хранить информацию о том, какие сегменты есть у других подключенных пиров. Порядок обмена выбирается таким образом, чтобы сначала клиенты обменивались наиболее редкими сегментами: таким образом повышается доступность файлов в раздаче. В то же время выбор сегмента среди наиболее редких случаен, и поэтому можно избежать ситуации, когда все клиенты начинают скачивать один и тот же самый редкий сегмент, что негативно бы отразилось на производительности.

Сегменты делятся на блоки определенного размера, и каждый клиент запрашивает именно эти блоки. Одновременно могут запрашиваться блоки из разных сегментов. Более того, клиенты поддерживают скачивание блоков одного сегмента у разных пиров.

Текущие сборки доступны для следующих операционных систем:

«Microsoft Windows» (»Microsoft Windows 7» и выше; для «Microsoft Windows XP» последняя рабочая версия 1.4.111), Mac OS X (10.6 и выше), GNU/Linux, FreeBSD, Android (2.2 и выше), iOS (5 и выше), Windows Phone (8, 8.1, 10).

Достоинства «Resilio Sync» в следующем:

– нет необходимости в настройке серверной части;

– данные передаются напрямую между устройствами;

– чем больше устройств, тем больше скорость передачи данных;

– при изменении больших данных, передаются только те блоки, в которых содержатся изменения;

– можно делиться определенными директориями или файлами с другими людьми;

– на определенных клиентах можно хранить данные в зашифрованном виде. Удобно для размещения резервных копий на «virtual private server» (VPS) или «virtual dedicated server» (VDS).

Не обошлось и без недостатков:

– не поддерживаются версии файлов;

– не смотря на обещания разработчиков, исходный код до сих пор закрыт, что не позволяет провести независимый аудит.

Далее будет рассмотрен аналог «Resilio Sync», но с поддержкой версий файлов и открытым исходным кодом.

### 2.1.5 Syncthing

Syncthing — кросс-платформенное программное обеспечение с открытым исходным кодом, позволяющее реплицировать файлы между несколькими устройствами по технологии точка-точка.

Данное программное обеспечение было разработано в качестве альтернативы как распространённым закрытым сервисам облачного хранения и реплицирования файлов (Яндекс.Диск), так и распределённой репликации

данных (Resilio Sync). Оно является чем-то более открытым, заслуживающим доверия и децентрализованным. Пользовательские данные — только пользовательские. Они сами выбирают, где хранить данные, доступны ли они третьей стороне и как именно они пересылаются через Интернет.

Целью проекта было устранение основных недостатков имеющихся решений (проблемы безопасности, отсутствие открытой лицензии и т.д.). И, хотя самим автором проект никогда не позиционировался как открытая замена BitTorrent Sync, многие воспринимают его именно в этом качестве.

Syncthing может работать как в локальной сети, так и в глобальной сети интернет. Передача всех данных происходит по защищенным каналам (TLS), чтобы исключить возможность прослушивания. Репликация происходит по дате изменения файла. Есть поддержка синхронизации на уровне блоков, то есть при небольших изменениях в файле будут переданы только изменившиеся блоки, а не весь файл.

Для обнаружения устройствами друг друга используется собственный открытый протокол. Для глобального обнаружения используется один или несколько специализированных discovery-серверов, для локального — широковещательный домен. Также возможно использование собственного discovery-сервера.

Для обмена данными между устройствами используется специально разработанный протокол обмена блоками. Взаимодействие по этому протоколу происходит между двумя или более узлами сети, которые образуют кластер. Каждый узел имеет один или несколько репозиториев файлов, описанных локальной моделью, которая содержит метаданные и контрольные суммы (хэши) блоков. Локальная модель распространяется между всеми узлами в кластере. Объединение всех локальных моделей формирует глобальную модель, включающую в себя наиболее изменённые версии файлов. Каждый узел стремится синхронизировать локальное хранилище с глобальной моделью, запрашивая у других узлов в кластере отсутствующие или обновившиеся

блоки.

Syncthing не использует P2P-сеть (Point-To-Point) для обнаружения устройств или для решения проблемы NAT, но все соединения между устройствами происходят непосредственно по принципу точка-точка. Следует отметить, что возможность соединения непосредственно по IP-адресу/DNS-имени без использования обнаружения, является наиболее безопасной. Открытый ключ в паре с адресом не попадают в общий, в той или иной степени, доступ.

Основные достоинства в следующем:

– возможность полного отключения глобального обнаружения (репликация и обмен данными происходит только в пределах локальной сети;

– настройка (выбор) доверенных и недоверенных сетей (по названиям, адресам и т.д.);

– поддержка сохранения последовательных версий файла по нескольким алгоритмам, включая пользовательский, что очень важно в задачах резервного копирования;

– возможность переключения клиентов в режим только получения данных (причем на отдельные директории);

– отличная документация по всем доступным функциям и возможностям.

Официальные сборки доступны для следующих операционных систем и архитектур:

– Linux: 32 bit, 64 bit, ARM, AArch64, MIPS, mips64, mips64le, MIPS-LE, PPC64, PPC64-LE;

– Windows: 32 bit, 64 bit;

– FreeBSD: 32 bit, 64 bit, ARM;

– Solaris: 64 bit;

– Dragonfly BSD: 64 bit;

– NetBSD: 32 bit, 64 bit;

– OpenBSD: 32 bit, 64 bit, ARM;
– Mac OS: 32 bit, 64 bit;

Исходный код открыт под лицензией MPLv2.

## 2.2 Выбор программных средств

Так как некоторые существующие технологии обладают частью необходимого функционала и обладают открытым исходным кодом, рационально использовать их наработки для создания требуемого программного комплекса.

В данном разделе будет выбрана основа для программного комплекса, а так же язык программирования, среды программирования и поддерживаемые операционные системы.

### 2.2.1 Выбор основы для программного комплекса

Ранее были рассмотрены средства для организации доступа, репликации и резервного копирования данных. Одна встроенная технология, две централизованных (»Яндекс.Диск» и «ownCloud») и две децентрализованных (»Resilio Sync» и «Syncthing»). У всех из них есть определенные достоинства и недостатки.

Основной задачей было выбрать доступный и безопасный способ репликации данных сотрудников в корпоративной среде, подключенных как через локальную сеть, так и через глобальную.

«Яндекс.Диск» не подходит из-за хранения данных в своём облаке, что запрещается для критически-важных данных и данных защищаемых федералами законами (например федеральный закон "О персональных данных" N 152-ФЗ).

В ownCloud нет возможности передавать измененные данные блоками,

что пагубно сказывается на скорости передачи данных и не подходит по заданию.

«Resilio Sync» хоть и не хранит данные на своих серверах, но использует их для глобального обнаружения пиров (клиентов). В совокупности с закрытым исходным кодом это очень пагубно сказывается на безопасности.

«Syncthing» не ориентирован на использование в корпоративных системах с многопользовательским доступом (как и «Resilio Sync»). Зато открытый исходный код позволяет модифицировать его под все необходимые возможности.

Поэтому в качестве основы для разрабатываемого программного комплекса был выбран проект — «Syncting».

### 2.2.2 Выбор языка программирования

Для разработки программной системы в качестве языка программирования был выбран язык C++. Главной причиной выбора данного языка программирования стала поддержка кросс-платформенной библиотеки Qt. Использование данной библиотеки позволит облегчить создание модулей программы. Программы, написанные с использованием Qt, легче переносятся между разными платформами (операционными системами). Стоит также отметить и другие достоинства выбранного языка программирования:

– C++ содержит средства разработки программ для широкого спектра задач, от низкоуровневых утилит и драйверов до весьма сложных программных комплексов [1];

– вычислительная производительность: язык спроектирован так, чтобы дать программисту максимальный контроль над всеми аспектами структуры и порядка исполнения программы (не платишь за то, что не используешь);

– поддержка различных стилей программирования: традиционное императивное программирование, процедурное программирование,

функциональное программирование;

– поддержка автоматического вызова деструкторов объектов в адекватном порядке (обратном вызову конструкторов), что упрощает и повышает надёжность управления памятью и другими ресурсами (открытыми файлами, сетевыми соединениями, соединениями с БД);

– кросс-платформенность: стандарт языка накладывает минимальные требования на ЭВМ для запуска скомпилированных программ;

– доступность: для C++ существует огромное количество учебной литературы, переведённой на всевозможные языки. Язык имеет высокий порог вхождения, но среди всех языков такого рода обладает наиболее широкими возможностями.

Еще одна причина выбора C++ и библиотеки Qt – это «сигналы» и «слоты».

