DOI 10.37882/2223–2966.2023.01.18

# ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ОБЛАЧНОГО ХРАНИЛИЩА И ПРОЦЕССОВ ОБМЕНА ДАННЫМИ ЧЕРЕЗ НЕГО

## PRACTICAL IMPLEMENTATION OF CLOUD STORAGE AND DATA EXCHANGE PROCESSES THROUGH IT

***D. Kornienko***
***I. Ivannikov***

*Summary.* The article considers the problem of implementation and functioning of cloud data storage, and also develops models and methods for increasing the bandwidth of distributed telecommunication systems of highly accessible cloud data storage based on new access protocols. As a separate aspect, a comparative analysis of the proposed methods with existing ones was carried out and their effectiveness was proved. The system architecture is designed based on the proposed and modeled methods of organizing access to cloud storage. The practical implementation of cloud storage and data exchange processes through it has been carried out.

*Keywords:* personal data, protection, configuration, organization.

***Корниенко Дмитрий Васильевич***
*Кандидат физико-математических наук, доцент, Елецкий государственный университет им. И.А. Бунина*
*dmkornienko@mail.ru*
***Иванников Илья Сергеевич***
*Аспирант, Елецкий государственный университет им. И.А. Бунина*
*ivannikov.work@yandex.ru*

*Аннотация.* В статье рассматривается проблема внедрения и функционирования облачных хранилищ данных, а также разрабатывается модели и методы повышения пропускной способности распределенных телекоммуникационных систем высокодоступных облачных хранилищ данных на основе новых протоколов доступа. В качестве отдельного аспекта проведен сравнительный анализ предложенных методов с существующими и доказана их эффективность. Спроектирована архитектура системы на основе предложенных и смоделированных методов организации доступа к облачному хранилищу. Проведена практическая реализация облачного хранилища и процессов обмена данными через него.

*Ключевые слова:* персональные данные, защита, конфигурация, организация.

Данная статья является продолжением ранее опубликованной статьи о практической реализации облачного хранилища и процессов обмена данными через него. В данной части мы продолжим рассмотрение проблемы внедрения и функционирования облачных хранилищ данных. Информационный обмен между компонентами системы должен осуществляться с использованием локальных вычислительных сетей и глобальных сетей передачи данных [1; 2].

После получения запроса брандмауэром от пользователя, запрос попадает поочередно на доступный сервер приложений, статус доступности которого постоянно прослеживается на первом уровне. Два хранилища после аппаратного подключения, на аппаратном уровне подключаются как единое хранилище. Также на программном уровне настроена синхронизация данных между хранилищами, для обеспечения резервирования данных. Так, чрезвычайно быстро увеличивается быстродействие процессоров и объем оперативной памяти сервера программы без математически сложных алгоритмов не могут использовать полностью и упираются в количество пользователей, целесообразным является использование контейнеров (виртуализация на уровне операционной системы). Тогда, структурно сервер приложения будет состоять из нескольких контейнеров и балансировщика нагрузки (рис. 1).

Для мониторинга доступности сервисов в контейнерах существует несколько программных средств, самым популярным из которых является keepalived. Их недостатком при использовании в нашей архитектуре является отсутствие группировки сервисов. Поскольку одним из ключевых требований является возможность доступа к данным по FTP протоколу различных пользователей разных компаний (каждая компания имеет свой hostname), нам нужно для каждой компании иметь свой IP-адрес доступа (специфика использования FTP). FTP-server имеет возможность поддерживать одновременное использование многих IP-адресов. В таком случае keepalived на каждый IP-адрес будет пробовать создавать соединения и проверять доступность, а если их будет достаточное количество, то нагрузки на сервер для проверки доступности контейнеров будут увеличиваться в геометрической прогрессии. Для решения данной проблемы, спроектирована система, которая полностью заменяет функции keepalived, и имеет возможности:

