УДК 004.75 **DOI: 10.18413/2518-1092-2026-11-2-0-4**

Хлопов А.М.
Сулханов И.И.

СИСТЕМНЫЙ АНАЛИЗ АРХИТЕКТУР И МОДЕЛЬ ВЫБОРА ДЛЯ РАСПРЕДЕЛЁННЫХ СИСТЕМ СИНХРОННОЙ ДОСТАВКИ ПОТОКОВОГО МУЛЬТИМЕДИЙНОГО КОНТЕНТА С КООРДИНАЦИЕЙ УПРАВЛЯЮЩИХ СИГНАЛОВ

Белгородский государственный технологический университет им. В.Г. Шухова,
ул. Костюкова, 46, г. Белгород, 308012, Россия

e-mail: 20044002ilya@gmail.com

Аннотация
В статье рассматривается задача системного анализа архитектур распределённых систем синхронной доставки потокового мультимедийного контента с координацией управляющих сигналов в сценариях совместного просмотра. Актуальность обусловлена тем, что пользовательский опыт определяется одновременно характеристиками медиадоставки и контура управления, а несогласованность этих контуров приводит к рассинхронизации, задержкам реакции и деградации интерактивности. Проблема заключается в сложности архитектурного выбора, при котором функциональные требования к синхронизации потоков и нефункциональные требования системы должны быть учтены до этапа выбора конкретных протоколов и реализации. Методы: применяется подход системного анализа с декомпозицией решения на медиаплоскость и плоскость управления; выделяются базовые классы архитектур по принципу организации медиадоставки (pull-based и push-based). Результаты: предложена модель выбора архитектурного класса, основанная на анализе функциональных и нефункциональных требований к синхронизации потоковых данных, дополняемая анализом рисков выбранного класса, способных привести к пересмотру требований. Выводы: результаты могут быть использованы как методологическая рамка раннего архитектурного отбора при проектировании систем синхронного VOD-просмотра и низколатентных real-time сценариев, снижая вероятность поздней смены архитектурного подхода.

Ключевые слова: синхронная доставка мультимедиа; совместный просмотр; архитектура распределённых систем; real-time системы; pull-based архитектуры; push-based архитектуры; координация управляющих сигналов; системный анализ
Для цитирования: Хлопов А.М., Сулханов И.И. Системный анализ архитектур и модель выбора для распределённых систем синхронной доставки потокового мультимедийного контента с координацией управляющих сигналов // Научный результат. Информационные технологии. – Т.11, №2, 2026. – С. 36-49. DOI: 10.18413/2518-1092-2026-11-2-0-4

Khlopov A.M.
Sulkhanov I.I.

SYSTEMIC ANALYSIS OF ARCHITECTURES AND A SELECTION MODEL FOR DISTRIBUTED SYSTEMS OF SYNCHRONOUS STREAMING MULTIMEDIA CONTENT DELIVERY WITH CONTROL SIGNAL COORDINATION

Belgorod State Technological University named after V.G. Shukhov,
46 Kostyukova St., Belgorod, 308012, Russia

e-mail: 20044002ilya@gmail.com

Abstract
The paper addresses the problem of systemic analysis of architectures for distributed systems providing synchronous delivery of streaming multimedia content with coordinated control signals

in watch-together scenarios. The relevance of the study is determined by the fact that user experience depends simultaneously on media delivery characteristics and the control plane, while inconsistency between these layers leads to desynchronization, delayed reactions, and degradation of interactivity. The problem lies in the complexity of architectural decision-making, where functional requirements for stream synchronization and non-functional system requirements must be considered prior to selecting specific protocols and implementation approaches. Methods: a systems analysis approach is applied with decomposition into media and control planes; fundamental architecture classes are identified based on the organization of media delivery (pull-based and push-based). Results: a model for selecting an architectural class is proposed, based on the analysis of functional and non-functional requirements for stream synchronization, supplemented by risk analysis of the chosen class that may necessitate requirement revision. Conclusions: the results can be used as a methodological framework for early architectural selection in the design of synchronous VOD systems and low-latency real-time scenarios, reducing the likelihood of late-stage architectural changes.
Keywords: synchronous multimedia delivery; co-watching; distributed system architecture; real-time systems; pull-based architectures; push-based architectures; control signal coordination; systems analysis
For citation: Khlopov A.M., Sulkhanov I.I. Systemic Analysis of Architectures and a Selection Model for Distributed Systems of Synchronous Streaming Multimedia Content Delivery with Control Signal Coordination // Research result. Information technologies. – T. 11, № 2, 2026. – P. 36-49. DOI: 10.18413/2518-1092-2026-11-2-0-4

ВВЕДЕНИЕ

В последние годы всё шире используются сервисы совместного просмотра видео, когда несколько пользователей, на разных устройствах, смотрят один и тот же контент и могут вместе управлять воспроизведением (пауза, перемотка, запуск и т.д.). На практике такая функциональность требует решения двух связанных задач: надёжной доставки мультимедийного потока каждому участнику и скоординированного распространения управляющих сигналов, чтобы состояние «комнаты» просмотра у всех совпадало [10, 12, 26].

Особенность подобных систем состоит в том, что медиапоток и управляющие события распространяются по разным каналам и с разными задержками. Если задержка доставки видео у пользователей отличается, то даже корректно отправленная команда (например, «пауза») может примениться на разных моментах таймлайна. В результате возникает рассинхронизация, которая может достигать значимых величин и заметно ухудшать пользовательский опыт, особенно при потоковой передаче по сегментированным HTTP-подходам, где задержки у разных зрителей могут различаться на секунды [6, 13, 14, 19, 23–25, 30].

В статье рассматривается архитектурный уровень решения данной задачи – без привязки к конкретным реализациям или протоколам. Анализ фокусируется на способах организации медиаплоскости и плоскости управления, их взаимодействии и влиянии на синхронность, задержки и масштабируемость системы. На основе системного анализа выделяются типовые классы архитектурных подходов и исследуются их характерные свойства и ограничения в контексте совместного просмотра мультимедийного контента.

