УДК 004.274

Ле Т.З., магистрант
институт информационных систем
МИРЭА — Российский Технологический Университет
РФ, г. Москва

Научный руководитель: Алпатов А.Н., к.т.н.
МИРЭА — Российский Технологический Университет
РФ, г. Москва

ГИБРИДНЫЕ ПРОТОКОЛЫ В КЛИЕНТ-СЕРВЕРНЫХ АРХИТЕКТУРАХ СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ: КЛАССИФИКАЦИЯ И АНАЛИЗ ПРИМЕНИМОСТИ

Аннотация. В работе исследуется проблема выбора транспортного протокола в клиент-серверных архитектурах систем реального времени. Обосновывается принципиальная недостаточность монопротокольных подходов для систем с гетерогенными требованиями к передаче данных. Вводится понятие гибридного протокола как архитектурного решения, при котором разнородные потоки данных маршрутизируются по независимым транспортным каналам с различными гарантиями доставки. Предлагается классификация гибридных протоколов по двум основаниям: по критерию разнородности данных и по критерию управления надёжностью. Результаты анализа применяются к задаче проектирования клиент-серверной архитектуры систем интерактивной физической симуляции.

Ключевые слова: гибридный протокол; WebSocket; WebRTC; клиент-серверная архитектура; реальное время; задержка; надёжность доставки; QUIC; классификация протоколов.

Le C.G., student
Department of Information Systems
MIREA — Russian Technological University
Russia, Moscow

Scientific Supervisor: Alpatov A.N., Candidate of Sciences in Technology
MIREA — Russian Technological University
Russia, Moscow

HYBRID PROTOCOLS IN CLIENT-SERVER ARCHITECTURES OF REAL-TIME SYSTEMS: CLASSIFICATION AND APPLICABILITY ANALYSIS

Annotation. This paper examines the problem of transport protocol selection in client-server architectures of real-time systems. The fundamental inadequacy of single-protocol approaches for systems with heterogeneous data transmission requirements is demonstrated. The concept of a hybrid protocol is introduced as an architectural solution in which heterogeneous data streams are routed over independent transport channels with different delivery guarantees. A classification of hybrid protocols along two axes — data heterogeneity and reliability control — is proposed. The findings are applied to the design of client-server architectures for interactive physical simulation systems as a specific and illustrative case.

Keywords: hybrid protocol; WebSocket; WebRTC; client-server architecture; real-time; latency; delivery reliability; QUIC; protocol classification.

Введение.

Клиент-серверные системы реального времени — интерактивные симуляции, сетевые игры, системы удалённого рендеринга, промышленная телеметрия — предъявляют к транспортному уровню требования,

находящиеся в принципиальном противоречии друг с другом. Высокочастотные потоки обновлений состояния критичны к задержке и допускают потерю отдельных пакетов. Управляющие события, напротив, требуют гарантированной доставки и сохранения порядка, но нечувствительны к задержке в пределах десятков миллисекунд. Применение единственного протокола вынуждает проектировщика жертвовать одним требованием в пользу другого [1].

Сложившаяся практика часто сводится к выбору TCP-based протоколов — прежде всего WebSocket — как наименее рискованного варианта. Это решение корректно с точки зрения надёжности, но неоптимально для высокочастотных потоков: механизм повторной передачи TCP блокирует весь последующий поток при потере одного пакета, что при частоте обновлений 30–60 в секунду приводит к воспринимаемым задержкам значительно раньше, чем к фактической потере данных [2].

Гибридные протоколы — архитектурные решения, при которых разнородные потоки данных передаются по независимым каналам с различными транспортными свойствами — представляют собой закономерный ответ на описанное противоречие. Несмотря на то, что данный подход применяется в ряде промышленных систем, его систематическое описание, классификация и анализ применимости в контексте современного веб-стека остаются недостаточно разработанными в отечественной литературе.

