Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУ «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА ИУ7 «ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

Методы достижения консенсуса в многопользовательских играх

Студент **ИУ7-54Б** _____ **А.Г. Карапетян**

Руководитель _____ **А.С. Кострицкий**

2024 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой ИУ7
(Индекс)
_____ И. В. Рудаков
(И.О.Фамилия)
«20» сентября 2024 г.

З А Д А Н И Е
на выполнение научно-исследовательской работы

по теме

Методы достижения консенсуса в многопользовательских играх

Студент группы **ИУ7-54Б**

Карапетян Анна Григорьевна

Направленность НИР (учебная, исследовательская, др.): **учебная.**

Источник тематики (кафедра, предприятие, НИР): **кафедра.**

График выполнения НИР: 25% к 4 нед., 50% к 9 нед., 75% к 13 нед., 100% к 15 нед.

Техническое задание

Проанализировать предметную область: ввести основные определения, обозначить основные вехи развития. Формализовать задачу достижения консенсуса в многопользовательских играх. Перечислить методы или группы методов решения, сформулировать критерии сравнения. Сравнить перечисленные методы по сформулированным критериям.

Оформление научно-исследовательской работы:

1. Расчетно-пояснительная записка на **12-15** листах формата А4.
2. Перечень графического (иллюстративного) материала (плакаты, слайды и т. п.): Презентация на **3** слайдах. В презентации должны быть отражены формализованная постановка задачи и результаты сравнения методов решения.

Дата выдачи задания «20» сентября 2024 г.

Руководитель НИР _____ **А. С. Кострицкий**
(Подпись, дата) (И.О.Фамилия)

Студент _____ _____
(Подпись, дата) (И.О.Фамилия)

Временная метка генерации: 24.11.24 г., 22:06:45

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Многопользовательские игры представляют собой сложные распределенные системы, в которых согласованность данных между участниками играет ключевую роль. Успех таких игр во многом зависит от обеспечения честности игрового процесса, предотвращения читерства и согласования действий между игроками в реальном времени. Однако проблема достижения консенсуса в подобных системах остается актуальной, поскольку читеры и злоумышленники активно ищут способы нарушить целостность данных. Разработка методов достижения консенсуса в таких условиях требует учета специфики игровых процессов, архитектуры систем и существующих угроз.

Целью данной научно-исследовательской работы является сравнение существующих методов достижения консенсуса в многопользовательских играх.

Для достижения цели необходимо решить следующие задачи:

1. исследовать основные принципы консенсуса в распределенных системах и их применение в многопользовательских играх;
2. рассмотреть ключевые методы защиты игрового процесса от читерства и их роль в обеспечении консенсуса;
3. формализовать процесс достижения консенсуса в контексте многопользовательских игр;
4. определить основные критерии оценки этих методов в условиях игр;
5. провести сравнительный анализ методов консенсуса с учетом специфики игровых систем.

1 Консенсус в распределенных системах

В распределенных системах консенсус играет важнейшую роль, позволяя узлам достичь согласованности данных даже в условиях сбоев или недобросовестных участников. Это особенно актуально для многопользовательских игр, где необходимо обеспечить согласованное состояние игрового мира для всех игроков. Рассмотрены ключевые алгоритмы достижения консенсуса и их использование в игровой индустрии.

Paxos (1989 г.) [5], разработанный Лесли Лэмпортом, является одним из первых и наиболее известных алгоритмов консенсуса. Он обеспечивает согласованность в распределенных системах при наличии частичных отказов и сетевых задержек. Paxos состоит из трех фаз: предложения значения, принятия значения и подтверждения принятого значения. Этот алгоритм гарантирует, что система всегда достигает согласованного состояния, даже если некоторые узлы вышли из строя.

Применение Paxos в играх: используется для репликации игровых данных между серверами, гарантирует, что все игроки видят одинаковое состояние игрового мира, даже в случае отказа одного из серверов, позволяет восстановить данные после сбоя, минимизируя потерю игрового прогресса.

Raft (2014 г.) [4], разработанный Онгаро и Оустерхутом, был создан как более понятная альтернатива Paxos. Raft структурирует процесс консенсуса вокруг лидера, который принимает решения и реплицирует данные другим узлам. Если лидер выходит из строя, алгоритм выбирает нового лидера.

Особенности Raft: простота реализации по сравнению с Paxos, высокая производительность благодаря централизованному управлению через лидера.