В программировании графического интерфейса, когда мы меняем один компонент, мы часто хотим, чтобы другой получил об этом уведомление. В общем случае, мы хотим, чтобы объекты любого типа могли общаться с другими. Например, если пользователь нажимает кнопку «закрыть», мы, вероятно, хотим, чтобы была вызвана функция окна «close()».

Другие библиотеки добиваются такого рода общения, используя обратный вызов. Обратный вызов – это указатель на функцию, таким образом, если мы хотим, чтобы функция уведомила нас о каких-нибудь событиях, мы передаем указатель на другую функцию (обратно-вызываемую) этой функции. Функция, в таком случае, делает обратный вызов, когда необходимо. Обратный вызов имеет два основных недостатка. Во-первых, мы никогда не можем быть уверены, что функция делает обратный вызов с корректными аргументами. Во-вторых, обратный вызов жестко связан с вызывающей его функцией, так как эта функция должна точно знать, какой обратный вызов надо делать.

В Qt используется другая техника – сигналы и слоты [2]. Сигнал вырабатывается, когда происходит определенное событие. Слот – это функция,

которая вызывается в ответ на определенный сигнал. Компоненты (виджеты) Qt имеют много предопределенных сигналов и слотов, но мы всегда можем сделать дочерний класс и добавить наши сигналы и слоты в нем.

### 2.2.3 Выбор среды программирования

На данный момент существует множество различных сред программирования (IDE), отличающихся функциональностью, удобством и востребованностью. Для разработки данной программного комплекса был выбран «Qt Creator».

Основной причиной выбора «Qt Creator» являлось то, что программный комплекс будет разрабатываться с помощью кросс-платформенной библиотеки Qt, поэтому процесс разработки будет наиболее удобен в IDE, которая идёт в комплекте. Более того, в «Qt Creator» встроен WYSIWYG (сокращение от английского «What You See Is What You Get», – что видишь, то и получишь) редактор, который позволяет в более удобном и наглядном варианте создавать интерфейс разрабатываемого программного комплекса.

### 2.2.4 Выбор операционной системы

В настоящее время пользователям предлагается большое разнообразие в выборе операционных систем (ОС). Среди них можно выделить несколько основных групп:

- Microsoft Windows NT (NT4 и старше);
- POSIX-сертифицированные/совместимые (GNU/Linux, Mac OS, BSD).

Каждая из операционных систем обладает своими особенностями, но наибольшая часть пользователей в организациях использует операционные системы семейства «Windows NT».

По условиям задания на ВКР, разрабатываемый программный комплекс

предназначен для работы в операционных системах семейства «Windows NT», начиная с «Microsoft Windows 7» (с установленным «Service Pack 1») и старше или «GNU/Linux».

## 3 Структура программного комплекса

Так как в качестве основы был выбран проект «Syncthing», то основная задача разработанного программного комплекса – автоматизированное централизованное управление взаимодействием «Syncthing» на рабочих станциях, с учетом многопользовательских операционных систем.

Программный комплекс разработан на языке программирования C++ с использованием кросс-платформенной библиотеки QT. Поддерживает операционные системы семейства «Windows NT» начиная с «Windows 7 SP1» и «GNU/Linux». Он состоит из двух частей — серверной и клиентской.

Общая структура программного комплекса представлена на рисунке 3.1.


Рисунок 3.1 — Общая структура программного комплекса

Основной модуль серверной части, а так же основной и пользовательский

модули клиентской части, представленные на рисунке 3.1 будут описаны далее.

Для модуля «Syncthing» будут описаны структуры данных (локальные и глобальные модели), протоколы обнаружения участников и обмена блоками.

## 3.1 Серверная часть программного комплекса

Серверная часть обеспечивает:

– обмен уникальными идентификаторами. Когда на новой рабочей станции впервые запускается клиентская часть, она передаёт на серверную часть уникальный идентификатор сгенерированный «Syncthing». Серверная часть уведомляет об этом администратора системы и ожидает от него ответа. При одобрении станции её идентификатор записывается в список участников обмена (список идентификаторов). Список рассылается всем участникам обмена по защищенным каналам связи. Таким образом, рабочая станция включается в кластер репликации данных. При удалении идентификатора станции из списка, он заново рассылается клиентам и дальнейший обмен данными с этим участником прекращается;

– сбор статистики участников обмена. «Syncthing» хранит подробную статистику репликации данных (какие данные, когда и куда они посылаются, возникающие ошибки). Данные посылаются клиентской частью через защищенные каналы связи;

– хранение данных всех пользователей системы. Так как для реплицирования данных требуется минимум два активных участника обмена (две одновременно включенных рабочих станции), то в серверную часть так же включен «Syncthing», который собирает данные со всех участников обмена. Так обеспечивается избыточность – участник обмена, который всегда активен и хранит актуальные данные для репликации;

– передача клиентской части дистрибутива «Syncthing», для обновления до актуальной версии;

– управление настройками клиентской части.

Блок-схема алгоритма серверной части представлена на рисунке 3.1.1.

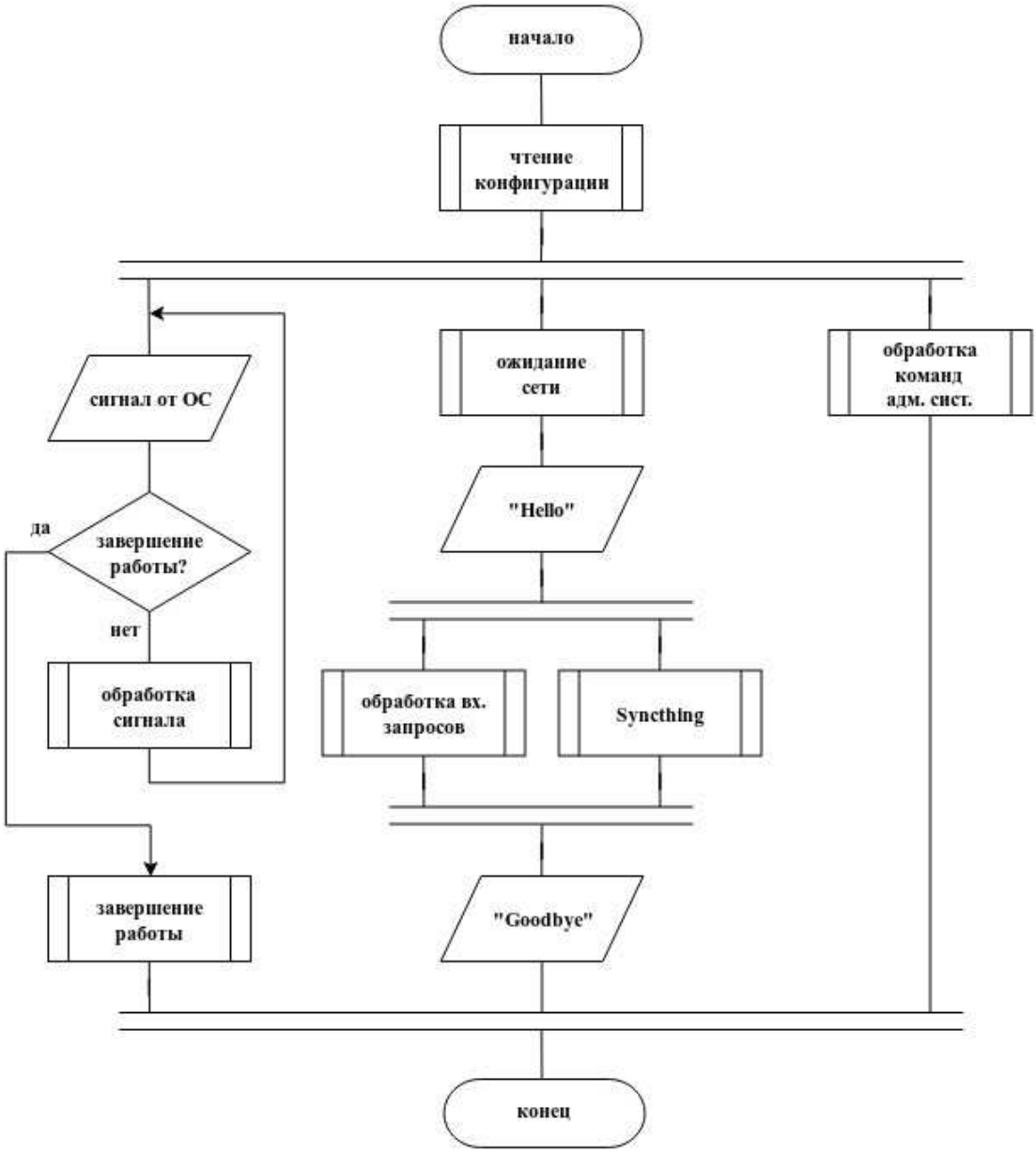


Рисунок 3.1.1 — Блок-схема алгоритма серверной части

Алгоритм серверной части, представленный на рисунке 3.1.1, разделяется на несколько параллельных потоков:

– обработка событий (сигналов) от операционной системы. К примеру

обработка сигнала «SIGTERM», отвечающего за завершение процессов в POSIX-совместимых операционных системах;

- обработка входящих запросов от клиентской части.
- обработка команд администратора системы;
- запуск «Syncthing».