Цель настоящей работы — провести аналитический обзор гибридных протоколов в клиент-серверных архитектурах, предложить их классификацию и сформулировать критерии выбора архитектурного подхода в зависимости от характеристик передаваемых данных.

Основная часть.

1. Анализ ограничений монопротокольных архитектур

Для понимания природы проблемы рассмотрим два полярных подхода и их ограничения применительно к задачам реального времени.

TCP-based архитектура (WebSocket). Протокол TCP обеспечивает надёжную упорядоченную доставку через механизм подтверждений (ACK) и повторной передачи при потере. Следствием этого является явление head-of-line blocking: при потере пакета весь последующий поток данных в рамках соединения блокируется до успешной повторной доставки утраченного пакета [2]. Для сервиса обновлений состояния с частотой 60 кадров/с (интервал 16 мс) повторная доставка пакета с задержкой 50–100 мс означает, что к моменту его прибытия накопились 3–6 более актуальных обновлений. Устаревший пакет не только бесполезен, но и задерживает их обработку.

UDP-based архитектура. Протокол UDP лишён механизма подтверждений и обеспечивает минимальную задержку за счёт отсутствия контроля доставки. Потеря пакета не обнаруживаема на транспортном уровне и не вызывает блокировки потока. Эти свойства делают UDP традиционным выбором для потоков класса «высокочастотные обновления, потеря допустима». Однако прямое применение UDP для управляющих событий — изменений конфигурации, команд синхронизации — создаёт недопустимый риск: потеря такого события нарушает согласованность состояний клиента и сервера без какого-либо механизма обнаружения [3].

Таким образом, монопротокольная архитектура вынуждена принять один из двух компромиссов: либо применять TCP ко всему трафику и жертвовать задержкой высокочастотных потоков, либо применять UDP и возлагать ответственность за надёжность управляющих событий на прикладной уровень, фактически реализуя ненужную часть TCP поверх UDP.

2. Понятие гибридного протокола

Под гибридным протоколом в контексте настоящей работы понимается архитектурное решение клиент-серверной системы, при котором логически разнородные потоки данных передаются по двум или более независимым транспортным каналам, каждый из которых оптимизирован под характеристики своего потока.

Необходимо разграничить гибридные протоколы от смежных понятий. Мультиплексирование (HTTP/2, QUIC) предполагает несколько логических потоков внутри одного физического соединения с единой политикой управления перегрузкой — это не является гибридным протоколом в принятом смысле. Протокол QUIC, однако, занимает промежуточное положение: независимые потоки (streams) в рамках UDP-соединения QUIC имеют раздельное управление надёжностью, что частично реализует свойства гибридного протокола на транспортном уровне [4].

Признаками гибридного протокола являются: наличие не менее двух независимых транспортных каналов с различными гарантиями доставки; явное разделение потоков данных по каналам на основании их транспортных требований; независимое управление жизненным циклом каналов.

3. Классификация гибридных протоколов

Предлагается классифицировать гибридные протоколы по двум основаниям: по критерию разнородности данных (что разделяется) и по критерию управления надёжностью (как обеспечивается надёжность).

Таблица 1. Классификация гибридных протоколов

Класс	Основание разделения	Типичная реализация	Примеры систем
Частота / надёжность	Высокочастотный поток vs управляющие события	UDP-канал + TCP-канал	Сетевые игры (Quake, Valve Source Engine)
Направленность	Сервер→клиент vs клиент→сервер	SSE (сервер) + HTTP POST (клиент)	Push-уведомления, мониторинг
Тип содержимого	Бинарный поток vs текстовые события	WebRTC DataChannel + WebSocket	Физические симуляции, удалённый рендеринг
Приоритет	Критичные данные vs фоновые данные	Два потока QUIC с разным приоритетом	HTTP/3 + приоритизация ресурсов

На практике гибридные протоколы чаще всего относятся к первому классу (разделение по частоте и надёжности), поскольку именно это противоречие является наиболее острым в системах реального времени. Остальные классы нередко возникают как следствие или дополнение к нему.