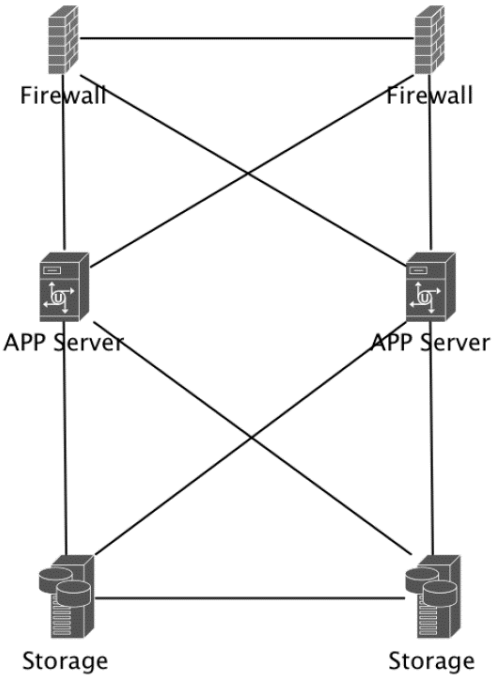


Рис. 1. Общая архитектура единичного хранилища

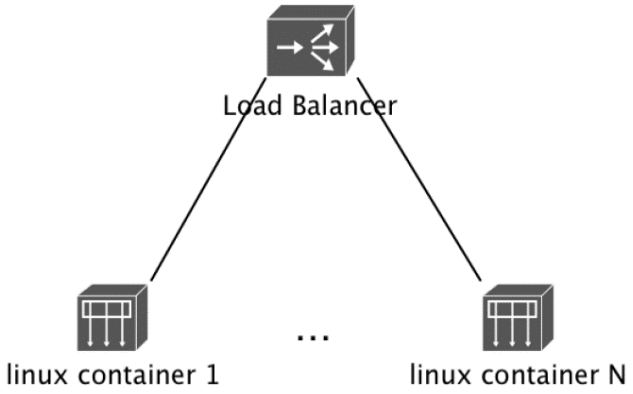


Рис. 2. Общая структура сервера приложения

- группировать IP-адреса и сервисы как один сервис (в случае недоступности, данный сервис перестает быть доступен по всем IP-адресам, которые указаны. И наоборот, в случае повторной доступности сервиса, сервисы по всем IP-адресам становятся снова доступны);
- хранение статистики за всеми IP-адресами и сервисам с учетом: о количестве запросов; о передаваемых данных; о полученных данных; о потере работоспособности сервиса; о восстановлении работоспособности сервиса.

Архитектурно Сателлит соответствует архитектуре сервера приложения из хранилища данных. Ему необходимо большого и надежного хранилища данных, поскольку он выступает исключительно «прокси-сервером» между клиентом и его данным. Также у нас нет необходимости его полностью дублировать, поскольку он не несет в себе критических данных и в случае выхода из строя, автоматически запросы, которые шли на него будут идти на ближние с ним, другие сателлиты.

В работе современного облачного хранилища данных, которое использует информационные системы, должны быть обеспечены как можно более рациональная организация информационный потоков, так и существенное повышение их интенсивности, то есть ускорение передачи и обработки информации, поступающей от ее источника к потребителю. Для решения этих задач при проектировании информационной системы, прежде всего, проводится анализ информационных потоков, к:

- рассмотреть все звенья системы обработки и сохранения данных, начиная с получения исходных сведений, постепенное преобразование и формирование конечных данных, направляемых управляемой системе как команды в качестве отчетной и иной информации. При этом определяется роль каждого элемента системы в облачном хранилище данных, выполняемых системой и зафиксированных в схеме обработки данных, уточняется их структура и функции;
- построить схему информационных связей всех элементов системы между собой и внешней средой. В схеме могут оставаться сведения о конкретных формах информационных связей и указываются их количественные и временные характеристики;
- обнаружить первичные (выходные) для системы данные. Анализ потоков информации — важнейший этап в рационализации существующей системы хранилища данных, который должен обеспечить выполнение целевых задач проектирования.