Запросы и команды поступают по защищенным каналам связи.

Защищенные соединения описаны в главе 3.1.1 пояснительной записки.

Каждый поступивший запрос от клиентской части добавляется в очередь на исполнение (выполнение соответствующих действий). Запросы из очереди выполняются в порядке поступления (первый вошел, первый вышел), в количество потоков установленных в конфигурационном файле (минимум 1). Для выполнения команд администратора системы выделяется отдельный поток.

Количество потоков устанавливается системным администратором серверной части, путем запуска исполняемого файла серверной части с аргументом «threads=N», где N — количество потоков (от 1 до 255).

Указывать потоки следует с осторожностью, так как слишком малое число потоков приведет к «подвисанию» (задержкам обработки запросов и команд) серверной части, в случае, когда клиентов достаточно много. Слишком большое число потоков может исчерпать ресурсы вычислительной машины, на которую установлена серверная часть.

Структура очереди запросов и команд описана в главе 3.1.2 пояснительной записки.

### 3.1.1 Защищенные соединения

Для установления защищенного клиент-серверного соединения используются сертификаты открытого ключа (стандарт X.509). Эти сертификаты предоставляются удостоверяющими центрами (УЦ). Кроме того, X.509 определяет формат списка аннулированных сертификатов и алгоритм

проверки подписи путём построения пути сертификации.

X.509 предполагает наличие иерархической системы удостоверяющих центров для выдачи сертификатов.

Для каждого экземпляра (клиента или сервера) удостоверяющим центром выдаётся собственный уникальный (именной) сертификат, подписанный секретным ключом удостоверяющего центра.

Так как программный комплекс предназначен для использования в изолированных (от внешнего мира, интернета) корпоративных сетях, то допускается использование удостоверяющего центра организации. Для этого требуется использовать программу «openssl» (входит в стандартный набор программ операционных систем «GNU/Linux»).

Процесс создания сертификатов и секретных ключей не является темой данной пояснительной записки, поэтому рассматриваться не будет. Но для понимания процесса авторизации рассмотрим содержимое сертификата стандарта X.509, которое представлено на рисунке 3.1.1.1.

```
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      8e:2b:d7:ae:77:7a:15:bd
  Signature Algorithm: GOST R 34.11-94 with GOST R 34.10-2001
    Issuer: C = RU,
            ST = Krasnoyarsk Krai,
            L = Krasnoyarsk,
            O = SomeOrg,
            CN = Server,
            emailAddress = it@dreamteam.ru
    Validity
      Not Before: Mar  9 02:40:22 2019 GMT
      Not After : Jan  1 00:00:00 2050 GMT
    Subject:  C = RU,
              ST = Krasnoyarsk Krai,
              O = SomeOrg,
              OU = MAIN,
              CN = Server,
              emailAddress = it@dreamteam.ruSubject
    Public Key Info:
        Public Key Algorithm: GOST R 34.10-2001
    X509v3 extensions:
      X509v3 Basic Constraints:
        CA:FALSE
      X509v3 Subject Key Identifier:
        FD:96:58:12:20:EF:D0:B9:E1:E8:15:D8:DC:A9:D5:99:65:66:76:65
      X509v3 Authority Key Identifier:
        keyid:FD:96:58:12:20:EF:D0:B9:E1:E8:15:D8:DC:A9:D5:99:65:66:76:6
        serial:8a:2b:d7:ae:77:7a:15:bd
      X509v3 Key Usage: critical
        Digital Signature, Key Encipherment
      X509v3 Extended Key Usage: critical
        TLS Web Server Authentication
      Netscape Cert Type:
        SSL Server
  Signature Algorithm: GOST R 34.11-94 with GOST R 34.10-2001
    69:8c:0d:ce:51:47:05:c6:93:94:d0:81:fe:80:c9:5d:6e:fe:
    ...
    4b:a9:16:94:35:6b:84:b6:06:86:8d:b9:f3:98:ec:4c:2f:a5:
```

Рисунок 3.1.1.1 — Содержимое сертификата стандарта X.509

Как видно из рисунка 3.1.1.1, в каждом сертификате в обязательном порядке содержатся следующие данные:

– серийный номер;

– срок годности (от и до);
– кем и кому выдан;
– разрешения;
– алгоритм формирования ключа.

При установлении защищенного соединения как сервер, так и клиент проверяют (сравнивают) сертификаты друг-друга на соответствие необходимым требованиям:

– сертификаты выданы одним удостоверяющем центром;
– сертификаты предназначены для программного комплекса;
– срок действия сертификатов не истек.

Если сертификаты соответствуют требованиям, то защищенное соединение устанавливается.

Следует отметить, что в представленном на рисунке 3.1.1.1 сертификате используется алгоритм GOST, который обязателен (требование ФСТЭК, ФСБ) для защищенных каналов связи, в которых передаются персональные данные (федеральный закон "О персональных данных" N 152-ФЗ).

### 3.1.2 Структура очереди запросов и команд

Очередь запросов и команд представляет из себя список действий для выполнения. Каждое действие может обладать набором разнообразных аргументов. Обязательным аргументами для каждого действия являются — действие (описаны в разделах 3.1.4 и 3.1.5 пояснительной записки) и идентификатор участника (описан в разделе 3.1.3).

Примерная структура очереди запросов и команд представлена на рисунке 3.1.2.1.

[0]:
идентификатор: MFZWI3D-BONSGYC-YLTMRWG-C43ENR5-QXGZDMM-FZWI3DP-BONSGYY-LTMRWAD
действие: запрос
запрос: получение списка идентификаторов участников обмена

[1]:
идентификатор: AFZWI3D-BONSGYC-YLTMRWG-C43ENR5-QXGZDMM-FZWI3DP-BONSGYY-LTMRWAD
действие: команда
команда: отправить список идентификаторов участников обмена
кому: MFZWI3D-BONSGYC-YLTMRWG-C43ENR5-QXGZDMM-FZWI3DP-BONSGYY-LTMRWAD
...
[n]:...

Рисунок 3.1.2.1 — Примерная структура очереди запросов и команд

### 3.1.3 Идентификатор участника обмена

Идентификатор участника обмена представляет из себя контрольную сумму от индивидуального (уникального) сертификата.

Контрольная сумма генерируется с использованием алгоритма SHA-256, кодируется алгоритмом «Base32» и разделяется на несколько блоков.

Пример идентификатора участника обмена представлен на рисунке 3.1.3.1.

MFZWI3D-BONSGYC-YLTMRWG-C43ENR5-QXGZDMM-FZWI3DP-BONSGYY-LTMRWAD

Рисунок 3.1.3.1 — Пример идентификатора участника обмена

Идентификатор автоматически создаётся средствами «Syncthing» при первом запуске на рабочей станции, будь то клиент или сервер.

В списке идентификаторов дополнительно содержатся имена участников обмена, которым принадлежат эти идентификаторы. Именем может быть что угодно, например — инвентарные номера вычислительных машин.

### 3.1.4 Входящие запросы от клиентской части

В соответствие с разделом 3.1.2 пояснительной записки, все поступающие запросы от клиентской части добавляются в очередь планировщика и обрабатываются им по порядку поступления.

Список доступных запросов и их аргументов представлен в таблице 3.1.4.1.

Таблица 3.1.4.1 — Список доступных запросов и их аргументов

| Запрос | Аргументы | Описание |
|---|---|---|
| auth | Идентификатор участника (cid) | Запрос авторизации клиентской части для дальнейшей возможности получения списка идентификаторов и конфигурации пользователей. |
| get cids | Идентификатор участника (cid) | Запрос на получение списка идентификаторов участников обмена |
| get config | Идентификатор участника (cid)<br>Идентификатор пользователя (uid) | Запрос на получение конфигурации клиентской части (если указан только cid)<br>Если указан uid, то запрос на получение конфигурации пользователя (если указан еще и uid) |

Запрос «auth» отправляется первым при установлении соединения с серверной частью. Пока результат обработки этого запроса не равен «approved» (одобрено), серверная часть будет приостанавливать выполнение всех остальных запросов в очереди от этого клиента (клиентской части).