Применение Raft в играх: подходит для серверных систем с репликацией состояния, где один сервер принимает решение, а остальные подтверждают его, обеспечивает быстрое восстановление при сбоях лидера, что минимизирует время простоя.

Practical Byzantine Fault Tolerance (PBFT) (1999 г.) [7], предложенный Кастро и Лисков, является алгоритмом, устойчивым к злоумышленникам и сбоям. Он способен работать при наличии до одной трети вредоносных узлов, что делает его подходящим для распределенных систем, требующих высокой степени доверия.

Суть алгоритма: PBFT обеспечивает согласованность в распределенной

системе, даже если часть узлов ведет себя недобросовестно. Алгоритм состоит из трех фаз:

— pre-prepare – лидер (primary) инициирует процесс, отправляя запросы узлам;

— prepare – узлы обмениваются сообщениями для подтверждения корректности данных;

— commit – узлы фиксируют согласие, переходя к следующему состоянию.

Для согласия требуется одобрение минимум $2/3$ корректных узлов, что обеспечивает устойчивость при наличии до $1/3$ вредоносных.

Применение в играх: PBFT используется для защиты от читерства и атак в распределенных игровых системах, обеспечивая согласованность игровых данных даже при наличии недобросовестных участников.

2 Архитектуры многопользовательских игр

2.1 Клиент-серверная архитектура

Клиент-серверная архитектура предполагает, что основная игровая логика выполняется на сервере. Сервер выступает центральным узлом, который управляет игровым процессом, а клиенты подключаются к нему для участия в игре. Клиенты, по сути, представляют собой «упрощенные» визуализационные движки, которые обрабатывают ввод игрока, отображают графику и звук, а также передают информацию на сервер [3].

Особенностью клиентской части является так называемое предсказание действий игрока, что позволяет минимизировать задержки между действиями игрока и их отображением на экране. Это создает ощущение мгновенной реакции, несмотря на возможные сетевые задержки [3].

Сервер может быть запущен на выделенной машине (режим выделенного сервера), либо сервер и клиент могут работать на одном устройстве (режим «клиент поверх сервера»). Последний вариант часто используется в однопользовательских режимах игр, где сервер обслуживает единственного клиента [3].

Игровой цикл в таких играх может быть организован несколькими способами:

— клиент и сервер реализуются как отдельные процессы (программы);

— клиент и сервер выполняются в виде отдельных потоков в рамках одного процесса;

— клиент и сервер работают в одном потоке с единым игровым циклом, что снижает накладные расходы при локальной работе в режиме «клиент поверх сервера».

В клиент-серверной архитектуре, где сервер выступает как центральный узел управления, наибольшее применение нашли алгоритмы Paxos и Raft. Они поддерживают согласованное состояние серверов в условиях репликации данных.

Paxos хорошо подходит для распределенных серверов, так как гарантирует согласованность даже при частичных сбоях системы.

Raft, благодаря своей простоте и структурированности, часто используется для управления состоянием серверных систем с одним лидером, что обеспечивает высокую производительность и минимизацию задержек.

Таким образом, клиент-серверная архитектура выигрывает от использования этих алгоритмов, особенно в играх, где требуется надежная синхронизация данных между серверами для поддержания согласованного игрового процесса.

2.2 Peer-to-Peer архитектура

Peer-to-Peer (P2P) архитектура предполагает, что каждая машина в сети одновременно выполняет функции клиента и сервера. Каждая машина обладает авторитетом (правом управления) над определенными объектами в игровом мире. Для объектов, которыми управляет локальная машина, она действует как сервер. Для всех остальных объектов машина выступает в роли клиента, визуализируя состояние, предоставленное удаленным авторитетом [3].

Игровой цикл в P2P-архитектуре имеет более простую структуру, напоминая цикл однопользовательской игры. Однако внутренняя реализация усложняется тем, что код должен учитывать два режима работы: локальный объект с полным управлением и «прокси-объект», содержащий минимальное представление состояния удаленного объекта. Для этого часто создаются два типа игровых объектов: полноценные объекты, которыми управляет локальная машина, и их упрощенные прокси-версии [3].

Дополнительные сложности возникают из-за необходимости передачи авторитета над объектами между машинами. Например, если одна из машин выходит из игры, другие машины должны взять на себя управление ее объектами, когда в игру подключается новая машина, часть объектов может быть передана ей для распределения нагрузки [3].