Изучение потоков информации дает общее представление о функционировании системы и является

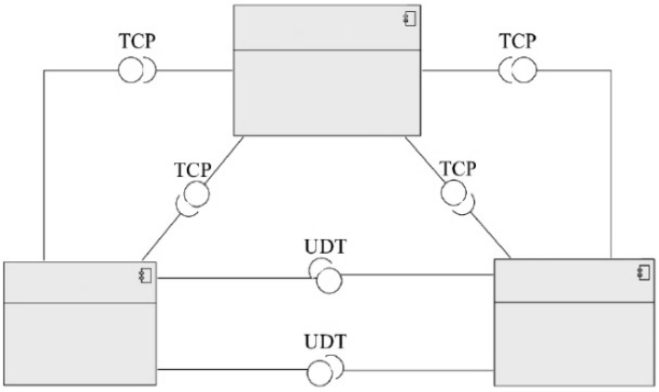


Рис. 3. Диаграмма взаимодействия основных элементов системы

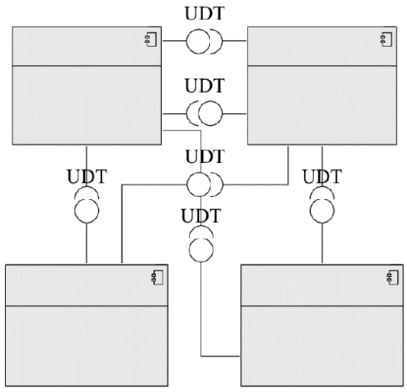


Рис. 4. Диаграмма размещения сателлита

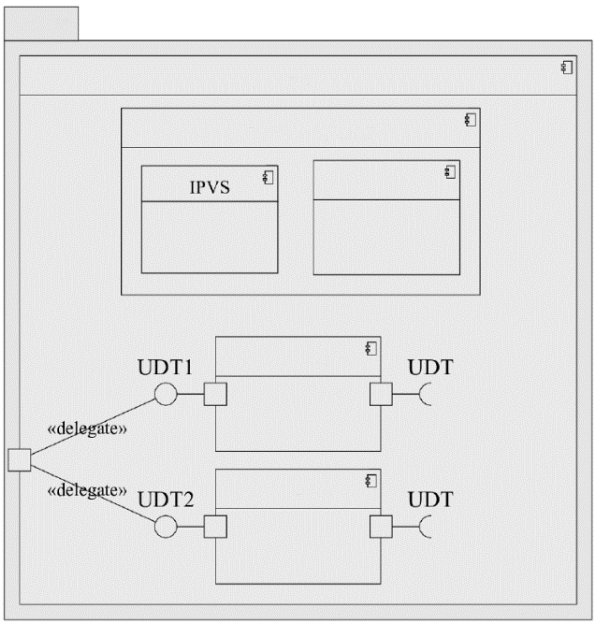


Рис. 5. Диаграмма размещения хранилища

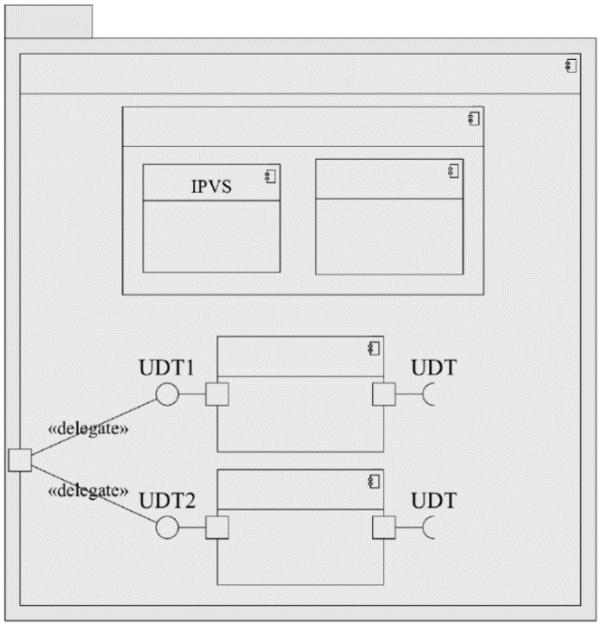


Рис. 6. Диаграмма взаимодействия хранилищ и сателлитов, разработана самостоятельно

