

УДК 004.75

Архитектура распределенной автоматизированной системы потокowego вещания видео

Щербаков Д. С., студент

*Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»*

Научный руководитель: Гапанюк Ю.Е., к.т.н, доцент

*Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»*

gapyu@bmstu.ru

Введение

В крупных населенных пунктах часто возникают разного рода проблемные ситуации, например, дорожно-транспортные происшествия, конфликты на дороге, неправомерные действия граждан или сотрудников полиции, чрезвычайные происшествия, массовые мероприятия и т.п. В связи с миниатюризацией вычислительных устройств и наличием в них компонентов, записывающих звук и изображения, а также беспроводных интерфейсов ближнего и дальнего радиуса действия появилась возможность использовать их в упомянутых ситуациях, в том числе, для обеспечения доказательной базы для конфликтующих сторон, оперативного информирования в чрезвычайных ситуациях, контроля за ходом различного рода мероприятий. Немаловажно и то, что разработанные методы кодирования и передачи информации позволяют использовать существующую инфраструктуру операторов мобильной связи для этих целей. В связи с этим, разработка комплексной системы, предназначенной для использования в приведенных ситуациях является актуальной.

Требования к архитектуре системы

Архитектура проектируемой системы должна предполагать использование устройств пользовательского класса (смартфоны, переносные камеры), баз данных, клиентского ПО, сетевой инфраструктуры провайдеров и организации для записи видео и его потокового вещания на серверы организации в режиме мягкого реального времени для последующего хранения и, при необходимости, одновременного вещания с серверов на автоматизированные рабочие места операторов. Под мягким реальным временем

понимается наличие задержки, большей 300 мс, в контексте оценки качества в стандарте [1].

Предполагается, что система будет использоваться в крупных населенных пунктах, возможно значительное количество устройств, одновременно осуществляющих вещание, следовательно, должны быть предусмотрены механизмы отработки отказа в случае выхода из строя одного или нескольких серверов, необходимости осуществления ремонта и замены серверного и сетевого оборудования. При этом желательно минимизировать потери переданных с устройства данных, поскольку невозможно заранее предсказать их важность. Также необходимо предусмотреть возможность масштабирования системы для увеличения количества одновременно вещающих устройств и упрощения виртуализации серверной части системы для размещения в центрах обработки данных.

Архитектура системы должна предусматривать автоматизированные рабочие места операторов, осуществляющих просмотр записанных и передаваемых с камер устройств видеоданных, с которых, однако, невозможно удаление уже записанных данных для исключения несанкционированного удаления «неудобных» записей в личных целях. Также должна быть реализована функция контроля камеры конкретного устройства: возможность включать и выключать запись на серверное хранилище и на постоянную память самого устройства, просматривать текущий видеопоток. При этом, исключается возможность выключать камеру устройства в случае, если запись на сервер запрошена с устройства также для избегания ситуаций преднамеренного или непреднамеренного отключения камеры в момент записи важного события.

На операторов в системе также возлагается задача классификации ситуаций, происходящих на видеозаписях для обеспечения каталогизации и более удобного поиска записей по ключевым словам. Для этого в архитектуре системы должен быть предусмотрен модуль, реализующий экспертную систему, позволяющую по ключевым вопросам классифицировать ситуацию, запечатленную на видеозаписи, и присваивающий необходимый набор тегов. Кроме экспертной системы, возможно использование различных алгоритмов машинного обучения, в том числе многослойных нейросетей.

Автоматизированные рабочие места администраторов должны осуществлять добавление, редактирование и удаления записей из базы данных, в том числе, и видеозаписей. Предполагается, что администраторов будет ограниченное количество, чтобы избежать несанкционированного удаления видеозаписей.

Архитектура системы

Предложенная архитектура представлена на рис. 1.