ОСНОВНАЯ ЧАСТЬ

Цель работы – выполнить системный анализ архитектурных подходов к синхронной доставке мультимедийного контента с координацией управляющих сигналов и сформировать модель выбора архитектуры под конкретные требования, а также описать типовые риски архитектур. В работе вводятся критерии сравнения, выделяются базовые классы архитектурных решений и приводится их сопоставление по ключевым параметрам, после чего предлагается итоговая модель выбора.

МАТЕРИАЛЫ И МЕТОДЫ ИССЛЕДОВАНИЯ

Постановка задачи и основные понятия

В работе рассматривается задача синхронной доставки потоковых данных группе пользователей. Потоки данных и управляющие сигналы передаются по разным каналам и с разными задержками, что приводит к рассогласованию состояния. Анализируются архитектурные подходы, применяемые для решения данной задачи в распределённых системах. На основе анализа формируется модель выбора архитектуры на основании требований и рисков [1, 4, 9].

Далее вводится базовая модель предметной области и основные понятия, используемые в работе.

Комната – логическая сессия совместного просмотра, объединяющая группу пользователей, которые воспроизводят один и тот же мультимедийный контент в согласованном режиме.

Состояние комнаты – совокупность параметров, необходимых для поддержания синхронности и воспроизводимости сессии. Минимально включает:

- идентификатор/адрес воспроизводимого контента (URL, media ID, stream ID);
- текущую целевую позицию воспроизведения на таймлайне (плейхед);
- режим воспроизведения (play/pause, скорость, при наличии);
- и может включать вспомогательные данные: список участников, роли, параметры доступа, признаки актуальности состояния (версия и т.д.), метки времени и т.п.

Плейхед – логический указатель времени на таймлайне контента, задающий целевую позицию воспроизведения, которая должна быть согласована между участниками комнаты.

Управляющие события – дискретные команды, изменяющие состояние комнаты и/или требующие согласованного применения на клиентах. Типовые события: start/pause/resume, seek(t), смена контента, подключение/отключение участника, смена роли, изменение настроек воспроизведения и т.п. Источник события – пользователь (UI-действие) либо система (правила синхронизации, восстановление после сбоев, таймеры, бизнес-логика).

Рамка системного анализа архитектур

Для системного сравнения решений удобно разделить общую задачу синхронного стриминга на две плоскости: медиаплоскость и плоскость управления. Этот подход аналогичен широко применяемому разделению плоскости данных и плоскости сигнализации в телекоммуникациях [2, 3, 15]. В контексте совместного просмотра:

- Медиаплоскость охватывает всё, что связано с доставкой аудио-видеоданных от источника контента к клиентским плеерам. Сюда входит выбор протокола и способа стриминга (например, сегментированный HTTP-поток типа HLS/DASH, либо же мало-задержочный поток типа WebRTC, RTP и т.п.), организация буферизации на клиенте, использование адаптивного битрейта (ABR) для подстройки качества под канал, механизмы декодирования и вывода медиа. Медиаплоскость определяет такие параметры, как базовая задержка потока (например, HLS обычно имеет latency порядка десятков секунд, WebRTC – сотни миллисекунд), равномерность задержек у разных клиентов, устойчивость к потере пакетов, и т.д. Именно от решений в медиаплоскости зависит, насколько синхронно вообще могут получать контент разные пользователи – есть ли у них общий ориентир по времени или каждый “живёт” в своём временном лаге относительно источника [6, 13, 14, 19, 22–25, 29].
- Плоскость управления охватывает обмен служебными сообщениями и командами для обеспечения синхронности. Это логический слой, который следит, чтобы состояние комнаты оставалось согласованным. В плоскости управления решаются вопросы: как рассылаются управляющие события (от одного клиента напрямую другим или через центральный сервер сигнализации); кто и где хранит текущее эталонное состояние (например, хранят ли клиенты локально позицию плейхеда или есть центральный координатор, считающийся источником

правды); как новые участники узнают текущее состояние при присоединении; как контролируется право инициировать те или иные команды (например, все ли могут жать Play/Pause или только “хост” сессии). Также плоскость управления может включать сопутствующие функции – обмен сообщениями/чатом, реакции, однако их можно рассматривать отдельно от синхронизации воспроизведения. Главное – механизм распространения сигналов синхронизации должен быть оперативным и надёжным, будь то широковещательные UDP-пакеты, WebSocket-сообщения или peer-to-peer канал данных [10-12, 16, 17, 21, 26].

Оси классификации архитектур. Используя указанное разделение, можно классифицировать архитектурные решения по нескольким ключевым осям (параметрам):

1. Способ организации медиапотока (доставка медиа). Здесь варианты лежат в спектре от буферизированных HTTP-подходов до реалтайм-подходов. Буферизированные подходы подразумевают, что каждый клиент независимо загружает потоковое видео через сегментированные файлы (HLS, MPEG-DASH) либо прогрессивное скачивание, используя существующую CDN-инфраструктуру. Реалтайм-подходы предполагают, что поток доставляется синхронно по сути в режиме реального времени – либо через прямые peer-to-peer соединения, либо через сервер ретрансляции, с минимальной буферизацией (WebRTC, RTMP/RTSP в режиме live, multicast и пр.). Между ними возможны компромиссы, например: использование Low Latency HLS (ускоренная версия HTTP-потока с маленькими сегментами ~1 с и т.д.) или CMAF, которые сокращают задержку, но все еще опираются на HTTP; либо использование P2P-сетей для доставки сегментов. Важное различие – независимая и общая доставка: при HTTP-подходе каждый клиент получает свои копии сегментов и медиаплоскость не гарантирует единого времени у всех (без внешней синхронизации), тогда как при общем потоке (например, один WebRTC-поток на всех) все клиенты получают один и тот же кадр “одновременно”.

2. Способ организации доставки управляющих событий. Управляющие сигналы могут передаваться несколькими путями. Через центральный узел, когда все клиенты подключаются к серверу сигнализации, который принимает команды от одного участника и ретранслирует их всем остальным. Децентрализованно, peer-to-peer, когда клиенты сами обмениваются сигналами между собой напрямую. А также гибридно, например при иерархической передаче, когда сигнал от инициатора идёт «ведущему» или серверу, а от него – остальным, либо при использовании существующей структуры контента для передачи сигналов. Важно также, как часто или по какому триггеру шлются сигналы синхронизации: возможно, как только при событии, так и периодически посылать метки с текущим состоянием, чтобы корректировать рассинхронизацию.