первым шагом в анализе эффективного проектирования высоконагруженной системы обработки, хранения и передачи данных. Дальнейшее исследование информационных потоков позволяет выявить элементы информационного отображения объектов, отношения между ними, структуру и динамику потоков информации. Движение данных в системе, сопровождающееся соответствующими информационными потоками, является основой для обеспечения работы облачного хранилища данных. Между всеми элементами системы, функционирующие в среде хранилища данных, происходит непрерывное движение информационных потоков, которые обеспечивают поступления информационных данных, необходимых для осуществления анализа и передачи клиентских данных. Основными элементами системы являются (рис. 2): хранилище данных — один из главных элементов системы, на котором физически и долгосрочно хранятся клиентские данные (рис. 3); сателлит доступа к данным — элемент доступа к данным в системе, который находится как можно ближе» к пользователям (потребителей данных), и имеет возможность временного хранения (кэширования) (рис. 4); DNS-сервер — элемент системы, принимающий динамические обновления от других элементов системы, и предоставляет достоверную и актуальную информацию пользователям данных.


Рис. 7. Диаграмма размещения
Контейнера хранилища

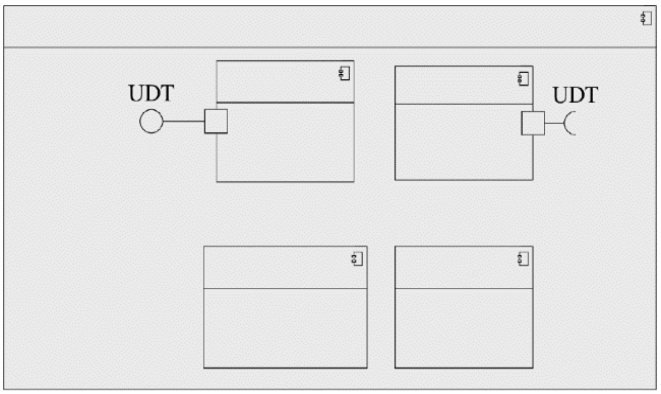


Рис. 8. Диаграмма размещения
Контейнера сателлит

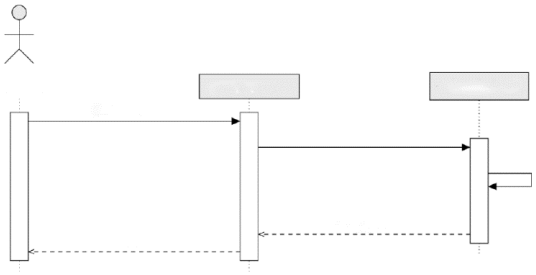

Рис. 9. Диаграмма последовательности
поиска ближайшего сателлита

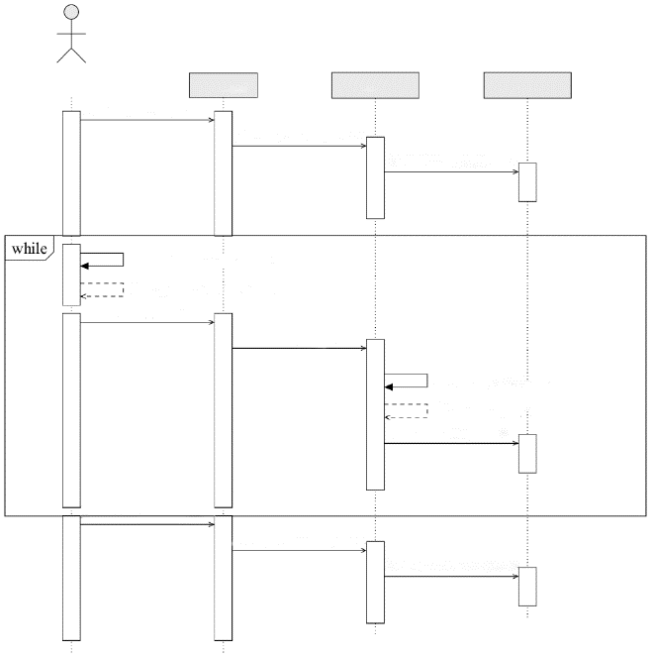


Рис. 10. Диаграмма последовательности
загрузки данных

Для актуализации информации о местонахождении доступных элементов системы и их доступности, используется DNS-сервер. Он выступает ключевым элементом информационного взаимодействия других элементов системы. Для этого каждый элемент системы с определенной периодичностью присылает информацию о себе и авторизуется на DNS-сервере используя TCP соединение (рис. 2). В свою очередь, другие элементы системы обмениваются информационными данными с помощью протокола данных, который основывается на UDT (рис. 5). Хранилища данных, как элемент системы управления данными активно потребляет информационные ресурсы и, соответственно, является получателем и отправителем информационных потоков. Сателлит как элемент потребления данным также активно потребляет информационные ресурсы в двустороннем направлении. Для детального анализа потоков данных, стоит рассмотреть каждый элемент системы в разрезе данных и подсистем данного элемента.

