Авторитарный протокол в сетевых боевиках

Из первых рук

Презентация компании
www.firstlinesoftware.ru
2017

“Извини за грубость исполнения модели. У меня не было времени сделать в масштабе или раскрасить.”
Эммет “Док” Браун, х.ф. “Назад в Будущее”.

“Моя сосна! Космический мерзавец, ты сломал мою сосну!”
Фермер Пибоди, х.ф. “Назад в Будущее”.

Отличие авторитарного и неавторитарного протокола

В качестве примера будем рассматривать классический сетевой шутер от первого лица.

Мир представлен в общем случае статическими сущностями (карта), которые не меняются. В процессе игры информация о них не передается в сетевом протоколе и сетевыми сущностями, информация о которых должна передаваться между клиентами.

Игровой мир

Статические сущности:

- Сцена (карта)

Сетевые сущности:

- Игроки
- Гранаты
- Выстрелы (трассеры)

Неавторитарный протокол

Полностью доверяем клиенту, включая взаимодействие между его сущностями и сущностями других игроков.

- Передаем события, которые регистрируются на клиенте: попадания, а в общем случае, любое взаимодействие между сущностями одного игрока и другого
- Передаем информацию о сущностях клиента (игрок, граната, трассеры)
- Сервер агрегирует информацию от клиентов, производит вычисления результата, рассылает обновление мира клиентам

Неавторитарный протокол

Плюсы

- Легко реализуется
- Не требует лагокомпенсации, предсказания, интерполяции

Минусы

- Легко подделать пакет с нужными событиями
- Не имеет защиты от изменений работы клиента (спидхак, подделка дамага, телепорт, да всё что угодно!)
- Порождает большое количество нежелательных артефактов при сетевых задержках

Авторитарный протокол

Доверяем клиенту только передачу пользовательского ввода и отображение мира

- Клиент передает пользовательский ввод на сервер
- Сервер рассчитывает местоположение сущностей клиента и их взаимодействие
- Сервер рассчитывает результирующий мир и передает его клиенту для отображения

Авторитарный протокол

Плюсы

- Больше честной игры, подделка сетевых данных не влияет на геймплей
- Самая ответственная часть геймплея - регистрация взаимодействия клиентских сущностей обсчитывается на сервере

Минусы

- Сложнее запрограммировать
- Присутствуют некоторые логические парадоксы, связанные с задержкой в цепочке клиент-сервер-клиенты. Часть из них имеют удовлетворительные решения.

Итоги сравнения

К черту плюсы и минусы, как геймер, я выбираю вариант с максимально честным геймплеем. Говорят, что для читеров есть специальное место в аду!

Как работает авторитарный протокол

Тик

Сервер симулирует мир дискретно - 1 тик, например, 66 раз в секунду во время тика

- Пользовательские команды
- Симулируется физика
- Проверяются правила игры
- Обновляются сущности
- Определяется кому из клиентов необходимо выслать обновленный снимок мира

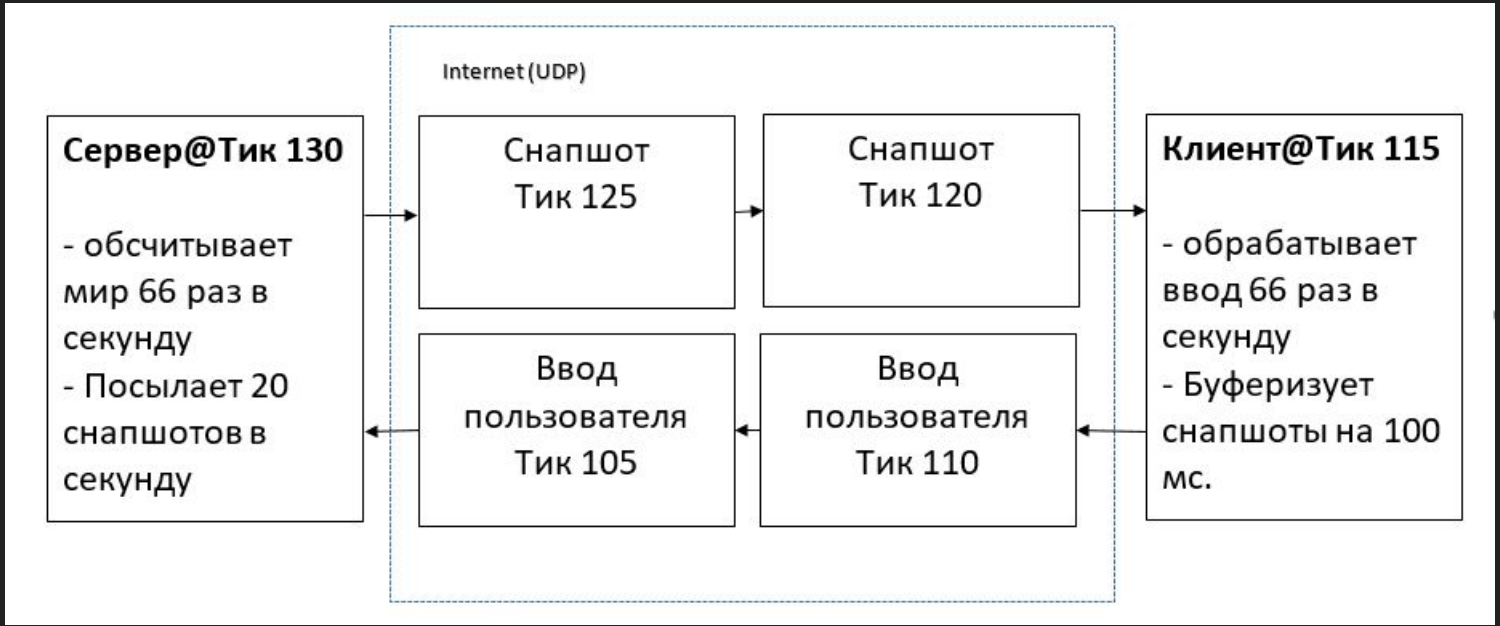
Снимок (снапшот) и тик на клиенте

Разная пропускная способность клиентов ограничивает количество снимков и отправляемого ввода.