3. Расположение “истины” состояния (кого считать источником правды). Иначе говоря, где хранится и откуда берётся эталонное значение общего времени воспроизведения, по которому все выравниваются. Возможны несколько принципиально разных подходов. (a) Централизованный эталон: выделяется сервер или сервис, который хранит текущее состояние комнаты (например, “все должны быть на позиции 5 мин 20.0 сек”) и именно он диктует остальным клиентам, где они должны находиться. Клиенты регулярно отчитываются о своём положении, а сервер при необходимости даёт команду “подтянуться” или “приостановить/ускорить” конкретным участникам, чтобы все вновь выровнялись. Такой подход часто называют server authoritative – сервер авторитарно определяет состояние. (b) Ведущий клиент (host authoritative): роль эталона играет один из клиентов, обычно создатель комнаты или первый подключившийся. Его таймлайн считается “ведущим”, а все остальные клиенты подстраиваются под него. Например, при старте все начинают воспроизведение синхронно с ведущим; если у ведущего случился буферинг и он остановился – у всех остальных система тоже приостановит видео, ожидая, пока лидер продолжит, и т.п. Достоинство – нет необходимости постоянно опрашивать всех, достаточно отслеживать лидера; недостаток – если у ведущего проблемы, они передадутся всем. (c) Распределённая или договорная “истина”: состояние выводится из объединённых данных всех клиентов. Например, можно вычислять среднее положение по всем клиентам и брать его за эталон (никто не лидер, но все стремятся к общему среднему времени). Или более тонкий вариант – взвешенное среднее с учетом типов устройств (например, мобильные могут чуть отставать, поэтому вес их позиции меньше). Ещё подход –

привязка ко внешнему времени: все клиенты синхронизируют системные часы через NTP и меткой служит реальное время. Например, если договориться, что воспроизведение должно идти так, будто начато ровно в 12:00:00 GMT, то каждый клиент может сам вычислять, где он должен быть, исходя из текущего времени. Подобный приём реализуется посредством метаданных в потоке: так, HLS может вставлять теги EXT-X-PROGRAM-DATE-TIME с меткой времени начала сегмента, и если все устройства имеют синхронные часы, то абсолютное время служит ориентиром [18, 19, 27, 28]. При распределённом подходе сложнее управление в реальном времени (например, пауза), но повышается устойчивость: отсутствие единой точки отказа и возможность компенсировать индивидуальные задержки.

Заданные выше оси позволяют описать любую конкретную реализацию. Однако цель данной работы – не анализ частных платформ, а обобщение до классов архитектурных решений. Поэтому в следующем разделе, комбинируя варианты по осям, мы выделим базовые классы подходов к синхронной доставке и рассмотрим их характеристики.

РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЯ И ИХ ОБСУЖДЕНИЕ

В рамках данной работы анализ архитектурных подходов и формирование результатов не разделяются на отдельные этапы, поскольку результаты исследования представляют собой непосредственно выводы системного анализа. Поэтому в данном разделе совместно представлены полученные классификации, модель выбора архитектуры и их интерпретация в контексте рассматриваемых сценариев.

Классы архитектур синхронной доставки

С точки зрения медиаплоскости решения удобно классифицировать по тому, кто инициирует доставку медиаданных – источник (push) или клиент (pull). В рамках данной работы выделяются два базовых класса:

1. Pull-based client-driven segment streaming (HLS/DASH-класс) – клиент самостоятельно запрашивает медиасегменты и управляет буфером/адаптацией качества; синхронизация достигается преимущественно на плоскости управления и за счёт алгоритмов коррекции.
2. Push-based real-time streaming (WebRTC/RTP-класс) – после установления сессии источник/сервер передаёт медиаданные непрерывным потоковым образом; синхронность в значительной мере обеспечивается транспортом и общей временной базой.

Каждый класс далее описывается через медиаплоскость, плоскость управления и источник “истины” состояния комнаты.

Pull-based подходы (независимая сегментная доставка с координацией событий)

Суть архитектуры.

Медиаплоскость строится на сегментированной доставке: клиентский плеер получает манифест и далее сам запрашивает последовательность сегментов (HLS/DASH и аналогичные ABR-схемы), формирует буфер и воспроизводит контент локально. Плоскость управления реализуется отдельно (как правило, централизованным сервером сигнализации): передаются события *play/pause/seek* и служебные сообщения для поддержания состояния комнаты.

Ключевой момент: буфер и задержка формируются индивидуально на каждом клиенте, поэтому медиаплоскость сама по себе строгую синхронность не обеспечивает.

Как достигается синхронизация.

Синхронизация реализуется поверх сегментной доставки через:

- согласованную модель состояния комнаты (*server-authoritative* или *host-authoritative*);
- периодические маяки с текущей позицией и состоянием воспроизведения;
- корректирующие действия – *seek*, кратковременная пауза, временное изменение скорости воспроизведения;

- политику работы с буфером (пороги допустимого рассогласования и т.д.) [8, 10, 12, 26, 27].

Сильные стороны.

- Масштабируемость медиаплоскости: нагрузка на доставку контента переносится на стандартную HTTP-инфраструктуру и/или CDN.
- Невысокая сложность медиачасти: используются готовые плееры и зрелые протоколы сегментной доставки, не требуется собственный real-time медиасервер.
- Использование ABR и типовых цепочек доставки: механизмы адаптации качества и кэширования хорошо отлажены для массового просмотра.

Ограничения.

- Дрейф плейхеда из-за независимых буферов и сетевой вариативности: без активной коррекции рассинхронизация накапливается.
- Нижняя граница задержки задаётся сегментацией и размерами буфера: команда может быть доставлена быстро, но мгновенное применение ограничено уже загруженными сегментами.
- При стремлении к строгому синхрону усложняются алгоритмы коррекции (пороги, частота выравнивания), повышается риск “дёргания” воспроизведения.

Типовые области применения.

Совместный просмотр VOD-контента и сценарии, где важны масштаб и простота, а задержки порядка секунд и “практическая” синхронность в малых/средних группах приемлемы.

Push-based подходы (общий поток с минимальной задержкой)

Суть архитектуры.