4. Анализ современных технологий для реализации гибридных протоколов

Рассмотрим актуальный веб-стек с точки зрения пригодности для построения гибридных протоколов каждого класса (таблица 2).

Таблица 2. Характеристики протоколов применительно к гибридным архитектурам

Протокол	Транспорт	Гарантия доставки	Направление	Роль в гибриде
WebSocket	TCP	Надёжная, упорядоченная	Двунаправленный	Канал управляющих событий (класс D)
WebRTC DataChannel (reliable)	SCTP/UDP	Надёжная, упорядоченная	Двунаправленный	Канал управляющих событий (класс D)
WebRTC DataChannel (unreliable)	SCTP/UDP	Ненадёжная, без порядка	Двунаправленный	Высокочастотный канал (класс A)
SSE	TCP/HTTP	Надёжная (TCP)	Только сервер→клиент	Однонаправленный push-канал
HTTP/3 (QUIC)	UDP	По потоку, без HOL	Двунаправленный	Единый протокол без HOL blocking

Особого рассмотрения заслуживает WebRTC DataChannel. В отличие от всех остальных перечисленных протоколов, он позволяет настраивать семантику доставки на уровне отдельного канала: параметры ordered и maxRetransmits при создании DataChannel определяют, является ли канал надёжным упорядоченным (TCP-семантика), надёжным неупорядоченным, или ненадёжным с ограниченным числом повторных попыток. Это уникальное свойство позволяет реализовать полноценный гибридный протокол первого класса в рамках единой технологии, открывая два независимых DataChannel с разными параметрами надёжности [5].

Протокол QUIC занимает принципиально иную нишу: он устраняет head-of-line blocking на транспортном уровне за счёт независимых потоков внутри UDP-соединения, однако все потоки остаются надёжными. QUIC не является гибридным протоколом в принятом смысле, но снижает остроту проблемы для ряда сценариев, делая единый надёжный протокол менее вредоносным для высокочастотных потоков [4].

5. Архитектурные паттерны реализации

На основании анализа выделяются три устоявшихся паттерна реализации гибридных протоколов в современных веб-системах.

Паттерн 1: WebSocket + WebRTC DataChannel. Наиболее распространённый в браузерных системах реального времени. WebSocket обеспечивает надёжный управляющий канал (аутентификация, синхронизация, команды), WebRTC DataChannel в ненадёжном режиме — высокочастотный поток состояния. Преимущество — широкая экосистемная поддержка WebSocket; недостаток — необходимость WebRTC signaling-сервера и прохождения NAT для установки P2P-соединения, что усложняет серверную реализацию [5].

Паттерн 2: Два WebRTC DataChannel с различной семантикой. Оба канала открываются в рамках одного WebRTC-соединения с разными параметрами: reliable/ordered для управляющих событий и unreliable/unordered для обновлений состояния. Преимущество — единая технология, общий ICE/DTLS handshake; недостаток — повышенная сложность серверной реализации WebRTC по сравнению с WebSocket [3, 5].

Паттерн 3: HTTP/3 с приоритизацией потоков. Применим в системах, где браузер взаимодействует с сервером исключительно через HTTP/3 (QUIC). Независимые потоки с различным приоритетом снижают задержку критичных обновлений. Ограничение — HTTP/3 является протоколом «запрос-ответ» и требует серверного push или дополнительного механизма для инициирования передачи со стороны сервера [4].