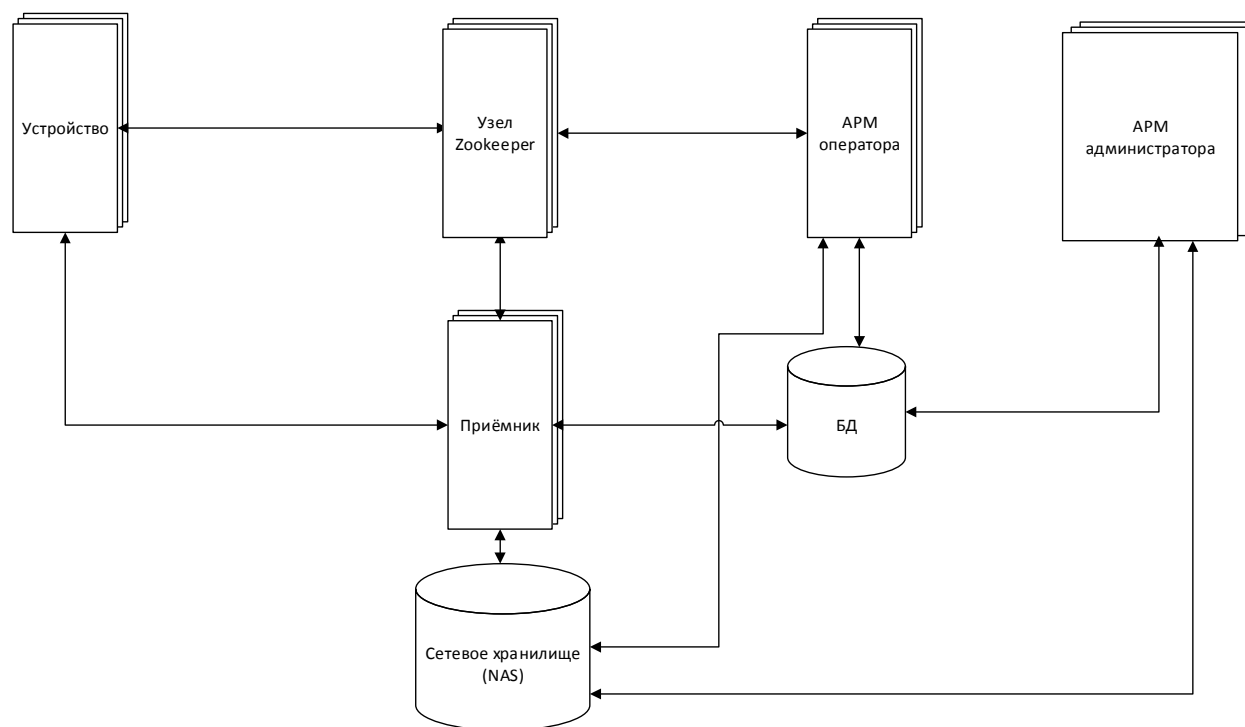


Рис. 1. Архитектура распределенной автоматизированной системы потокового вещания видео

Ниже представлено краткое описание составных частей:

- 1 Устройство. Устройство обладает необходимыми компонентами для осуществления записи изображения и звука и передачи данных по мобильным сетям третьего и четвертого поколения (UMTS, LTE, LTE-A);
- 2 Узел Zookeeper. Хранит распределенную между другими узлами память о состояниях устройств и приёмников. При наличии $2f + 1$ узлов, f узлов могут выйти из строя без потери распределенного состояния (согласно документации проекта Apache Zookeeper [3]);
- 3 Приёмник. Осуществляет запись поступающих потоков данных с устройств на сетевое хранилище и, по необходимости, вещание этих же данных клиентам, которые запросили поток с конкретных устройств;
- 4 АРМ оператора. Клиентское приложение, обладающее возможностью получения записей из базы данных, просмотра информации о текущем статусе устройства, просмотра потока видео с различных устройств и записей видео. Также используется для классификации ситуаций, происходящих на видеозаписях за счёт встроенного механизма работы с базой знаний;

- 5 АРМ администратора. Клиентское приложение, обладающее возможностью редактирования данных в БД, также всеми возможностями АРМ оператора;
- 6 База данных. БД хранит информацию об организации, сотрудниках, устройствах, свойствах устройств, автомобилях, записях. Также содержит таблицы, необходимые для использования в качестве базы знаний о ситуациях на видеозаписях;
- 7 Сетевое хранилище. Служит для хранения видеозаписей;

Связь между устройствами и узлами Zookeeper осуществляется через мобильную сеть провайдера (построенную по стандартам UMTS, LTE или LTE-A), соединенную по магистральной сети с вычислительным центром, в котором развернуто серверное оборудование и вышеупомянутое ПО. Соединение между приёмником и сетевым хранилищем желательно организовывать посредством специальных технологий и протоколов, применяемых в крупных вычислительных центрах, таких как FibreChannel. Связи с автоматизированными рабочими местами и базами данных можно осуществлять, как через локальную сеть, так и посредством соединения географически распределенных локальных сетей организации посредством виртуальных частных сетей (VPN). Выбор таких связей лежит на инженерах, разворачивающих систему.

В качестве библиотеки для работы с мультимедиа используется GStreamer [4]. Эта библиотека предоставляет унифицированный программный интерфейс для работы с драйверами аудио- и видеоустройств, кодеками различных стандартов сжатия аудио и видео, работу по синхронизации мультимедиа потоков и обработку событий, сетевыми протоколами такими как RTP, RTCP, RTSP и другие возможности.

Поскольку требуется хранить информацию о подключенных устройствах и их статусах в распределенном виде используется система из $2f + 1$ узлов Apache Zookeeper, в которых происходит хранение переменных состояния.