Медиаплоскость строится на потоковой доставке реального времени: после установления сессии медиаданные передаются единым потоком (WebRTC/RTP-класс протоколов и аналогичные решения), буфер минимален, используются временные метки и механизмы восстановления часов [7, 16, 17, 20–22, 28, 29]. Распространение потока организуется либо как P2P-соединения для малых групп, либо через медиасервер ретрансляции (SFU/MCU) для большего числа участников.

Плоскость управления выделяется в отдельный канал (сигнализация, управление сессией, вспомогательные события), однако синхронность уже в значительной мере обеспечивается самим транспортом и общей временной базой.

Как достигается синхронизация.

Синхронность возникает естественным образом: клиенты получают данные из общего потока и воспроизводят их по общей временной базе (*timestamp*, *clock recovery*, *jitter buffer*). Управляющие события чаще выражаются в изменении состояния источника или медиасессии (старт/стоп, смена источника, переключение дорожек). Коррекция рассинхронизации сосредоточена в механизмах компенсации задержек и дрейфа часов, а не в регулярных *seek*-операциях на уровне приложения.

Сильные стороны.

- Минимальная рассинхронизация и малая задержка: разница отображения между участниками ограничена сетевой задержкой и обработкой.
- Быстрая реакция на управляющие события: отсутствует крупный индивидуальный буфер, поэтому управление воспринимается более “живым”.
- Естественная поддержка интерактивности: голос/видео-чат, реакции и совместные действия работают в едином временном контексте.

Ограничения.

- Повышенная сложность и стоимость инфраструктуры: требуется сигнальная подсистема, NAT-traversal, медиасерверы, мониторинг качества и отказоустойчивость.
- Ограничения масштабируемости: fan-out на большую аудиторию требует существенных ресурсов и обычно дороже по сравнению с CDN-доставкой.
- Чувствительность к качеству сети: при малом буфере проблемы сети быстро проявляются фризами, падением качества или ростом задержки.

Типовые области применения.

Совместный просмотр live-контента, интерактивные форматы и обучающие сессии, где критичны минимальная задержка и строгая синхронность, а также малые группы, где real-time стек оправдан.

О гибридных подходах и границах применимости понятия

Интуитивно может показаться, что между pull-based и push-based медиадоставкой существует “гибридный” класс. Однако такое определение некорректно: pull-based и push-based подходы различаются не настройками, а самим механизмом передачи (кто инициирует доставку и как формируется буфер). Поэтому “гибрид” в смысле *смешения протоколов медиадоставки* внутри одной и той же медиаплоскости обычно не образует самостоятельного, устойчивого архитектурного класса.

Тем не менее, термин “гибрид” допустимо использовать в двух более строгих смыслах.

1) Гибрид как сближение свойств при неизменном механизме доставки.

На практике встречаются решения, в которых одна архитектура стремится по свойствам к другой:

- сегментная pull-based доставка может приближаться к real-time по задержке за счёт низколатентных режимов, уменьшения сегментов/чанков, ускоренного обновления плейлиста и усиления контуров синхронизации (частые маяки, более строгие политики коррекции, опора на временные метки и синхронизацию часов) [6, 13, 19, 23, 24, 28, 30];
- push-based real-time стек может частично “заимствовать” свойства массовой доставки (например, через каскадирование ретрансляции, многорегиональную инфраструктуру, адаптацию качества), но остаётся push-based по механике передачи.

В обоих случаях архитектура по-прежнему принадлежит исходному классу (pull-based или push-based), а “гибридность” относится лишь к приближению эксплуатационных свойств (задержка, устойчивость, точность синхронизации, стоимость).

2) Гибрид как композиция на уровне системы.

Понятие гибрида применимо на уровне системы, когда разные фичи реализованы разными архитектурами. Например, сервис может обеспечивать:

- совместный просмотр VOD-контента на pull-based сегментной доставке (HLS/DASH- класс),
- и одновременно двустороннюю голосовую/видео-связь участников на push-based real-time доставке (WebRTC/RTP-класс).

В таком случае система является гибридной по архитектуре в целом, но каждая конкретная функциональность внутри неё реализована одним из базовых классов медиадоставки.

Теперь, определив два базовых класса решений для медиаплоскости и уточнив границы употребления термина «гибрид» и то, почему в данной работе он не рассматривается как самодостаточный класс архитектуры, сведём различия подходов более формально и рассмотрим потенциальные проблемы (риски), возникающие при их реализации.

Модель выбора архитектуры и применение

На основании проведённого системного анализа архитектур синхронной доставки потокового мультимедийного контента предлагается модель выбора архитектуры, ориентированная на уровень системы и групп функциональностей, а не на отдельные протоколы или инженерные реализации. Цель модели заключается не в количественной оптимизации параметров и не в детальном проектировании, а в выявлении архитектурно корректного способа организации доставки потоковых данных и управляющих сигналов с учётом как функциональных, так и нефункциональных требований.

Модель исходит из того, что архитектурные ограничения во многих случаях задаются функциональными требованиями системы и лишь затем уточняются через требования к задержкам (синхронности) и масштабируемости.

Ограниченность однокритериального и чисто нефункционального подходов

Если рассматривать выбор архитектуры только через призму качества пользовательского опыта, выраженного минимальной задержкой и строгой синхронностью воспроизведения, модель выбора становится вырожденной: во всех случаях предпочтение будет отдаваться push-based архитектурам, обеспечивающим доставку данных в реальном времени на основе общей временной базы.

Однако подобный подход неприменим в системах промышленного масштаба. Он игнорирует ограничения, связанные с нагрузкой, масштабируемостью и эксплуатационными затратами, и не позволяет объяснить, почему в реальных системах совместного просмотра широко применяются pull-based решения [3, 5, 9, 15].

Введение второго нефункционального критерия – масштабируемости (capacity), понимаемой как способность системы обслуживать растущее число пользователей и сессий при допустимом уровне задержек, устраняет вырожденность модели. Тем не менее, даже двухкритериальная модель, основанная исключительно на нефункциональных требованиях, остаётся недостаточной. Она не объясняет ситуации, в которых выбор архитектуры фактически предопределён функциональными ограничениями и не может быть разрешён компромиссом между задержкой и нагрузкой.