Статус «approved» будет отправлен только если администратор системы добавит идентификатор клиентской части в список участников обмена.

### 3.1.5 Команды администратора системы

В соответствие с разделом 3.1.2 пояснительной записки, все поступающие команды от администратора системы добавляются в очередь

планировщика и обрабатываются им по порядку поступления, в отдельном потоке.

Список доступных команд и их аргументов представлен в таблице 3.1.5.1.

Таблица 3.1.5.1 — Список доступных запросов и их аргументов

| Команда | Аргументы | Описание |
|---|---|---|
| add cid | Идентификатор участника (cid) | Добавление идентификатора участника обмена в список участников обмена |
| add user | Идентификатор пользователя (uid) | Добавление пользователя для последующего хранения его файлов на серверной части |
| del cid | Идентификатор участника (cid) | Удаление идентификатора участника обмена из списка участников обмена |
| del user | Идентификатор пользователя (uid) | Удаление пользователя как участника обмена. |
| get config | Идентификатор участника (cid)<br>Идентификатор пользователя (uid) | Получение конфигурации клиентской части с указанным cid. Если указан идентификатор пользователя (uid) — получение конфигурации конкретного пользователя. |
| get status | Идентификатор участника (cid) | Получение текущего статуса клиентской части с указанным cid (режим работы и т.д.). Если вместо cid указать параметр «all», то будет получен статус всех участников обмена, в алфавитном порядке. |
| set config | Идентификатор участника (cid)<br>Идентификатор пользователя (uid) | Установка конфигурации клиентской части (если указан только cid) и пользователя (если указан uid). |
| set mode | Идентификатор участника (cid)<br>Режим (mode) | Включить/выключить режим автономной работы |
| show cids | Идентификатор участника (cid) | Отобразить список авторизованных (добавленных) участников обмена |
| show config | Идентификатор участника (cid) | Отобразить текущую конфигурацию клиентской части с указанным cid. Допускается параметр — all. |
| show log | Идентификатор участника (cid) | Отобразить журнал |
| upd | Идентификатор участника (cid) | Обновление модуля «Syncthing» для клиентской части с указанным идентификатором (cid). Допускается параметр — all. |

### 3.1.6 Сбор статистики и журналирование действий

Каждый экземпляр клиентской части записывает все выполненные запросы и полученные команды от серверной части. Все эти данные записываются в системный журнал (подобие текстового файла).

Также, отдельно модулем «Syncthing» записываются все события (соединения и изменения в данных), которые им выполняются.

Серверная часть с определенной периодичностью рассылает на клиентские части команду — «отправить журнал и статистику». Клиентская часть при получении данной команды отправляет накопленную информацию со времени предыдущей команды.

Администратор системы может принудительно запросить актуальную статистику и журнал выполнив определенную команду (раздел 3.1.5 пояснительной записки).

### 3.1.7 Отправка уведомлений

При возникновении запросов и событий которые требуют внимания (команды) администратора системы (например обнаружение и попытка авторизации неизвестного участника обмена) отправляются на указанный в настройках электронный почтовый ящик по протоколу SMTP (Simple Mail Transfer Protocol).

Для поддержки отправки уведомлений на электронную почту для серверной части должен быть создан персональный электронный ящик, указан IP-адрес сервера SMTP, порт, логин и пароль.

## 3.2 Клиентская часть программного комплекса

Клиентская часть разделена на два модуля – системный и

пользовательский. Каждый из них описан в соответствующих разделах.

### 3.2.1 Системный модуль клиентской части

Системный модуль клиентской части работает как служба и обеспечивает:

– получение уникального идентификатора от «Syncthing» и отправка его на серверную часть для авторизации;

– прием от серверной части списка участников обмена (списка идентификаторов) и передача его в «Syncthing»;

– изучение списка пользователей рабочей станции и добавление директорий с их данными в «Syncthing» для репликации (для операционных систем семейства «Windows NT» – это ветвь реестра «HKEY_LOCAL_MACHINE \ SOFTWARE \ Microsoft \ Windows NT \ CurrentVersion \ ProfileList». Для «GNU/Linux» требуется пакет «sssd», настроенный для взаимодействия с доменом «Active Directory»);

– прием от серверной части списка поддиректорий (включений, исключений) для репликации. Дополнение списка с учетом текущих пользователей рабочей станции и передача его в «Syncthing»;

– переход в автономный режим при обнаружении VPN подключения или при получении команды с серверной части. Репликация будет проводиться только «по требованию» от пользовательского модуля и только для данных пользователя, который запросил репликацию (необходимо в условиях лимитированного подключения к корпоративной сети);

– обработка запросов от пользовательского модуля клиентской части;

– обновление «Syncthing» до актуальной версии.

Блок-схема алгоритма системного модуля клиентской части представлена на рисунке 3.2.1.1.

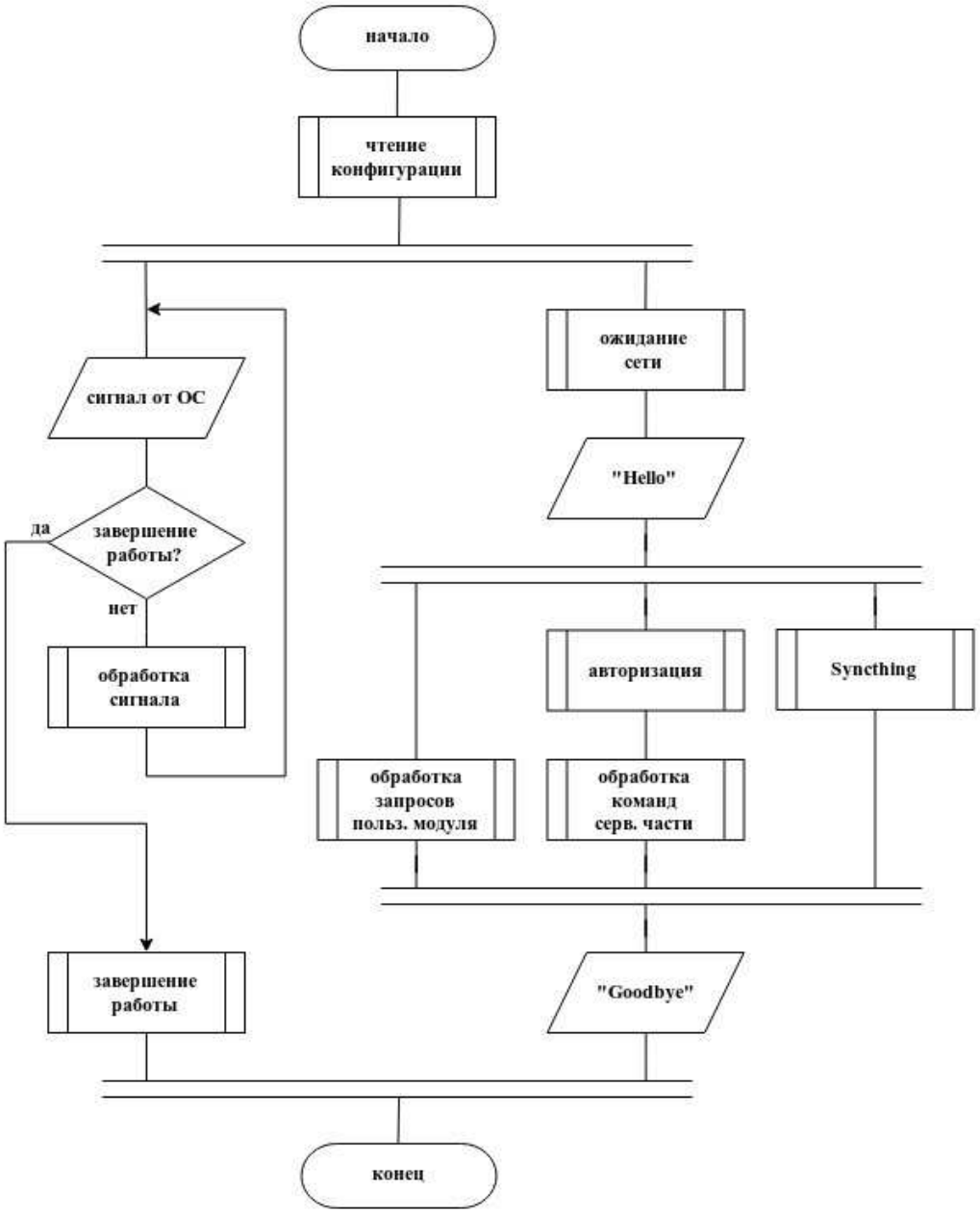


Рисунок 3.2.1.1 — Блок-схема алгоритма системного модуля клиентской части

Она схожа с блок-схемой серверной части (рис. 3.2.1.1), за исключением наличия авторизации рабочей станции на серверной части. Причем только для получения команд от серверной части. Для взаимодействия с пользовательской частью и запуска «Syncthing» авторизация не требуется (для возможности репликации данных, когда серверная часть недоступна).