Для максимального использования ресурсов системы, каждый физический сервер поделятся на максимально независимые блоки (контейнеры). Контейнеры — это система виртуализации на уровне операционной системы для запуска нескольких изолированных экземпляров ОС (operating system) Linux на одном компьютере. Они не используют виртуальные машины, а создают виртуальное окружение с собственным пространством процессов и сетевым стеком. Все экземпляры контейнеров используют один экземпляр ядра ОС, для максимальной эффективности их использования. Для обеспечения работы системы из нескольких контейнеров, в качестве одной системы использовано два типа (рис. 3, 4): контейнер «Балансер нагрузки»; контейнер приложения.

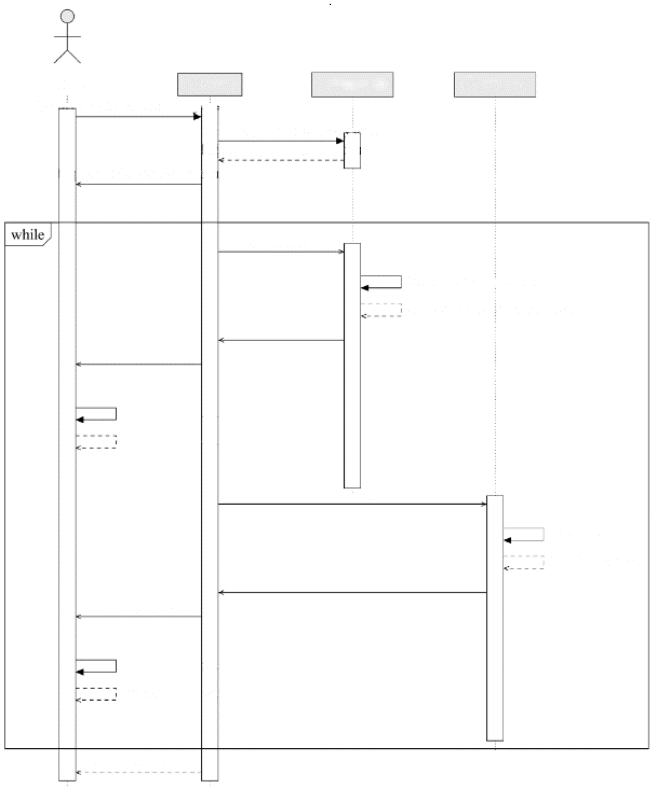


Рис. 11. Диаграмма последовательности доступа к данным

Балансер нагрузки выполняет роль прокси-сервера, который проверяет доступность контейнеров и переадресует полученные запросы в доступный контейнер для их обработки. Контейнер приложения — это полноценная копия программного приложения который обрабатывает запрос пользователя и возвращает ему результат работы. Контейнер хранилища данных (рис. 6) получает и выполняет запросы исключительно по протоколу UDT для чего в контейнере запущены независимые процессы UDT-сервер и UDT-клиент. С помощью данного протокола контейнер получает данные с других хранилищ даны или от клиентов через сателлиты, и отсылает данные другим хранилищам данных и клиентам с использованием сателлитов доступа к данным.

Контейнер приложения сателлита (рис. 7) содержит в себе:

- UDT-сервер и UDT-клиент-для обмена данными с другими элементами глобальной системы хранилища данных.
- Web-сервер для простого доступа к данным для пользователей.
- Proftpd-сервер — предназначен для расширения функционала доступа к данным с помощью протоколов: FTP, FTPS, SFTP.

Рассмотрев функционал системы как отдельные бизнес-процессы системы, мы получим: Поиск ближайшего сателлита доступа к данным. Загрузка данных на хранилище. Доступ к данным. Репликация данных.