Для обеспечения безопасности передачи данных предполагается использование виртуальных частных сетей (OpenVPN, IPSec или других, в зависимости от используемых в организации). Такое решение вносит дополнительную задержку, которую можно частично уменьшить за счёт использования сетевого оборудования с аппаратным ускорением шифрования как на стороне вычислительного центра, так и на стороне записывающего устройства.

Координация действий различных компонентов распределенной системы

Поскольку в системе участвует множество отдельных составных частей, работающих асинхронно по определенным алгоритмам и подверженных различного рода отказам возникает задача консенсуса в системе ненадежных вычислителей. Разработано

множество различных алгоритмов, решающих данную задачу, а проблемы, возникающие при реализации распределенных систем подробно описаны в статьях Лампорта (см., например, [2]).

Одной из систем, реализующих нужные в разрабатываемой автоматизированной системе возможности, является Apache Zookeeper [3]. Zookeeper был спроектирован и реализован в компании Yahoo, а затем передан организации Apache. Apache Zookeeper позволяет крупным распределенным приложениям осуществлять действия по координации, такие как выбор лидирующего узла, передачу статусов и синхронизацию. Моделью данных является древовидная иерархия узлов с данными, называемых znode, которые клиенты Zookeeper используют для координации действий. Zookeeper не использует семафоры, а вместо них применяются совместно используемые объекты с доступом без ожидания и строгими гарантиями определенного порядка следования операций над этими объектами. С учетом этих гарантий клиентские библиотеки и реализуют действия по координации. Одним из главных предположений, принимаемых во внимание при разработке Zookeeper, являлось то, что порядок следования модификаций более важен, чем синхронизация за счёт блокировки.

Частью Apache Zookeeper является широковещательный протокол Zab (Zookeeper Atomic Broadcast), основным свойством которого является линейная упорядоченность операций. Линейная упорядоченность, согласно создателям Zookeeper, является ключевой особенностью при реализации гарантий для клиентских приложений. При этом, на каждом узле Zookeeper хранится реплика состояния системы. Эти реплики остаются согласованными, за счёт использования протокола Zab.

Обычно размещение системы составляет 3 – 7 серверов, однако реализация Apache Zookeeper позволяет использовать и большее количество серверов. Клиент системы присоединяется к одному из серверов, всегда предоставляющему согласованное представление состояния сервиса Zookeeper (а именно, иерархии узлов с данными). При этом, при наличии $2f + 1$ серверов, f из них могут выйти из строя без потери работоспособности сервиса.

Zookeeper разработан принимая во внимание необходимость одновременного подключения десятков тысяч клиентов, следовательно, к реализации предъявлялись строгие требования по пропускной способности. При разработке предполагалось использование системы при соотношении операций чтения к операциям записи большем, чем 2:1. За счёт того, что чтение производится из реплики состояния, находящейся на конкретном сервере, скорость обработки запроса на считывание данных из узла производится быстрее, нежели обработка запроса на запись. Таким образом,

отказоустойчивость и пропускная способность по чтению масштабируются с увеличением количества серверов. Однако, пропускная способность по записи данных не масштабируется при добавлении большего количества серверов: она ограничена пропускной способностью широковещательного протокола Zab.