### 3.2.2 Пользовательский модуль клиентской части

Пользовательский модуль работает как подпрограмма, запущенная от имени активного пользователя рабочей станции и обеспечивает:

– отображение текущего состояния данных (прием, передача, список последних событий) через уведомления;

– добавление поддиректорий пользователя (включение, исключение) для репликации путем подачи запроса системному модулю клиентской части. Список поддиректорий заданный на серверной части имеет приоритет;

– вызов репликации «по требованию» путем подачи запроса системному модулю клиентской части;

– включение «последовательных версий файлов» для критичных, по мнению пользователя, данных. Системному модулю передается запрос на включение данной функции с расположением файлов относительно корневой директории профиля пользователя. Для операционных систем семейства «Windows NT» –это обычно «C:\Users\[username]», где «[username]» — логин пользователя. Также передаётся количество версий, которые требуется хранить (по последней дате изменения).

Блок-схема алгоритма пользовательского модуля представлена на рисунке 3.2.2.1.

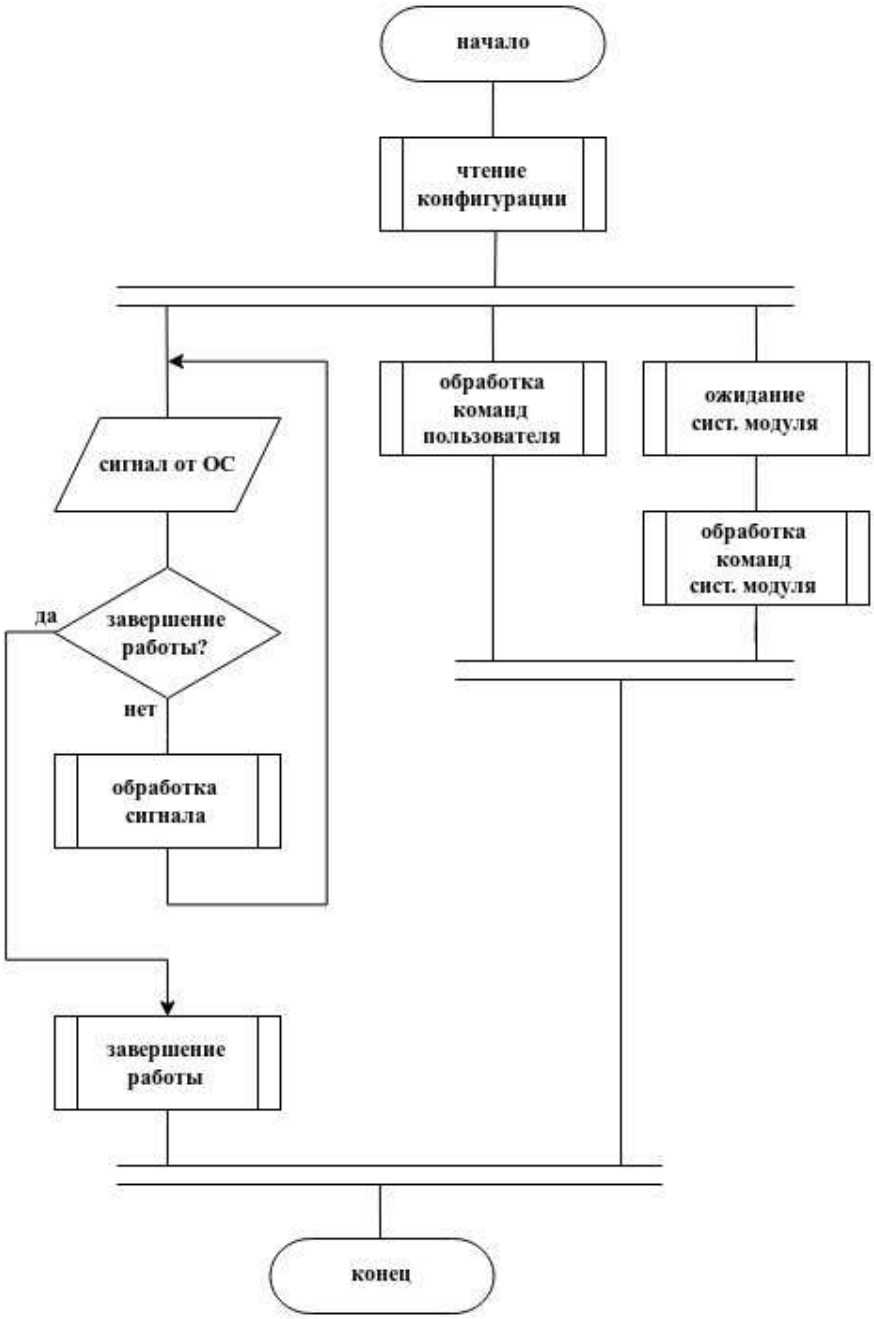


Рисунок 3.2.2.1 — Блок-схема алгоритма пользовательского модуля

### 3.2.3 Входящие запросы от пользовательского модуля

В соответствие с разделом 3.1.2 пояснительной записки, все поступающие запросы от клиентской части добавляются в очередь планировщика и обрабатываются им по порядку поступления. Таким же

способом устроена связь системного модуля клиентской части с пользовательским модулем.

Список доступных запросов и их аргументов представлен в таблице 3.2.3.1.

Таблица 3.2.3.1 — Список доступных запросов и их аргументов

| Запрос | Аргументы | Описание |
|---|---|---|
| exclude | Директория или файл (path) | Исключить директорию или файл из обмена |
| include | Директория или файл (path) | Добавить директорию или файл для обмена |
| set mode | Режим (mode) | Включить/выключить режим автономной работы |
| show config | Нет | Отобразить конфигурацию текущего пользователя (с учетом конфигурации полученной от серверной части для конкретного участника обмена и конкретного пользователя) |
| show status | Нет | Отобразить текущий статут клиентской части для конкретного пользователя (например текущее состояние репликации данных) |
| sync | Нет | В автономном режиме запрашивает разовую синхронизацию данных с другими участниками обмена |

### 3.3 Протокол локального обнаружения

Протокол локального обнаружения (Local Discovery Protocol) служит для автоматического обнаружения всех участников обмена в широковещательном домене (одной подсети). Принцип его работы достаточно прост.

С определенной периодичностью как серверная, так и клиентская части посылают по каналам связи «hello»-сообщения. Внутри «hello»-сообщения содержится идентификатор участника обмена.

Другие участники обмена которым знаком этот идентификатор отвечают обратным сообщением на логический адрес (IP-адрес) с которого было

отправлено изначальное сообщение.

Таким образом участники находят друг-дуга в одной логической подсети.

Так как «hello»-сообщения не шифруются (иначе протокол не будет работать), то чисто теоретически существует вероятность атаки типа — «Discovery Spoofing». Суть этой атаки заключается в том, что можно попробовать подобрать идентификатор, который будет известен другому участнику обмена и попытаться получить его файлы. Но из-за того что идентификатор это всего лишь хэш (контрольная сумма) от сертификата участника, а защищенные соединения устанавливаются именно по сертификатам и закрытым ключам, то подбирать придётся и их.

### 3.4 Описание локальной модели данных

При добавлении директорий для обмена, «Syncthing» создаёт для них репозитории. Репозиторий представляет из себя таблицу содержащую метаданные (путь, размер, время создания, время изменения, занимаемый размер) для всех найденных файлов в директории и её поддиректориях. К примеру, для двух директорий «C:\Users\Пользователь1» и «C:\Users\Пользователь2» будет создано два репозитория – «Пользователь1» и «Пользователь2».