Поиск ближайшего сателлита доступа к данным изображен на рис. 8.

В случае, когда клиенту необходимо получить доступ к данным, программное приложение обращается к DNS-клиенту операционной системы с запросом получить IP-адрес системы используя доменное имя. В свою очередь DNS-клиент используя сеть DNS-серверов обращается к NS-серверам системы и он, используя IP-адресов клиента вычисляет оптимальный сателлит доступа к данным, который «ближайший» (минимальное количество шагов и максимальная скорость) к нему.

Загрузка данных на хранилище происходит согласно рис. 9. Получив IP адресов ближайшего сателлита, клиента авторизуется на сервисе и только после того, может получить доступ к данным. Когда клиенту необходимо загрузить документ, он с помощью программного приложения отправляет документ на сателлит с использование протокола TCP. Сателлит в свою очередь кэширует данные и отправляет данные на храни-

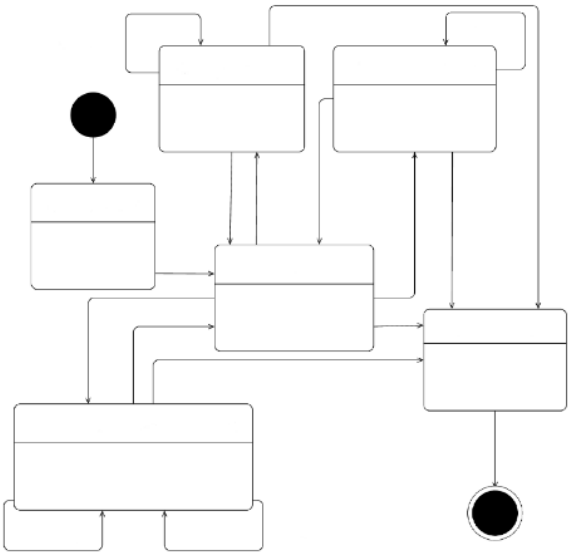

Рис. 12. Диаграмма состояний канала передачи данных

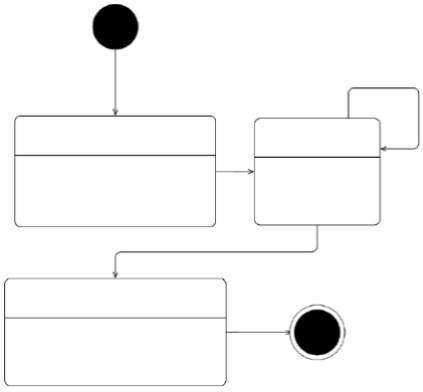

Рис. 13. Диаграмма состояний файла для передачи

лище получения напрямую, или через ближайшее хранилище с использованием протокола UDT.

Доступ к данным демонстрирует рис. 10. В случае необходимости получения документа клиент с помощью программного приложения отправляет запрос на получение документа к сателлиту с использование протокола TCP. Сателлит в свою очередь получает документ из хранилища напрямую или через другое хранилище с использованием протокола UDT. Документ временно кэшируется на сателлите на присылается клиенту (TCP).

В концепции глобальной сети существуют два подхода к реализации обмена данными: последовательная передача данных — передача данных последовательно по одному каналу связи; параллельная передача данных — передача данных параллельно по нескольким каналам связи. Преимуществом параллельной передачи данных является скорость передачи данных. Данный метод используется в компьютерной периферии для обмена данными в шинах данных. Основным недостатком данного подхода является зависимость от качества и проводимости проводников, используемых в данной передаче. При различных свойствах проводников биты в передаче данных могут приходить с задержкой, что приводит к значительным ошибкам.

Методы передачи данных также классифицируются на: синхронная передача данных; асинхронная передача данных. Асинхронный обмен является наиболее распространенной формой последовательного связи, что предполагает передача пакета данных в котором содержится информация о начале и конце передачи данных, информация для контроля ошибок и сами данные. Поскольку архитектура облачного хранилища данных предполагает как одновременную передачу данных в разных направлениях, так и прием данных из разных источников данных, была выбрана асинхронная последовательная передача данных. Также для эффективного распределения ресурсов был выбран асинхронный подход к проектированию системы.