Роль функциональных требований и группировка функциональностей

Предлагаемая модель исходит из того, что первичным шагом архитектурного выбора является анализ функциональных требований системы, а именно – выявление функциональностей, порождающих потоковые данные, и характера их взаимной синхронизации.

Ключевой вопрос модели формулируется следующим образом: существуют ли в системе две или более функциональности, порождающие независимые потоковые данные, которые по функциональным требованиям должны быть взаимно синхронизированы?

Под независимыми потоковыми данными понимаются аудио-, видео- или иные непрерывные потоки, формируемые различными участниками или компонентами системы и не являющиеся производными друг от друга. Примерами таких функциональностей могут служить голосовое общение нескольких участников, совместная демонстрация экрана и видеопоток, либо одновременная передача нескольких медиапотоков в рамках одного взаимодействия.

Если такие функциональности присутствуют и между ними требуется строгая синхронизация, модель утверждает, что они должны быть объединены в единую группу и реализованы в рамках push-based архитектуры. В данном случае выбор архитектурного класса определяется функционально и не является предметом оптимизации: использование pull-based подходов либо технически не позволяет обеспечить требуемую синхронность, либо приводит к неустойчивой системе при фиксированных ресурсах.

Таким образом, наличие группы функциональностей, порождающих независимые потоки, требующие синхронизации, является достаточным условием для выбора push-based архитектуры для данной части системы.

Архитектурный выбор для оставшихся функциональностей

Функциональности, не входящие в указанные группы, то есть не порождающие независимые потоковые данные, требующие взаимной синхронизации, не накладывают жёстких архитектурных ограничений. Для них архитектурный выбор осуществляется на основе нефункциональных требований, прежде всего требований к задержке, допустимой рассинхронизации и масштабируемости.

В таких случаях модель допускает использование pull-based архитектур как предпочтительного варианта с точки зрения масштабируемости и эксплуатационной простоты. Для этих функциональностей push-based подход может использоваться, но в основном определяется особенностями реализации, чем требованиями к синхронности.

Функциональности, не имеющие особых требований к синхронности, могут работать как в отдельных архитектурных контурах, так и поверх существующей push-based инфраструктуры.

Обобщённая формулировка модели

Предлагаемая модель сводится к следующим положениям:

1. Архитектурный выбор выполняется на более абстрактном уровне, чем уровень отдельных компонентов или протоколов, а именно на уровне групп функциональностей.
2. Если группа функциональностей порождает два и более независимых потока данных, которые должны синхронизироваться между собой, для неё рекомендован выбор push-based подхода.
3. Для функциональностей, не подпадающих под данное ограничение, архитектурный выбор определяется нефункциональными требованиями и сводится к компромиссу между синхронностью и масштабируемостью.
4. Попытка реализовать функционально обусловленные сценарии синхронизации в pull-based архитектуре при фиксированных ресурсах приводит к снижению масштабируемости без достижения требуемого качества синхронного взаимодействия.
5. Модель не заменяет инженерное проектирование, а задаёт архитектурную рамку, внутри которой выполняется дальнейший анализ рисков и затрат.

Роль модели в процессе проектирования

Предлагаемая модель применяется на ранних этапах проектирования системы. Сначала анализируются функциональные требования и формируются группы функциональностей, требующие потоковой синхронизации. Для таких групп архитектурный выбор фиксируется. Далее для остальных частей системы анализируются нефункциональные требования и выбираются архитектурные решения, оптимальные с точки зрения масштабируемости и эксплуатационных ограничений.

В случае выявления противоречий между требованиями к синхронности и доступными ресурсами модель предполагает не усложнение архитектуры за счёт неустойчивых инженерных компромиссов, а пересмотр требований и сценариев использования. Таким образом, модель служит инструментом архитектурной валидации требований и позволяет избежать проектных решений, заведомо приводящих к деградации системы по мере её развития.

Архитектурные риски и основания для пересмотра требований

Предложенная модель выбора архитектуры позволяет определить архитектурный класс, соответствующий совокупности функциональных и нефункциональных требований к системе синхронной доставки потокового мультимедийного контента. Однако даже корректное применение модели не устраняет архитектурные риски. Напротив, анализ рисков является неотъемлемой частью процесса выбора, поскольку именно на этом этапе выявляются внутренние противоречия требований и ограничения, неочевидные на стадии их формулирования.

В данном разделе рассматриваются архитектурные риски не как частные технические проблемы реализации, а как фундаментальные ограничения, способные привести к невозможности выполнения исходных требований и, как следствие, к необходимости их пересмотра.

Риски для функциональностей, требующих взаимной синхронизации потоков

Функциональности, порождающие два и более независимых по происхождению потоков данных, которые должны быть синхронизированы относительно друг друга (например, голос и видео участников, совместное воспроизведение с живыми комментариями, интерактивные медиасценарии), накладывают жёсткие архитектурные ограничения. Для этих функциональностей модель рекомендует использовать push-based архитектуру, так как другие подходы не дают стабильно обеспечивать требуемый уровень синхронности.

Риски при выборе push-based архитектуры

Основным недостатком push-based архитектур являются высокие требования к инфраструктуре и сложности масштабирования. Поддержка синхронного взаимодействия пользователей требует постоянных соединений, обработки медиапотоков в реальном времени и участия медиасерверов, из-за чего при увеличении числа пользователей возрастает нагрузка на вычислительные и сетевые ресурсы [5, 9, 20, 29].

На практике это может приводить к необходимости ограничения числа участников, снижения требований к синхронности или разделения сценариев использования. Дополнительным недостатком является чувствительность таких архитектур к нестабильности сети и отказам медиасерверов, которые могут одновременно затрагивать всех участников взаимодействия.

Следовательно, основной риск push-based архитектур связан с ограниченностью доступных инфраструктурных ресурсов при высоких требованиях к синхронности взаимодействия.

Риски при выборе pull-based архитектуры вопреки требованиям

Выбор pull-based архитектуры для функциональностей, требующих строгой синхронизации потоков, ограничивает минимально достижимую задержку и точность синхронизации из-за буферизованной доставки и клиентского управления загрузкой сегментов.

Попытки компенсировать эти ограничения за счёт усложнения плоскости управления (частые корректирующие команды, ускорения, паузы, частая синхронизация плейхедов) приводят к росту нагрузки, снижению предсказуемости поведения системы и ухудшению пользовательского опыта. При фиксированных ресурсах улучшение синхронности неизбежно сопровождается падением масштабируемости, что делает систему неустойчивой при росте нагрузки.