Каждый файл из репозитория разделяется на блоки определенного размера (от 128КБ до 16МБ, в зависимости от размера файла). Контрольные суммы (хэши) этих блоков добавляются к метаданным. К примеру, если блок равен 1Б, то файл в кодировке «UTF-8» с текстом «АБВ» будет разделен на три блока — «А», «Б» и «В».

Объединение всех созданных репозиториев называется — локальной моделью (ЛМ).

Пример локальной модели представлен на рисунке 3.4.1.

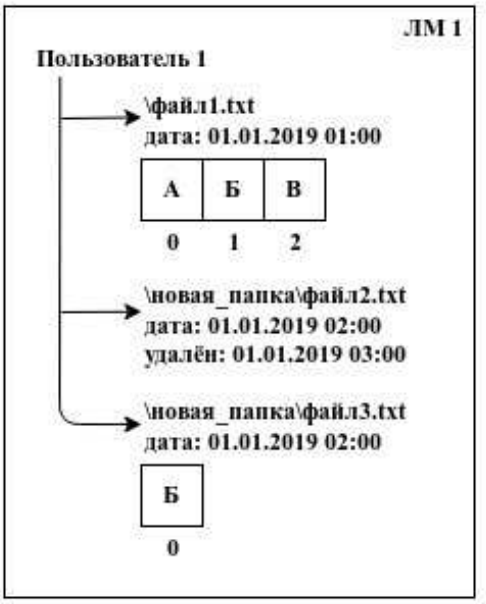


Рисунок 3.4.1 — Пример локальной модели

«Syncthing» отслеживает изменения в директориях и поддиректория и отмечает в локальной модели все произошедшие изменения. Отслеживание изменения может быть периодическим (например каждую минуту) или в реальном времени. Для отслеживания изменений в реальном времени используются подсистемы и функции, предоставляемые операционными системами.

Для «GNU/Linux» — это подсистема «inotify».

Для «Microsoft Windows» — это API (application programming interface) функция «FindFirstChangeNotificationA».

Как видно из локальной модели представленной на рисунке 3.4.1, для файла «\новая_папка\файл2.txt» помимо даты создания, указана и дата удаления.

Локальные модели данных необходимы для формирования глобальной (описана в главе 3.5 пояснительной записки).

### 3.5 Построение глобальной модели данных

После построения и отправки по защищенным каналам связи между всеми узлами в кластере локальных моделей данных, каждый узел объединяет их формируя глобальную модель, включающую в себя последние изменённые версии файлов (по пути и дате изменения).

К примеру, есть три участника обмена с репозиторием «Пользователь 1». Их локальные модели (ЛМ1, ЛМ2 и ЛМ3) представлены на рисунке 3.5.1.

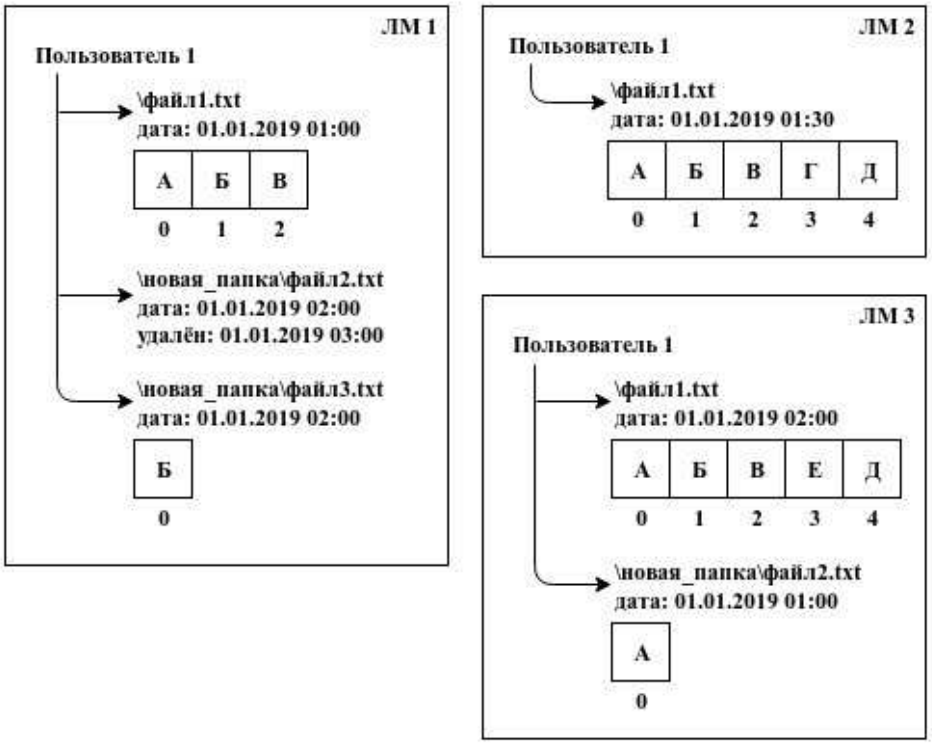


Рисунок 3.5.1 — Локальные модели трех участников обмена

Из локальных моделей (рисунок 3.5.1) видно, что самый «свежий» (по дате изменения) файл «\файл1.txt» находится у третьего участника. Файл «\новая_папка\файл2.txt» есть только у участника один и три, но файл участника один был свежее и был удалён. Файл «\новая_папка\файл3.txt» есть только у участника 1.

С учётом содержания блоков (по контрольной сумме), сформированная участниками обмена глобальная модель будет иметь вид, представленный на рисунке 3.5.2.

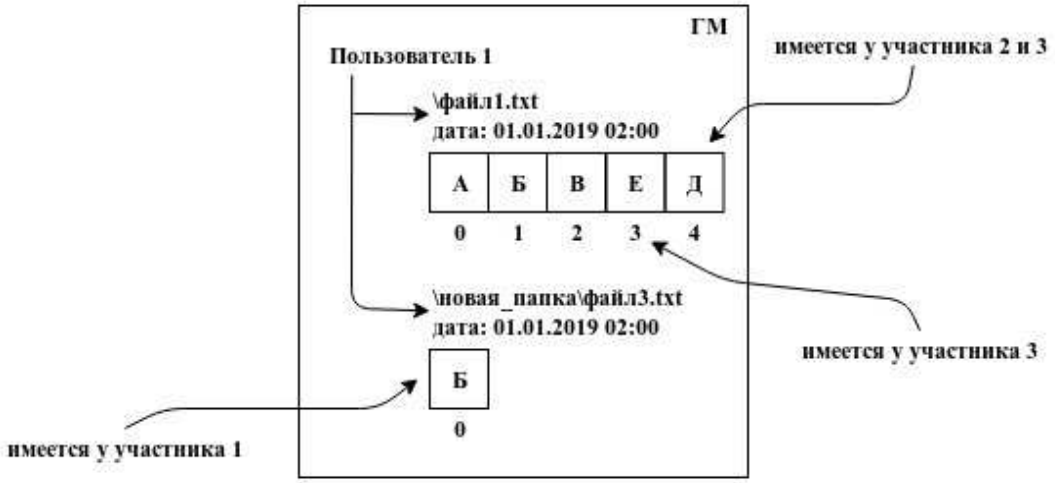


Рисунок 3.5.2 — Глобальная модель

На основе глобальной модели (рисунок 3.5.2) происходит репликация (синхронизация) данных в локальных хранилищах участников обмена. За это отвечает протокол обмена блоками (раздел 3.6 пояснительной записки).

### 3.6 Протокол обмена блоками

Протокол обмена блоками (Block Exchange Protocol), разработанный основателем «Syncthing» Джейкобом Боргом, служит одной цели — привести данные в локальном хранилище данных для конкретной вычислительной машины в соответствие сформированной глобальной модели (раздел 3.5 пояснительной записки).

Каждый узел (участник) будет стремиться синхронизировать локальное хранилище данных с глобальной моделью, запрашивая по защищенным каналам связи у других узлов в кластере отсутствующие или обновившиеся

блоки. Причем узлы будут стараться балансировать нагрузку на канал связи своих соседей.

Для наглядности посмотрим на локальную модель ЛМ1 и сформированную глобальную модель из предыдущего раздела (представлены на рисунке 3.6.1).