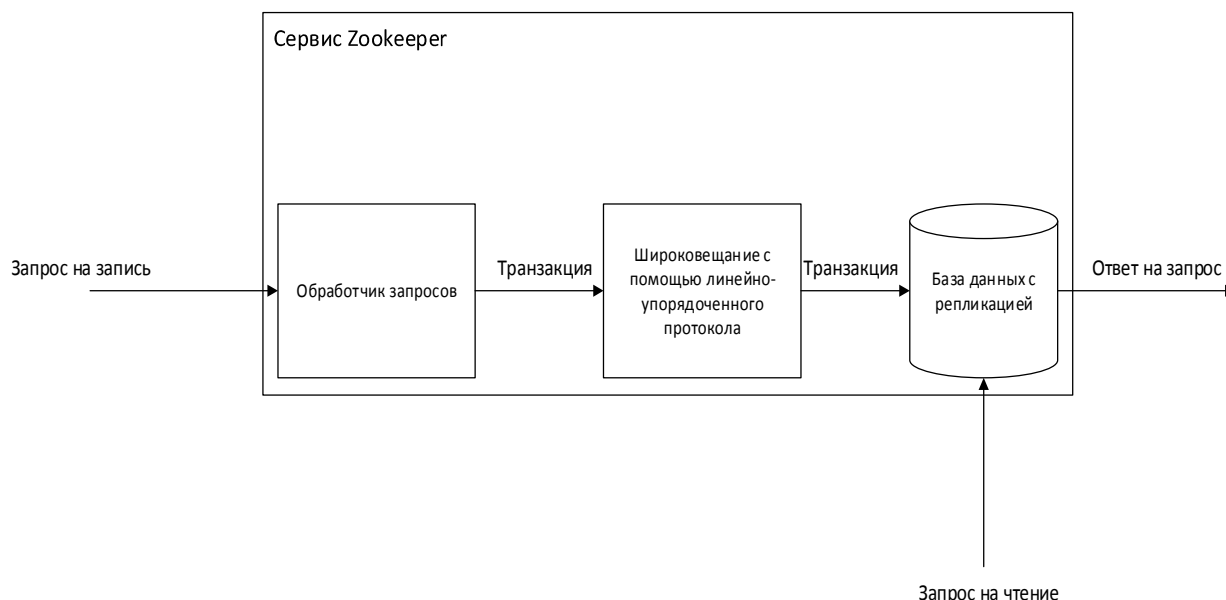


Рис. 2 Структурная схема логических компонентов Apache Zookeeper

На приведенном выше рисунке показано логическое построение системы. Запросы на чтение обслуживаются из локальной реплики базы данных, содержащей состояние системы. Запросы на запись преобразуются в идемпотентные транзакции и отправляются с помощью протокола Zab, перед тем как происходит формирование ответа на запрос.

Многие запросы на запись условны по своей природе, например, узел в иерархии может быть удален, если у него нет дочерних узлов; запрос на изменение данных будет произведен только если у данных в узле определенная версия. Безусловные запросы на запись модифицируют метаданные, такие как номера версий, делая их неидемпотентными.

При отправке всех модификаций через единый сервер, называемый лидером, производится преобразование неидемпотентных запросов в идемпотентные транзакции. Термин транзакция используется для обозначения идемпотентной версии запроса. Лидер может осуществлять преобразование, поскольку он может вычислить новое состояние, которое не будет модифицировано никем, кроме него. Идемпотентная транзакция является изменением, применяемым для преобразования к новому состоянию.

В предложенной архитектуре распределенной автоматизированной системы потокового вещания видео Apache Zookeeper используется для хранения примитивных структур данных, однако существуют и более сложные схемы хранения атомарных объектов (см., например, [5]).

Выводы

Использование единой системы для координации позволяет сделать систему масштабируемой за счёт увеличения числа узлов Zookeeper и клиентов, а также наличия у клиентов согласованного представления об общем состоянии распределенной системы. Apache Zookeeper подходит для обеспечения масштабируемости и отказоустойчивости автоматизированной системы потокового вещания видео по следующим соображениям:

- 1 Клиентское ПО устройств может использовать Apache Zookeeper для хранения информации о статусах, таких как статус соединения, статус записи с камеры, требование о записи потока на сервере-приёмник;
- 2 Серверы-приёмники получают распределённое и отказоустойчивое хранилище для информации о статусах клиентского ПО устройствам, а также для информации о наличии других серверов-приёмников (например, для отказоустойчивой записи потока с устройства несколькими, но не всеми, серверами-приёмниками, необходимо установить максимальное количество серверов, осуществляющих запись, о которых было бы известно всем серверам-приёмникам);
- 3 Клиентское ПО АРМ оператора также получает распределённое и отказоустойчивое хранилище для получения информации о статусах устройств, установки своих статусов (например, статус о требовании вещания с конкретного устройства), и получении информации о серверах-приёмниках (например, для запроса случайного IP-адреса сервера-приёмника, с которого будет осуществляться вещание конкретному клиенту);
- 4 Для добавления новых серверов-приёмников, устройств и АРМ администратора необходимо обеспечить их соединение с Zookeeper и согласованную работу по использованию разделяемых узлов (znode).

В итоге, предложена расширяемая архитектура с масштабируемым количеством клиентов и серверов, удовлетворяющая поставленным к программному обеспечению среднего звена требованиям.

Список литературы

1. ITU-T Recommendation G.114. International telephone connections and circuits General Recommendations on the transmission quality for an entire international telephone

- connection. One-way transmission time. Available at: <http://www.itu.int/rec/T-REC-G.114-200305-I/en>, accessed 01.03.2015.
2. Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. Commun. ACM 21, 7 (July 1978), 558-565. DOI: 10.1145/359545.359563
 3. Проект Apache Zookeeper. Режим доступа: <https://cwiki.apache.org/confluence/display/ZOOKEEPER/Index> (дата обращения 01.03.2015).
 4. Проект GStreamer. Режим доступа: <http://gstreamer.freedesktop.org/> (дата обращения 01.03.2015).
 5. Pierre Sutra, Etienne Riviere, and Pascal Felber. A practical distributed universal construction with unknown participants. In MarcosK. Aguilera, Leonardo Querzoni, and Marc Shapiro // Principles of Distributed Systems. 2014. Vol. 8878 of Lecture Notes in Computer Science. P. 485–500.
 6. Косяков М.С. Введение в распределенные вычисления. СПб: НИУ ИТМО, 2014. 155 с.