Таким образом, архитектура Peer-to-Peer оказывает значительное влияние на структуру игрового цикла и требует дополнительных механизмов для синхронизации и перераспределения данных между участниками сети [3].

Peer-to-Peer (P2P) архитектура накладывает дополнительные требования к методам консенсуса, так как отсутствует централизованный сервер, а доверие между узлами может быть ограниченным. В таких системах часто используется алгоритм PBFT, который обеспечивает защиту от вредоносных действий

отдельных участников.

PBFT хорошо подходит для P2P-архитектуры, так как: позволяет сохранять согласованность данных даже при наличии до одной трети вредоносных узлов, снижает риск манипуляций состоянием игровых объектов, которыми управляют разные узлы.

Однако из-за высокой вычислительной сложности PBFT применяется преимущественно в системах с небольшим числом участников. Для более масштабных игр возможны гибридные подходы, объединяющие элементы PBFT с централизованными алгоритмами, такими как Raft.

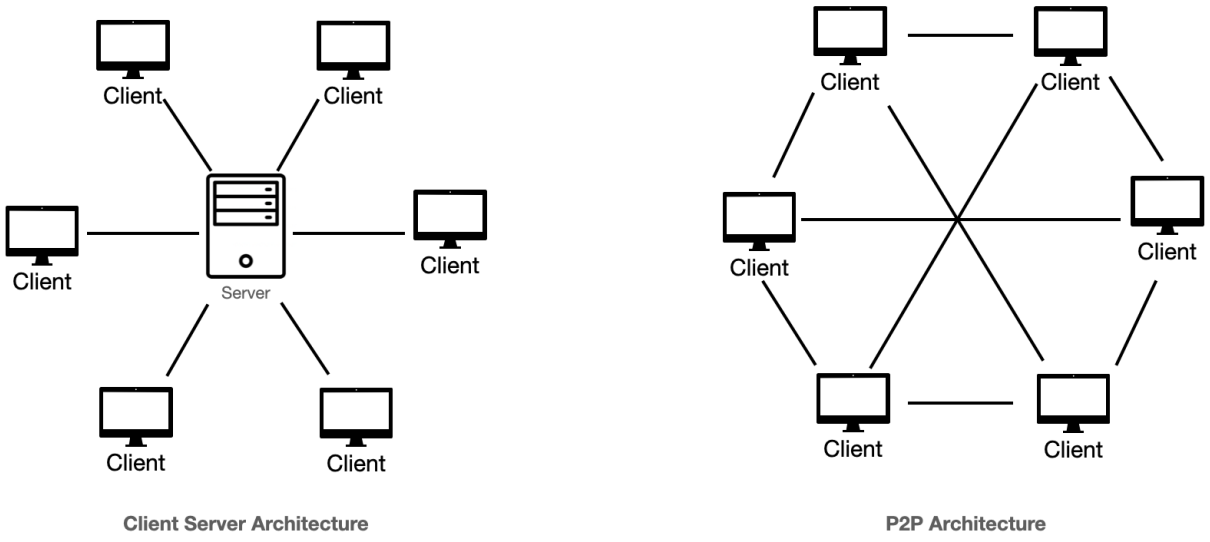

Рисунок 2.1 – Клиент-серверная и P2P архитектуры

3 Гибридные методы достижения консенсуса

В современных многопользовательских играх часто используются гибридные методы консенсуса, которые объединяют преимущества различных алгоритмов для повышения эффективности системы.

В архитектурах, где центральный сервер выполняет критические функции, Raft может использоваться для управления состоянием серверов, а PBFT — для обеспечения согласованности между участниками Peer-to-Peer сети.

Ярким примером применения подобных подходов является игра World of Warcraft (WoW) [11]. В этой MMORPG-игре используется клиент-серверная архитектура с элементами распределенной обработки. Центральные серверы отвечают за хранение и синхронизацию игрового состояния, включая данные об игроках, их взаимодействиях и событиях в мире. Однако локальные данные, такие как действия персонажа, могут временно обрабатываться на стороне клиента для снижения задержек, а затем синхронизироваться с сервером. В критических ситуациях, таких как массовые рейды или PvP-сражения, серверы применяют гибридные стратегии для обеспечения согласованности: часть данных распределяется между региональными узлами, чтобы уменьшить нагрузку, а ключевые события фиксируются на главном сервере.