Модуль системы передачи данных можно представить как систему состояний и переходов (рис. 11). При запуске процесса создается канал передачи данных, который переходит в статус «ожидания на события». Поскольку данный канал связи предусматривает, как передачу данных, так и прием данных, данный канал связи реагирует на следующие события:

Входное соединение — во время данного события системе необходимо аутентифицировать клиента и добавить данное соединение в пул соединений.

Входные данные — получить данные, расшифровать и выполнить действия, предусмотренные в данном пакете данных. В случае отсутствия данных у канала связи для чтения, и недополученные пакеты данных, данный пакет помещается в очередь недополученных пакетов данных, и система ожидает возможность получения данных по данному каналу связи.

Необходимость в передаче данных — проверка на существование соединения с удаленным клиентом, формирование пакетов данных, шифрования их и передачу их по каналу связи. В случае невозможности

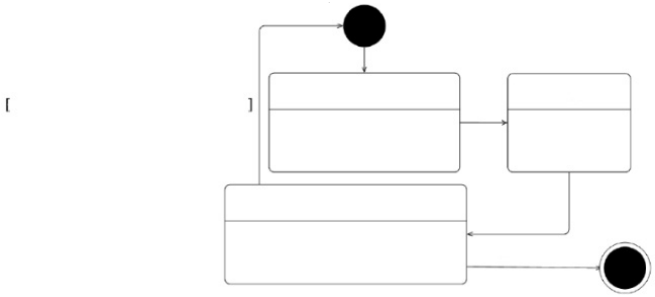

Рис. 14. Диаграмма состояний передачи пакета данных

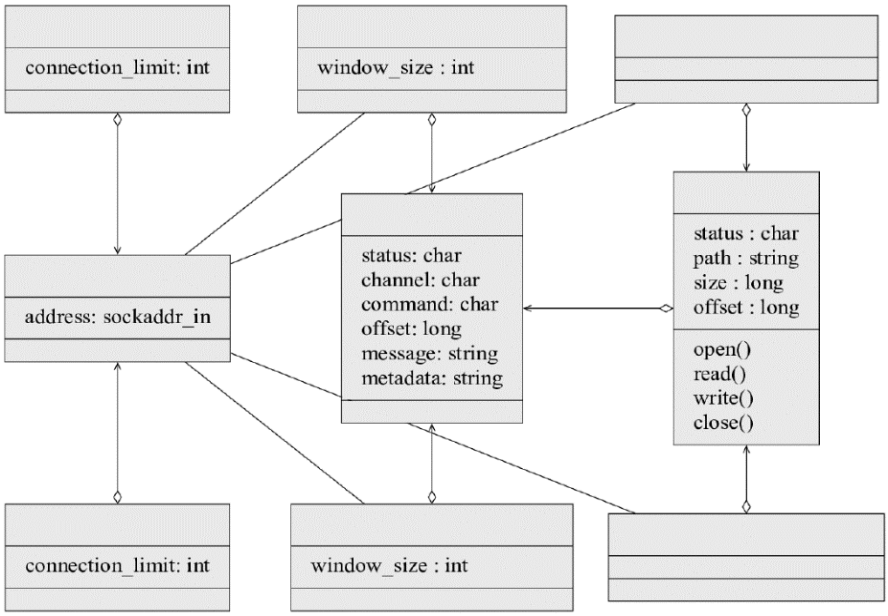


Рис. 15. Диаграмма классов системы

немедленно передать данные, пакеты данных перемещают в очередь пакетов и ожидают возможности передачи (освобождения выходного канала связи).

Ошибка соединения — в данном случае система оценивает данную ошибку, и принимает решение. Если система получила ложный пакет, или неожиданный для нее, система «просит» клиента повторно передать данные. Если произошел более крупный сбой системы, как сбой синхронизации шифрования данных, система разрывает соединение с данным клиентом, после чего клиент пересоздает соединение и повторно отправляет данные, на которые не получил подтверждения от сервера.

Поскольку клиентские данные хранят в облачном хранилище данных в виде файловой структуры, передача данных в системе сводится к передаче файлов. Передачу отдельного файла можно представить как передачу отдельных пакетов данных (рис. 12).