Риски для функциональностей, не требующих взаимной синхронизации потоков

Функциональности, порождающие потоковые данные, но не требующие строгой синхронизации с другими независимыми потоками, не накладывают жёстких архитектурных ограничений. Для этого класса функциональностей модель не формирует директивной рекомендации по выбору архитектурного класса. Архитектурный выбор в данном случае осуществляется исключительно на основе нефункциональных требований и компромисса между задержкой и масштабируемостью.

Отсутствие жёсткой рекомендации не устраняет риски, а, напротив, переносит ответственность за архитектурное решение на этап интерпретации требований.

Риски при смещении выбора в сторону push-based архитектуры

Выбор push-based архитектуры для функциональностей, не требующих взаимной синхронизации потоков, связан с риском избыточной сложности. Архитектура накладывает ограничения по масштабируемости и требует значительных ресурсов на поддержку real-time соединений, не предоставляя при этом функционального выигрыша, критичного для пользовательского опыта.

В таких случаях риск заключается в том, что архитектурные ограничения оказываются искусственно навязанными системе. Это может привести к преждевременному росту затрат, усложнению эксплуатации и снижению устойчивости системы без объективной необходимости. При изменении требований или росте нагрузки возникает потребность пересмотра архитектуры, несмотря на то что функциональные требования изначально этого не требовали.

Риски при смещении выбора в сторону pull-based архитектуры

Выбор pull-based архитектуры для данного класса функциональностей, как правило, соответствует текущим требованиям. Однако архитектурный риск возникает в случае недооценки возможной эволюции сценариев использования. Постепенное увеличение интерактивности, появление новых потоковых функциональностей или рост ожиданий пользователей могут привести к ситуации, в которой исходные допущения о допустимой задержке и рассинхронизации перестают быть актуальными.

В этом случае риск заключается не в технической невозможности реализации, а в архитектурной инерции: система оказывается привязана к подходу, плохо адаптируемому к новым требованиям. Выявление такого риска требует либо раннего закладывания механизмов расширения, либо готовности к пересмотру архитектурного класса в будущем.

Риски как инструмент уточнения требований

Рассмотренные риски показывают, что архитектурный выбор не может рассматриваться изолированно от требований к системе. В ряде случаев именно анализ архитектурных рисков позволяет выявить внутреннюю противоречивость требований, избыточность ограничений или неверную оценку будущего развития функциональностей.

В рамках предложенной модели риски рассматриваются не как сбои, подлежащие устранению исключительно инженерными средствами, а как сигналы необходимости пересмотра требований. Такой подход предполагает итеративный процесс: после выбора архитектурного класса и анализа характерных рисков принимается решение либо о переходе к детальному проектированию, либо об уточнении и корректировке исходной постановки задачи.

Это позволяет избежать архитектурных решений, которые формально удовлетворяют требованиям на начальном этапе, но оказываются неустойчивыми по мере развития системы и роста нагрузки.

ЗАКЛЮЧЕНИЕ

В данной работе выполнен системный анализ архитектурных подходов к построению распределённых систем синхронной доставки потокового контента с координацией управляющих сигналов. Рассмотрены базовые архитектурные классы, различающиеся способом организации медиадоставки и управления состоянием совместного просмотра, а также выявлены их фундаментальные свойства и ограничения. В процессе работы было установлено, что выбор архитектуры для систем синхронного взаимодействия определяется не отдельными протоколами или технологиями, а требованиями к синхронности, задержкам и масштабируемости системы. Между архитектурными подходами существует прямой компромисс: повышение точности синхронизации обычно приводит к росту нагрузки на инфраструктуру и снижению масштабируемости.

На основе проведённого анализа была сформулирована модель выбора архитектуры, ориентированная на сопоставление требований сценария с архитектурным классом системы. Дополнительные параметры при этом рассматриваются как ограничения и риски, учитываемые после выбора базового архитектурного подхода.

Предложенная модель может использоваться при формировании требований к системам синхронного взаимодействия и на ранних этапах проектирования распределённых интерактивных систем.

Список литературы

1. Волкова В.Н., Денисов А.А. Теория систем и системный анализ: учебник для вузов. 3-е изд. М.: Юрайт, 2021. 562 с. URL: https://urait.ru/bcode/488173 (дата обращения: 12.01.2026).

2. ГОСТ Р 57100-2025. Системная и программная инженерия. Описание архитектуры. 2025. URL: https://docs.cntd.ru/document/1312121066 (дата обращения: 18.12.2025).

3. ГОСТ Р ИСО/МЭК 25010-2015. Информационные технологии. Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов. 2015. URL: https://docs.cntd.ru/document/1200121069 (дата обращения: 09.01.2026).

4. Козлов В.Н. Системный анализ, оптимизация и принятие решений: учебное пособие. СПб.: Изд-во Политехн. ун-та, 2011. 244 с. URL: https://elib.spbstu.ru/dl/2/2887.pdf (дата обращения: 21.12.2025).

5. Митра Р., Надареишвили И. Микросервисы. От архитектуры до релиза. СПб.: Питер, 2023. 336 с.

6. Apple Inc. Low-Latency HLS. URL: https://developer.apple.com/documentation/http_live_streaming/using_low-latency_http_live_streaming (дата обращения: 14.01.2026).

7. Baugher M., McGrew D., Naslund M., Carrara E., Norrman K. The Secure Real-time Transport Protocol (SRTP). RFC 3711. 2004. URL: https://www.rfc-editor.org/rfc/rfc3711.html (дата обращения: 27.12.2025).

8. Boronat F., Lloret J., García M. Multimedia group and inter-stream synchronization techniques: a comparative study // Information Systems. 2009. Vol. 34, No. 1. P. 108–131. DOI: 10.1016/j.is.2008.05.001.

9. Clements P., Kazman R., Klein M. Evaluating Software Architectures: Methods and Case Studies. Boston: Addison-Wesley, 2001. 300 p.