В играх с распределенной архитектурой центральный сервер может разрешать конфликты, возникающие при достижении консенсуса между узлами, снижая вычислительную нагрузку на каждую отдельную машину.

Такие гибридные подходы особенно актуальны для игр с большим количеством игроков, где требуется одновременно высокая масштабируемость, устойчивость к атакам и надежность обработки данных.

4 Читерство в многопользовательских играх

Читерство в многопользовательских играх можно определить как любые действия игрока, направленные на получение нечестного преимущества над другими участниками игры, что нарушает правила, установленные разработчиками и игровым сообществом. Читерство приводит к искажению игрового баланса, ухудшению игрового опыта для других игроков и может создавать значительные проблемы для разработчиков.

Примеры атак, связанных с читерством:

— подмена данных – изменение сетевых пакетов для передачи неверной информации серверу или другим клиентам, например, подмена координат, чтобы «телепортировать» персонажа или обходить игровые ограничения;

— использование ботов – создание и использование программ, которые автоматизируют действия игрока, такие как фарм ресурсов, выполнение задач или участие в сражениях;

— автоматизация действий – применение программ или скриптов, ускоряющих реакции игрока (например, автонаведение для стрельбы в шутерах), что невозможно достичь в нормальных условиях.

Читерство тесно связано с нарушением консенсуса в многопользовательских играх, особенно в архитектурах, требующих синхронизации данных между клиентами и сервером. В клиент-серверной архитектуре подмена данных на клиентской стороне может привести к рассогласованию с сервером, что нарушает игровой баланс и требует сложных проверок на стороне сервера. В Peer-to-Peer архитектуре злоумышленник может манипулировать состоянием объектов, находящихся под его управлением, что повлияет на игровой процесс для всех участников сети.

Для предотвращения читерства разработчики внедряют различные механизмы защиты, включая верификацию данных на сервере, детектирование ботов с помощью анализа поведения игрока и алгоритмы консенсуса, устойчивые к атакам. Эти меры направлены на восстановление доверия между игроками и обеспечение справедливого игрового процесса.

4.1 Анти-читинг технологии

Анти-читинг технологии – это совокупность программных и аппаратных решений, предназначенных для предотвращения и выявления читерства в многопользовательских играх. Основная задача таких систем — поддержание честного игрового процесса, защита игроков от злоупотреблений и обеспечение соблюдения игровых правил.

4.2 Задачи анти-чит систем

Основные задачи анти-чит технологий включают:

— выявление сторонних программ – обнаружение приложений, которые могут вмешиваться в игровой процесс или изменять данные;

— проверка целостности игровых файлов – предотвращение модификации клиентской части игры, что может нарушить игровой баланс;

— мониторинг поведения игроков – анализ игрового процесса для выявления подозрительных действий, характерных для ботов или автоматизированных скриптов;

— противодействие подмене данных – защита от изменения сетевых пакетов или передачи недостоверной информации серверу.

4.3 Примеры технологий

Наиболее известные анти-читинг технологии включают:

— Valve Anti-Cheat (VAC) [10] – встроенная в платформу Steam [8] система, предназначенная для обнаружения запрещенного ПО. VAC работает на уровне анализа игровых файлов и процессов, автоматически блокируя игроков, если обнаружены нарушения;

— Easy Anti-Cheat (EAC) [2] – популярная система, используемая во многих современных играх. EAC защищает как от известных читов, так и от новых угроз с помощью технологий машинного обучения и поведенческого анализа;

— Battleye [1] – анти-чит технология, применяемая в таких играх, как PUBG [6] и Rainbow Six Siege [9], фокусируется на предотвращении модификации клиентской части игры и выявлении сетевых атак.

4.4 Роль анти-чит систем в поддержании консенсуса

Анти-чит технологии играют ключевую роль в поддержании консенсуса в многопользовательских играх. Они обеспечивают:

— целостность данных – предотвращение изменений клиентских данных, которые могут нарушить согласованное состояние игровых объектов;

— честное распределение ресурсов – блокировка автоматизированных инструментов, которые дают несправедливые преимущества;

— доверие между игроками – устранение угроз, связанных с читерством, способствует созданию честной и конкурентной среды.

Эффективность анти-чит систем во многом зависит от своевременного обновления и адаптации к новым видам атак. Системы, такие как VAC, Easy Anti-Cheat и Battleye, демонстрируют, что интеграция комплексных механизмов защиты позволяет существенно снизить уровень читерства и поддерживать справедливость игрового процесса.