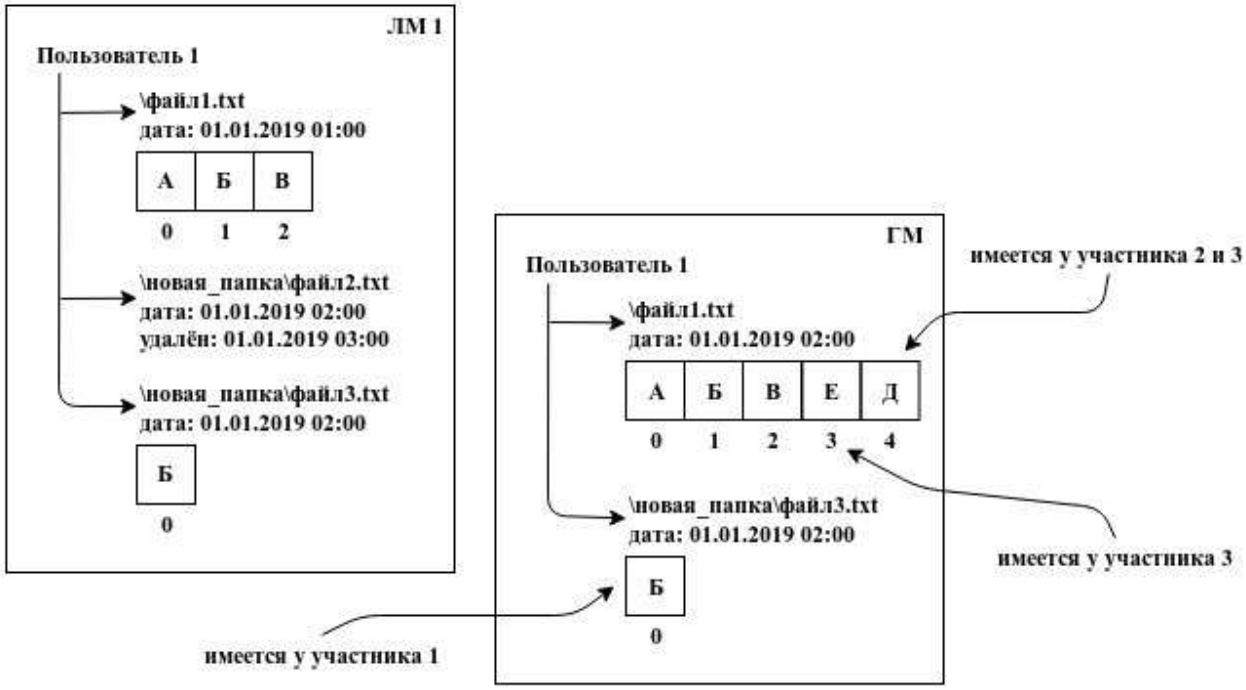


Рисунок 3.6.1 — Локальная модель ЛМ1 и сформированная глобальная модель

К примеру блок №3 для файла «\файл1.txt» есть только у участника №3, а блок №4 есть как у участника №3, так и у участника №2. Поэтому для участника №1 блок №3 файла «\файл1.txt» будет запрошен у участника №3, а блок №4 у участника №2.

Даже если серверная часть программного комплекса выключена или временно недоступна (техническое обслуживание), уже авторизованные клиенты (добавленные в кластер) смогут обмениваться данными между собой по этому протоколу в пределах широковещательного домена (одной подсети)

так как знают идентификаторы друг-друга и самостоятельно обмениваются локальными моделями формируя глобальную. Это удобно в случае работы нескольких рабочих станций за пределами основной корпоративной сети (к примеру, удаленный офис при обрыве связи). Когда серверная часть (или любой другой участник обмена) снова окажется в одной сети, она отправит сообщение «Hello» в пределах широковещательного домена. Все участники заново обменяются локальными моделями, формируя глобальную. На основе глобальной модели серверная часть запросит по этому протоколу все изменения в данных, которые произошли за время отсутствия.

### 3.7 Последовательные версии файлов

Поддержка последовательных версий файлов схожа с «корзиной». Для файлов которые выбрал пользователь или определенной командой задал администратор системы (раздел 3.1.5 пояснительной записки) создаётся «скрытая» поддиректория — «.stversions». В ней хранятся все предыдущие версии указанных файлов, помеченные временной меткой.

Доступны на выбор два алгоритма сохранения последовательных версий файлов — «Simple File Versioning» (простое управление версиями файлов) и «Staggered File Versioning» (версионное управление файлами).

С помощью «Simple File Versioning» файлы перемещаются в папку «.stversions» после изменения (удаления) на локальном или удаленном устройстве. Для этого алгоритма задаётся параметр — «Keep Versions», который сообщает Syncthing, сколько предыдущих версий файла он должен хранить. К примеру возьмем значение равным 5. Если файл будет заменен 5 раз или в конечном итоге удалён, то в поддиректории «.stversions» будет находиться 5 версий с метками времени для этого файла.

С помощью «Staggered File Versioning» файлы также перемещаются в поддиректорию «.stversions» при изменении или удалении (так же, как «Simple

File Versioning»), однако версии автоматически удаляются, если они старше максимального возраста или превышают количество файлов в определенном интервале. Для этого алгоритма задаётся параметр — «Max Age» (максимальный возраст). Используются следующие интервалы: в первый час версия меняется каждые 30 секунд, в первый день — каждый час, первые 30 дней — каждый день, после, до максимального срока — каждую неделю.

Для параметров «Keep Versions» и «Max Age» допустимо значение «0», которое включает хранение всех изменённых версий (без последующего автоматического удаления устаревших версий). Это значение следует использовать с осторожностью, так как при каждом изменении сохраняется копия всего файла, а локальное хранилище данных не резиновое.

### 3.8 Централизованное и децентрализованное хранение данных

Особенность архитектуры программного комплекса в том, что в нём одновременно используется и централизованное и децентрализованное хранение данных.

Вычислительная машина на которой установлена серверная часть собирает все доступные данные, всех пользователей, со всех участников обмена. Таким способом обеспечивается централизованное хранение данных — один участник обмена содержащий актуальные данные пользователя и всегда доступный для обмена.

Централизованное хранилище служит еще одной важной цели. Если вычислительная машина пользователя недоступна (выключена или вышла из строя) — ему чаще всего предоставляется другая (или он может воспользоваться вычислительной машиной отсутствующего коллеги). Данные этого пользователя не появятся в текущей вычислительной машине из неоткуда, они могут быть получены только от участника обмена, у которого они есть. Таким участником обмена выступает вычислительная машина с

установленной серверной частью.

Вычислительные машины, на которые установлена клиентская часть передают или принимают данные только тех пользователей, которые сохранены в операционной системе (список профилей пользователей описан в разделе 3.2.1 пояснительной записки).

Рассмотрим пример — сетевое подключение из четырех вычислительных машин. На одной из них установлена серверная часть (обычно это виртуальный сервер организации), на остальные установлена клиентская часть (вычислительные машины сотрудников). Сетевая топология такого подключения представлена на рисунке 3.8.1.

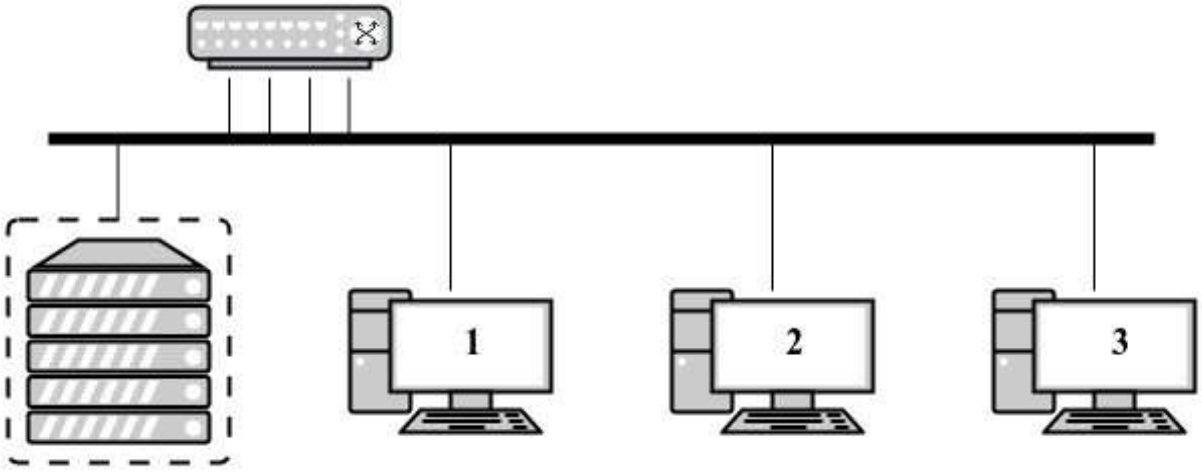


Рисунок 3.8.1 — Сетевая топология

Предположим что на каждой вычислительной машине, на которой установлена клиентская часть (1, 2 и 3) работает по два пользователя. Причем некоторые из этих пользователей иногда работают на разных вычислительных машинах.