Во время передачи любого количества данных система формирует пакеты передачи данных и добавляет их в очередь передачи данных. После чего система выполняет ряд функции: проверка на существование соединения с пунктом назначения данных. Если соединение отсутствует: создание соединения с пунктом назначения; аутентификация на пункте назначения. Поочередное шифрования па передача пакетов данных (из очереди) по данному каналу связи.

В случае заполнения выходного кэша система переходит в ожидание других событий.

После удачной отправки пакета данных система замечает в очереди передачи данных пакет, как отправленный. После удачного получения клиентом пакета данных, он отправляет пакет подтверждения. Серверная часть, получив данный пакет удаляет пакет из очереди передачи данных. В случае разрыва соединения

с получателем данных, система автоматически меняет статус отправленных пакетов данных, на новые, для повторной передачи данных (рис. 13). Логическую модель системы передачи данных в облачном хранилище данных можно представить в виде диаграммы классов (рис. 14). Из диаграммы классов очевидно, что концептуальная модель системы передачи данных в хранилище данных и сателлите аналогичны. Данный подход упрощает построение систем такого характера. Также из данной структуры очевидно, что «Соединение» является абстрактным классом и может использоваться протоколами, как TCP или UDT, для транспортировки данных. Такой подход дает существенные преимущества в случаях существования единого ограниченного канала связи, что является весьма актуальным для современного состояния связи в 1С системах.

При необходимости передачи данных (файла) система выполняет ряд простых действий: 1) Создание экземпляра класса «Файл». 2) Добавление его к «Очереди Исходных Файлов». 3) Получение ссылки на соединения (в случае, если отсутствует — создать). 4) Создание экземпляров класса «Пакет» в рамках свободного размера окна для соединения. 5) В случае готовности соединения для предо — получение «Пакета» из «Очереди Исходящих Пакетов», шифрование и передача по каналу связи. 6) Получение подтверждения окончания файла. 7) Удаление экземпляра класса «Файл».

С другой стороны, получатель данных (файла), выполняет следующие действия: 1) Получение пакета данных с данными о данных (файл) и его расшифровка. 2) Создание экземпляра «Файл». 3) Добавление его к «Очереди Входной Файлов». 4) Создания физического файла. 5) Получение пакетов с данными, расшифровка и наполнение ими физический файл, отправки подтверждения получения пакетов данных. 6) Получение пакета завершения файла. 7) Закрытие файла. 8) Удаление экземпляра класса «Файл». 9) Отправка подтверждения получения полного файла.

Процесс получения пакета данных можно разбить на два этапа: 1) получение заголовка пакета; 2) получение основного тела пакета.

В заголовке сообщения содержится информация о: канале связи — для мультиплексной передачи данных нескольких файлов в один момент времени; размере тела пакета данных; команде — тип данных, которые передаются в теле пакета (создание каталога, файла, данные из файла, подтверждение получения пакета…); также может содержаться дополнительная информация в зависимости от команды.

**Выводы.** Новый протокол обеспечивает эффективное использование ресурсов крайних узлов сети. В вопросе защищенности он поддерживает шифрование данных, то есть он способен разрешать подключение протоколов шифрования. Была проведена оптимизация эффективности реализации протокола на базе UDP и было показано, что по этим идеям можно реализовать эффективные и практические приложения на базе протоколов UDP. Например, использование среды протокола UDT (основанный на UDP), может легко поддерживать различные алгоритмы управления перегрузкой, например, высокоскоростные TCP или взрывное RBUDP.

Использование данного подхода позволило увеличить производительность как элемента передачи данных, так и системы в целом. Использование многоканальной связи позволило одновременной передачи данных по одному каналу связи, а универсализация архитектуры позволило использование различных протоколов обмена данных на различных участках сети, в зависимости от эффективности.

ЛИТЕРАТУРА

1. Maximov, R.V., Ivanov, I.I., Sharifullin, S.R. (2017). Network topology masking in distributed information systems. CEUR Workshop Proceedings, 2081, 83–87.
2. Kornienko, D.V. (2020). Organization of a system of digital education practices in the municipal sphere of general education. Journal of Physics: Conference Series, 1691(1), article number 012108.


Журнал «Современная наука: актуальные проблемы теории и практики»