Таблица 3. Сравнение архитектурных паттернов гибридных протоколов

Паттерн	Сложность реализации	Задержка (класс A)	Надёжность (класс D)	Применимость
WebSocket + WebRTC DataChannel	Средняя	Низкая	Высокая	Широкая
Два DataChannel (WebRTC)	Высокая	Низкая	Высокая	Ограниченная
HTTP/3 с приоритизацией	Средняя	Средняя	Высокая	HTTP-системы

6. Граничные условия применимости гибридных протоколов

Гибридный протокол не является универсальным решением и целесообразен только при одновременном выполнении ряда условий. Во-первых, система должна генерировать данные как минимум двух классов с несовместимыми транспортными требованиями — если весь трафик однороден (только управляющие события или только низкочастотные обновления), введение второго канала создаёт неоправданную сложность. Во-вторых, частота высокочастотного потока должна быть достаточной для того, чтобы head-of-line blocking TCP стал реально ощутимой проблемой — как правило, это пороговое значение находится в районе 10–20 обновлений в секунду при задержке канала более 20 мс [2].

Существенным ограничением является и инфраструктурная сложность. WebRTC требует ICE/STUN-сервера для прохождения NAT, процедуры signaling для установки соединения и обработки переподключения. В производственных системах это означает дополнительные компоненты инфраструктуры и усложнение отладки. Для систем с контролируемой сетевой средой (локальная сеть, корпоративная инфраструктура) накладные расходы TCP могут оказаться приемлемыми, делая гибридный протокол нецелесообразным [3].

Применительно к задаче интерактивной физической симуляции паттерн «WebSocket + WebRTC DataChannel» является обоснованным

выбором: симуляция генерирует высокочастотный поток координат частиц (класс A, частота 30–60/с, потеря некритична) и редкие параметрические события — изменение физических параметров, команды синхронизации (класс D, потеря недопустима). Именно данная комбинация делает гибридный протокол не просто предпочтительным, но единственным архитектурно корректным решением для указанного класса систем.

Заключение.

В работе проведён аналитический обзор гибридных протоколов в клиент-серверных архитектурах систем реального времени. Показано, что монопротокольные архитектуры принципиально неспособны одновременно удовлетворить взаимоисключающие требования высокочастотных потоков обновлений и надёжных управляющих событий — ключевой проблемы широкого класса систем реального времени.

Введено понятие гибридного протокола как архитектурного решения с раздельными транспортными каналами. Предложена классификация гибридных протоколов по критерию разнородности данных и критерию управления надёжностью, выделяющая четыре класса со своими типичными реализациями. Проведён сравнительный анализ технологий WebSocket, WebRTC DataChannel, SSE и HTTP/3 (QUIC) применительно к каждой роли в гибридной архитектуре.

Выделены три устоявшихся архитектурных паттерна и сформулированы граничные условия целесообразного применения гибридных протоколов. Установлено, что паттерн «WebSocket + WebRTC DataChannel» является наиболее практичным выбором для браузерных систем с одновременными требованиями минимальной задержки и надёжной доставки. Полученные результаты образуют теоретическую базу для проектирования транспортного уровня в широком классе клиент-серверных систем реального времени.

Список литературы:

1. Kopetz H. Real-Time Systems: Design Principles for Distributed Embedded Applications. — 2-е изд. — Springer, 2011. — 376 с.
2. Grigorik I. High Performance Browser Networking. — O'Reilly Media, 2013. — 395 с.
3. Fiedler G. Networked Physics in Virtual Reality [Электронный ресурс] // GDC 2015. — URL: https://gdcvault.com/play/1022195 (дата обращения: 01.03.2025).
4. Iyengar J., Thomson M. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000 [Электронный ресурс]. — URL: https://www.rfc-editor.org/rfc/rfc9000 (дата обращения: 01.03.2025).
5. Blum J. WebRTC for the Curious [Электронный ресурс]. — URL: https://webrtcforthecurious.com (дата обращения: 01.03.2025).
6. Fette I., Melnikov A. The WebSocket Protocol. RFC 6455 [Электронный ресурс]. — URL: https://www.rfc-editor.org/rfc/rfc6455 (дата обращения: 01.03.2025).
7. Simpson K. You Don't Know JS: Async & Performance. — O'Reilly Media, 2015. — 296 с.