5 Критерии оценки методов достижения консенсуса

Для оценки методов достижения консенсуса в распределенных системах, включая многопользовательские игры, применяются следующие основные критерии:

Продуктивность описывает эффективность метода в условиях высокой нагрузки. В контексте игр это включает: скорость обработки событий в реальном времени, таких как действия игроков или изменение состояния объектов, минимизация задержек при синхронизации данных между участниками сети.

Масштабируемость характеризует способность системы сохранять производительность при увеличении числа участников. В играх это означает: поддержку большого количества игроков без значительного увеличения задержек, эффективную работу в различных сетевых условиях, включая высокую задержку или потерю пакетов.

Устойчивость к атакам описывает способность системы сохранять согласованность данных даже при наличии злоумышленников. Это включает: противодействие попыткам подмены данных, нарушениям консенсуса и другим типам атак, которые могут нарушить нормальный процесс игры.

6 Сравнительный анализ методов консенсуса

В таблице ниже представлены основные характеристики методов консенсуса, используемых в многопользовательских играх, относительно друг друга:

Метод	Продуктивность	Масштабируемость	Устойчивость к атакам
Paxos	Средняя	Высокая	Средняя
Raft	Высокая	Средняя	Средняя
PBFT	Средняя	Низкая	Высокая

Рассмотренные методы консенсуса и архитектуры могут эффективно применяться в многопользовательских играх в зависимости от требований к производительности, устойчивости к атакам и масштабируемости.

ЗАКЛЮЧЕНИЕ

Цель данной научно-исследовательской работы была достигнута – было проведено сравнение существующих методов достижения консенсуса в многопользовательских играх.

Для достижения поставленной цели были выполнены следующие задачи:

1. исследованы основные принципы консенсуса в распределенных системах и их применение в многопользовательских играх;
2. рассмотрены ключевые методы защиты игрового процесса от читерства и их роль в обеспечении консенсуса;
3. формализован процесс достижения консенсуса в контексте многопользовательских игр;
4. определены основные критерии оценки этих методов в условиях игр;
5. проведен сравнительный анализ методов консенсуса с учетом специфики игровых систем.

В результате исследования было выявлено, что выбор метода консенсуса для многопользовательских игр зависит от требований конкретной игровой системы:

— PBFT показывает высокую устойчивость к атакам, однако его масштабируемость ограничена, что делает его подходящим для игр с высокой степенью доверия и малым количеством участников;

— Raft отличается высокой продуктивностью и является оптимальным выбором для систем с умеренными требованиями к масштабируемости;

— Paxos демонстрирует сбалансированные характеристики, что позволяет использовать его в сценариях, где требуется высокая масштабируемость и устойчивость к сбоям.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Battleye [Электронный ресурс]. — URL: https://www.battleye.com/ (дата обр. 15.12.2024).
2. Easy Anti-Cheat [Электронный ресурс]. — URL: https://www.easy.ac/en-US (дата обр. 15.12.2024).
3. *Gregory J.* Game Engine Architectur. — A K Peters, Ltd, 2009.
4. In Search of An Understandable Consensus Algorithm (Raft) [Электронный ресурс]. — URL: https://shubheksha.com/posts/2017/10/in-search-of-an-understandable-consensus-algorithm-raft/ (дата обр. 15.12.2024).
5. *Lamport L.* Paxos Made Simple. — 2001.
6. PUBG: Battlegrounds [Электронный ресурс]. — URL: https://www.pubg.com (дата обр. 16.12.2024).
7. *S.Tanenbaum A.*, *Steen M. V.* Distributed Systems: Principles and Paradigms. — Upper Saddle River, 2007.
8. Steam [Электронный ресурс]. — URL: https://store.steampowered.com (дата обр. 16.12.2024).
9. Tom Clancy's Rainbow Six Siege [Электронный ресурс]. — URL: https://www.ubisoft.com/en-gb/game/rainbow-six/siege (дата обр. 16.12.2024).
10. Valve Anti-Cheat [Электронный ресурс]. — URL: https://help.steampowered.com/en/faqs/view/571A-97DA-70E9-FF74 (дата обр. 15.12.2024).
11. World of Warcraft [Электронный ресурс]. — URL: https://worldofwarcraft.blizzard.com (дата обр. 16.12.2024).