10. ETSI TS 103 286-2 V1.1.1. Digital Video Broadcasting (DVB); Companion Screens and Streams; Part 2: Content Identification and Media Synchronization. 2017. URL: https://www.etsi.org/deliver/etsi_ts/103200_103299/10328602/01.01.01_60/ts_10328602v010101p.pdf (дата обращения: 03.01.2026).

11. Fette I., Melnikov A. The WebSocket Protocol. RFC 6455. 2011. URL: https://www.rfc-editor.org/rfc/rfc6455.html (дата обращения: 19.12.2025).

12. Geerts D., Vaishnavi I., Mekuria R., van Deventer O., Cesar P. Are we in sync?: synchronization requirements for watching online video together // Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2011). 2011. P. 311–314. DOI: 10.1145/1978942.1978986.

13. ISO/IEC 23000-19:2018. Information technology – Multimedia application format (MPEG-A) – Part 19: Common media application format (CMAF) for segmented media. 2018. URL: https://www.iso.org/standard/71975.html (дата обращения: 11.01.2026).

14. ISO/IEC 23009-1:2022. Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats. 2022. URL: https://www.iso.org/standard/83314.html (дата обращения: 22.12.2025).

15. ISO/IEC/IEEE 42010:2022. Systems and software engineering – Architecture description. 2022. URL: https://www.iso.org/standard/74393.html (дата обращения: 08.01.2026).

16. Keränen A., Holmberg C., Nandakumar S. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal. RFC 8445. 2018. URL: https://www.rfc-editor.org/rfc/rfc8445.html (дата обращения: 28.12.2025).

17. Mahy R., Matthews P., Rosenberg J. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). RFC 8656. 2020. URL: https://www.rfc-editor.org/rfc/rfc8656.html (дата обращения: 05.01.2026).

18. Mills D., Martin J., Burbank J., Kasch W. Network Time Protocol Version 4: Protocol and Algorithms Specification. RFC 5905. 2010. URL: https://www.rfc-editor.org/rfc/rfc5905.html (дата обращения: 17.12.2025).

19. Pantos R., May W. HTTP Live Streaming. RFC 8216. 2017. URL: https://www.rfc-editor.org/rfc/rfc8216.html (дата обращения: 10.01.2026).

20. Rescorla E. Security Considerations for WebRTC. RFC 8826. 2021. URL: https://www.rfc-editor.org/rfc/rfc8826.html (дата обращения: 23.12.2025).

21. Rosenberg J., Mahy R., Matthews P., Wing D. Session Traversal Utilities for NAT (STUN). RFC 8489. 2020. URL: https://www.rfc-editor.org/rfc/rfc8489.html (дата обращения: 06.01.2026).

22. Schulzrinne H., Casner S., Frederick R., Jacobson V. RTP: A Transport Protocol for Real-Time Applications. RFC 3550. 2003. URL: https://www.rfc-editor.org/rfc/rfc3550.html (дата обращения: 20.12.2025).

23. Seufert M., Egger S., Slanina M., Zinner T., Hoßfeld T., Tran-Gia P. A Survey on Quality of Experience of HTTP Adaptive Streaming // IEEE Communications Surveys & Tutorials. 2015. Vol. 17, No. 1. P. 469–492. DOI: 10.1109/COMST.2014.2360940.

24. Sodagar I. The MPEG-DASH Standard for Multimedia Streaming Over the Internet // IEEE MultiMedia. 2011. Vol. 18, No. 4. P. 62–67. DOI: 10.1109/MMUL.2011.71.

25. Stockhammer T. Dynamic adaptive streaming over HTTP: standards and design principles // Proceedings of the 2nd Annual ACM Conference on Multimedia Systems (MMSys 2011). 2011. P. 133–144. DOI: 10.1145/1943552.1943572.

26. van Deventer M.O., Stokking H., Hammond M., Le Feuvre J., Cesar P. Standards for Multi-Stream and Multi-Device Media Synchronization // IEEE Communications Magazine. 2016. Vol. 54, No. 3. P. 16–21. DOI: 10.1109/MCOM.2016.7432166.

27. Williams M., Gross K., van Brandenburg R., Stokking H. Inter-Destination Media Synchronization (IDMS) Using the RTP Control Protocol (RTCP). RFC 7272. 2014. URL: https://www.rfc-editor.org/rfc/rfc7272.html (дата обращения: 29.12.2025).

28. Williams M., Gross K., van Brandenburg R., Stokking H. RTP: сигнализация источника синхронизации тактовой частоты. RFC 7273. 2014. URL: https://www.rfc-editor.org/rfc/rfc7273.html (дата обращения: 07.01.2026).

29. World Wide Web Consortium (W3C). WebRTC: Real-Time Communication in Browsers. URL: https://www.w3.org/TR/webrtc/ (дата обращения: 16.12.2025).

30. Yin X., Jindal A., Sekar V., Sinopoli B. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP // Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM 2015). 2015. P. 325–338. DOI: 10.1145/2785956.2787486.

References

1. Volkova V.N., Denisov A.A. Theory of Systems and Systems Analysis: A University Textbook. 3rd ed. Moscow: Yurait, 2021. Accessed January 12, 2026. https://urait.ru/bcode/488173.

2. GOST R 57100-2025. System and Software Engineering. Architecture Description. 2025. Accessed December 18, 2025. https://docs.cntd.ru/document/1312121066.

3. GOST R ISO/IEC 25010-2015. Information Technology. Systems and Software Engineering. Systems and Software Quality Requirements and Evaluation (SQuaRE). System and Software Quality Models. 2015. Accessed January 9, 2026. https://docs.cntd.ru/document/1200121069.

4. Kozlov V.N. System Analysis, Optimization, and Decision Making: Study Guide. Saint Petersburg: Izd-vo Politekhn. un-ta, 2011. Accessed December 21, 2025. https://elib.spbstu.ru/dl/2/2887.pdf.

5. Mitra, R., and I. Nadareishvili. Microservices: From Architecture to Release. Saint Petersburg: Piter, 2023.

6. Apple Inc. “Low-Latency HLS.” Accessed January 14, 2026. https://developer.apple.com/documentation/http_live_streaming/using_low-latency_http_live_streaming.

7. Baugher M., McGrew D., Naslund M., Carrara E., Norrman K. “The Secure Real-time Transport Protocol (SRTP).” RFC 3711. 2004. Accessed December 27, 2025. https://www.rfc-editor.org/rfc/rfc3711.html.