Сетевая топология с указанием хранящихся данных представлена на рисунке 3.8.2

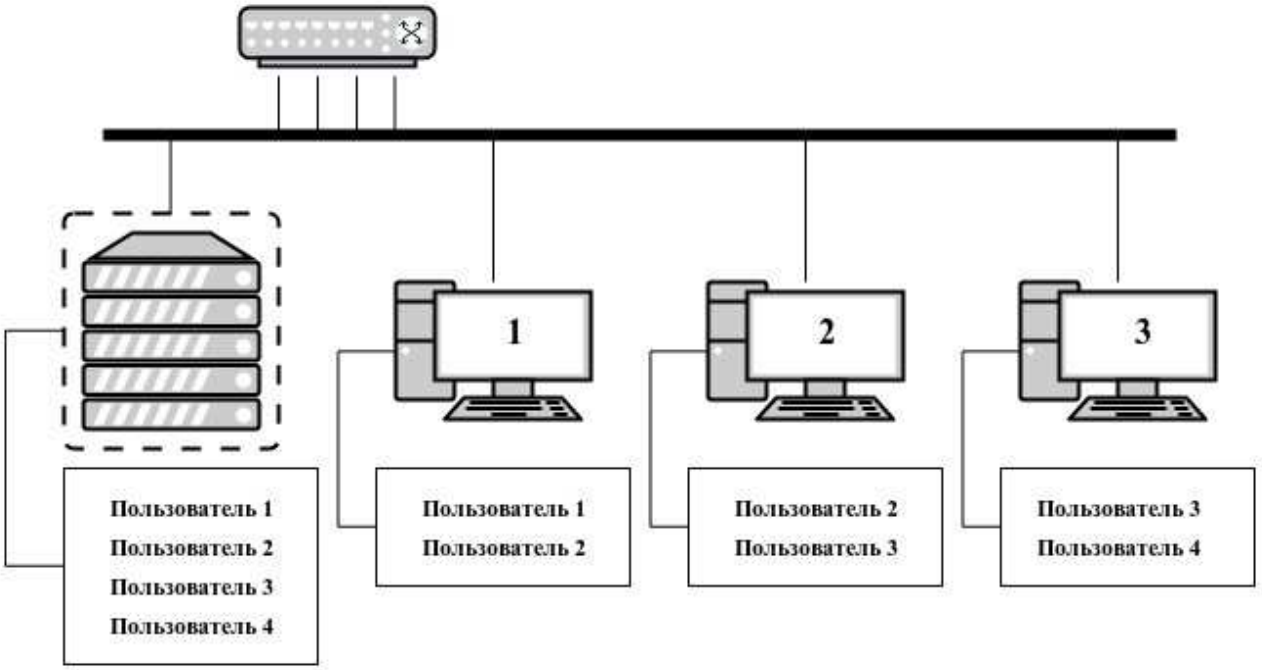


Рисунок 3.8.2 — Сетевая топология с указанием хранящихся данных

На основании вышеописанного и представленной на рисунке 3.8.2 топологии можно сказать следующее:

– каждая вычислительная машина имеет локальное хранилище данных пользователей, которые с ней работают;

– для данных всех пользователей доступно централизованное хранилище (хранение данных на серверной части программного комплекса);

– для пользователей 2 и 3 доступно децентрализованное хранилище (например данные пользователя 2 будут дополнительно храниться на вычислительной машине 1 и 2).

# ЗАКЛЮЧЕНИЕ

В выпускной квалификационной работе были поставлены и выполнены следующие задачи:

– произведен теоретический анализ предметной области, в ходе которого были рассмотрены технологии и программные продукты для репликации пользовательских данных;

– сформулированы требования к проектируемому программному комплексу;

– выбраны средства для разработки программного продукта. В качестве языка программирования был выбран C++, в качестве среды программирования – «Qt Creator». Программный комплекс предназначен для работы в операционных системах семейства «Windows NT», начиная с «Windows 7» (с установленным «Service Pack 1»), по условиям задания;

– разработана структурная схема программного комплекса на основе вышеуказанных требований.

Цель магистерской диссертации достигнута. В результате разработан программный комплекс для блочной репликации пользовательских данных в корпоративных сетях, который позволяет — автоматизировать перенос данных между рабочими станциями и их резервное копирование, что положительно скажется на рабочем процессе.

# СПИСОК СОКРАЩЕНИЙ

AD — active directory;

API — application programming interface (программный интерфейс приложения, интерфейс прикладного программирования);

BEP — block exchange protocol (протокол обмена блоками);

CGI — common gateway interface (общий интерфейс шлюза);

DNS — domain name system (система доменных имён);

HTTP — hypertext transfer protocol (протокол передачи гипертекста);

HTTPS — hypertext transfer protocol secure (протокол передачи гипертекста, защищенный);

IDE — integrated development environment (интегрированная среда разработки);

IIS — internet information services;

IP — internet protocol;

LDP — local discovery protocol (протокол локального обнаружения);

MPL — mozilla public license;

NAT — network address translation (преобразование сетевых адресов);

P2P — point-to-point (соединение типа точка-точка);

PHP — personal home page tools (инструменты для создания персональных веб-страниц);

POSIX — portable operating system interface (переносимый интерфейс операционных систем);

SHA-256 — Secure Hash Algorithm Version 2 (256 bit);

SMTP — Simple Mail Transfer Protocol (простой протокол передачи почты);

SQL — structured query language (язык структурированных запросов);

SSL — secure sockets layer (уровень защищённых сокетов);

TLS — transport layer security (протокол защиты транспортного уровня);

WEB — паутина (интернет-пространство);

WebDAV (DAV) — web distributed authoring and versioning;

WYSIWYG — what you see is what you get (что видишь, то и получаешь);

ГиБ — гибибайт;

ГМ — глобальная модель;

ЕИС — единая информационная система;

ИТ — информационные технологии;

ЛМ — локальная модель;

ОС — операционная система;

УЦ — удостоверяющий центр;

ФЗ — федеральный закон;

ФСБ — федеральная служба безопасности;

ФСТЭК — федеральная служба по техническому и экспортному контролю;

ЭВМ — электронно-вычислительная машина.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Страуструп, Б. Язык программирования C++. Специальное издание : / Б. Страуструп; пер. с англ. С. Анисимова, М. Кононова; под общ. ред. Ф. Анреева, А. Ушакова – Москва : Бином-Пресс, 2004. – 1104 с.

2. Документация Qt [Электронный курс] – Режим доступа: http://doc.qt.io.

3. Документация Syncthing [Электронный курс] – Режим доступа: https://docs.syncthing.net.

4. ГОСТ 19.701-90 (ИСО 5807-85) Единая система программной документации (ЕСПД). Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения. – Взамен ГОСТ 19.002-80, ГОСТ 19.003-80 ; введ. 01.01.1992. – Москва : Стандартинформ, 2010. – 23 с.

5. СТО 4.2-07-2014 Стандарт организации «Общие требования к построению, изложению и оформлению документов учебной деятельности. – Красноярск : ИПК СФУ, 2014. – 60

6. Документация Resilio Sync [Электронный курс] – Режим доступа: https://ru.wikipedia.org/wiki/Resilio_Sync.

7. Документация ownCloud [Электронный курс] – Режим доступа: https://ru.wikipedia.org/wiki/OwnCloud.

8. Документация Яндекс.Диск [Электронный курс] – Режим доступа: https://ru.wikipedia.org/wiki/Яндекс.Диск.

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
« 27 » 06 2019г.

# МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Программный комплекс для блочной репликации пользовательских данных в корпоративных сетях
тема

09.04.01 – «Информатика и вычислительная техника»
код - наименование направления

09.04.01.04 — «Технология разработки программного обеспечения»
код и наименование магистерской программы

| | | | |
|---|---|---|---|
| Руководитель | _____ 26.06.19 | канд. техн. наук, доцент | А.И. Постников |
| | подпись, дата | должность, ученая степень | инициалы, фамилия |
| Выпускник | _____ 25.06.19. | | М.С. Бурименко |
| | подпись, дата | | инициалы, фамилия |
| Рецензент | _____ | канд. техн. наук | Н.Г. Кузьменко |
| | подпись, дата | должность, ученая степень | инициалы, фамилия |
| Нормоконтролер | _____ 26.06.19 | | В.И. Иванов |
| | подпись, дата | | инициалы, фамилия |

Красноярск 2019