8. Boronat F., Lloret J., García M. Multimedia group and inter-stream synchronization techniques: a comparative study // Information Systems. 2009. Vol. 34, No. 1. P. 108–131. DOI: 10.1016/j.is.2008.05.001.

9. Clements P., Kazman R., Klein M. Evaluating Software Architectures: Methods and Case Studies. Boston: Addison-Wesley, 2001. 300 p.

10. ETSI TS 103 286-2 V1.1.1. Digital Video Broadcasting (DVB); Companion Screens and Streams; Part 2: Content Identification and Media Synchronization. 2017. URL: https://www.etsi.org/deliver/etsi_ts/103200_103299/10328602/01.01.01_60/ts_10328602v010101p.pdf (дата обращения: 03.01.2026).

11. Fette I., Melnikov A. The WebSocket Protocol. RFC 6455. 2011. URL: https://www.rfc-editor.org/rfc/rfc6455.html (дата обращения: 19.12.2025).

12. Geerts D., Vaishnavi I., Mekuria R., van Deventer O., Cesar P. Are we in sync?: synchronization requirements for watching online video together // Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2011). 2011. P. 311–314. DOI: 10.1145/1978942.1978986.

13. ISO/IEC 23000-19:2018. Information technology – Multimedia application format (MPEG-A) – Part 19: Common media application format (CMAF) for segmented media. 2018. URL: https://www.iso.org/standard/71975.html (дата обращения: 11.01.2026).

14. ISO/IEC 23009-1:2022. Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats. 2022. URL: https://www.iso.org/standard/83314.html (дата обращения: 22.12.2025).

15. ISO/IEC/IEEE 42010:2022. Systems and software engineering – Architecture description. 2022. URL: https://www.iso.org/standard/74393.html (дата обращения: 08.01.2026).

16. Keränen A., Holmberg C., Nandakumar S. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal. RFC 8445. 2018. URL: https://www.rfc-editor.org/rfc/rfc8445.html (дата обращения: 28.12.2025).

17. Mahy R., Matthews P., Rosenberg J. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). RFC 8656. 2020. URL: https://www.rfc-editor.org/rfc/rfc8656.html (дата обращения: 05.01.2026).

18. Mills D., Martin J., Burbank J., Kasch W. Network Time Protocol Version 4: Protocol and Algorithms Specification. RFC 5905. 2010. URL: https://www.rfc-editor.org/rfc/rfc5905.html (дата обращения: 17.12.2025).

19. Pantos R., May W. HTTP Live Streaming. RFC 8216. 2017. URL: https://www.rfc-editor.org/rfc/rfc8216.html (дата обращения: 10.01.2026).

20. Rescorla E. Security Considerations for WebRTC. RFC 8826. 2021. URL: https://www.rfc-editor.org/rfc/rfc8826.html (дата обращения: 23.12.2025).

21. Rosenberg J., Mahy R., Matthews P., Wing D. Session Traversal Utilities for NAT (STUN). RFC 8489. 2020. URL: https://www.rfc-editor.org/rfc/rfc8489.html (дата обращения: 06.01.2026).

22. Schulzrinne H., Casner S., Frederick R., Jacobson V. RTP: A Transport Protocol for Real-Time Applications. RFC 3550. 2003. URL: https://www.rfc-editor.org/rfc/rfc3550.html (дата обращения: 20.12.2025).

23. Seufert M., Egger S., Slanina M., Zinner T., Hoßfeld T., Tran-Gia P. A Survey on Quality of Experience of HTTP Adaptive Streaming // IEEE Communications Surveys & Tutorials. 2015. Vol. 17, No. 1. P. 469–492. DOI: 10.1109/COMST.2014.2360940.

24. Sodagar I. The MPEG-DASH Standard for Multimedia Streaming Over the Internet // IEEE MultiMedia. 2011. Vol. 18, No. 4. P. 62–67. DOI: 10.1109/MMUL.2011.71.

25. Stockhammer T. Dynamic adaptive streaming over HTTP: standards and design principles // Proceedings of the 2nd Annual ACM Conference on Multimedia Systems (MMSys 2011). 2011. P. 133–144. DOI: 10.1145/1943552.1943572.

26. van Deventer M.O., Stokking H., Hammond M., Le Feuvre J., Cesar P. Standards for Multi-Stream and Multi-Device Media Synchronization // IEEE Communications Magazine. 2016. Vol. 54, No. 3. P. 16–21. DOI: 10.1109/MCOM.2016.7432166.

27. Williams M., Gross K., van Brandenburg R., Stokking H. Inter-Destination Media Synchronization (IDMS) Using the RTP Control Protocol (RTCP). RFC 7272. 2014. URL: https://www.rfc-editor.org/rfc/rfc7272.html (дата обращения: 29.12.2025).

28. Williams M., Gross K., van Brandenburg R., Stokking H. RTP: сигнализация источника синхронизации тактовой частоты. RFC 7273. 2014. URL: https://www.rfc-editor.org/rfc/rfc7273.html (дата обращения: 07.01.2026).

29. World Wide Web Consortium (W3C). WebRTC: Real-Time Communication in Browsers. URL: https://www.w3.org/TR/webrtc/ (дата обращения: 16.12.2025).

30. Yin X., Jindal A., Sekar V., Sinopoli B. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP // Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM 2015). 2015. P. 325–338. DOI: 10.1145/2785956.2787486.

Хлопов Андрей Михайлович, кандидат физико-математических наук, доцент, доцент кафедры программного обеспечения вычислительной техники и автоматизированных систем, Белгородский государственный технологический университет им. В.Г. Шухова, г. Белгород, Россия
Сулханов Илья Игоревич, студент, Белгородский государственный технологический университет им. В.Г. Шухова, г. Белгород, Россия

Khlopov Andrey Mikhailovich, Candidate of Physical and Mathematical Sciences, Associate Professor, Associate Professor of the Department of Software, Computer Engineering and Automated Systems, Belgorod State Technological University named after V.G. Shukhov, Belgorod, Russia
Sulkhanov Ilya Igorevich, Student of the Department of Software, Computer Engineering and Automated Systems, Belgorod State Technological University named after V.G. Shukhov, Belgorod, Russia