Клиент опрашивает ввод раз в тик, но посылает пакетами по несколько команд в зависимости от пропускной способности канала.

Для оптимизации при посылке снимка используется дельта-компрессия.

Сетевой протокол



Буфер 100 миллисекунд необходим для интерполяции, но об этом позже

Авторитарный протокол

Время реакции, сетевые задержки и потери пакетов создают серьезные проблемы (регистрация попаданий, отображение мира).

Низкий пинг может быть серьезным преимуществом в сетевой игре. Ограниченная пропускная способность канала не дает нам обновлять мир у клиента с желаемой частотой.

Подробнее о решении проблем в авторитарном протоколе

Предсказание ввода

Задержка в перемещении (ватный персонаж).

Передаем на сервер только пользовательский ввод (+FORWARD +LEFT +BACK +RIGHT +JUMP +USE +SELECTWEAPON, а также статус кнопок мыши и вектор камеры игрока).

Обновление позиции сущности игрока приходят к нам с задержкой примерно большей или равной пингу.

Некоторые действия (прицеливание, точные серии прыжков) осуществить практически невозможно.

Предсказание ввода

Для расчета передвижений игрока применяется один и тот же код.

Последовательно, отдав последовательность ввода и на клиенте и на сервере, мы получим сущность в одном и том же итоговом положении.

Мы доверяем клиенту передвигать персонажа в сцене, не дожидаясь уточнения позиции от сервера, и получаем нормальную реакцию персонажа.

В случае, если подтвержденная сервером позиция отличается от полученной на клиенте, мы плавно "притягиваем" сущность игрока к подтвержденной.

Предсказание работает только для сущностей самого игрока.

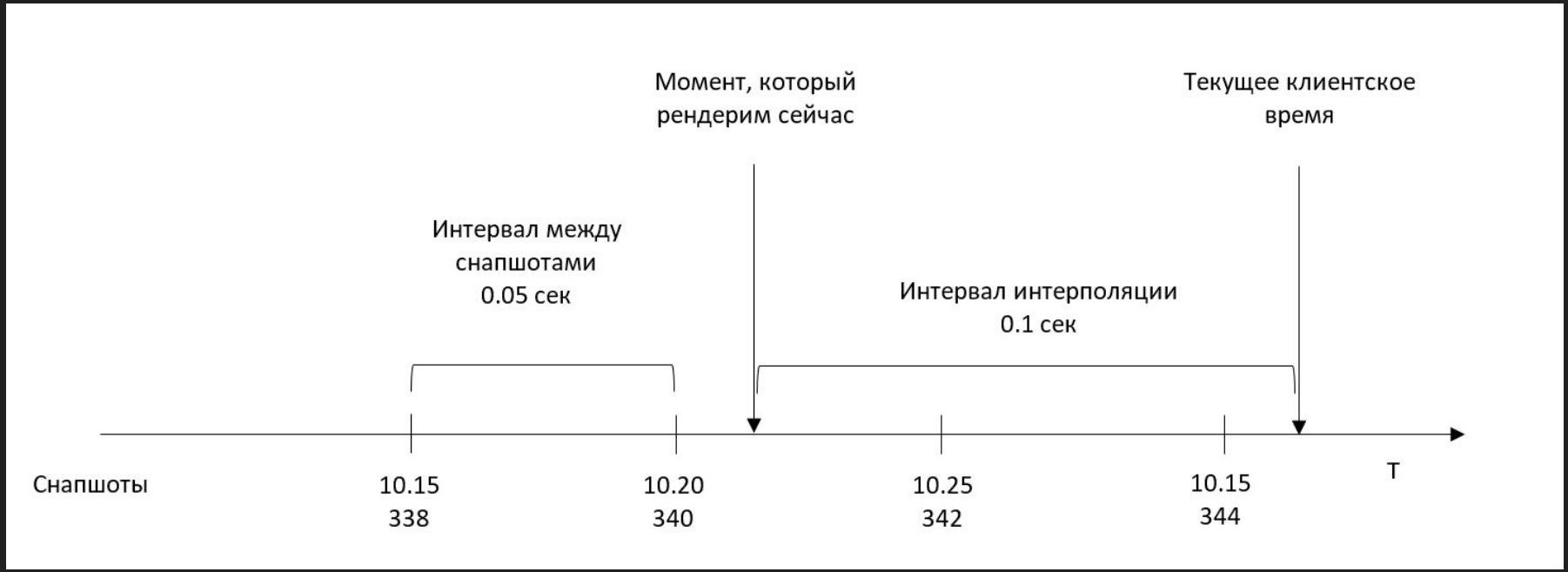
Интерполяция

Клиент получает снэпшоты сравнительно редко, реже тика и реже рендеринга кадра, что привело бы к прерывистому движению сущностей на экране. Потеря пакетов усугубляет ситуацию.

Для того, чтобы этого избежать, весь рендеринг происходит с опозданием. Рендерится промежуток между предпоследним и последним снапшотом.

Таким образом, у нас есть два подтвержденных сервером состояния, между которыми мы можем интерполировать положение сущностей и анимации.

Интерполяция



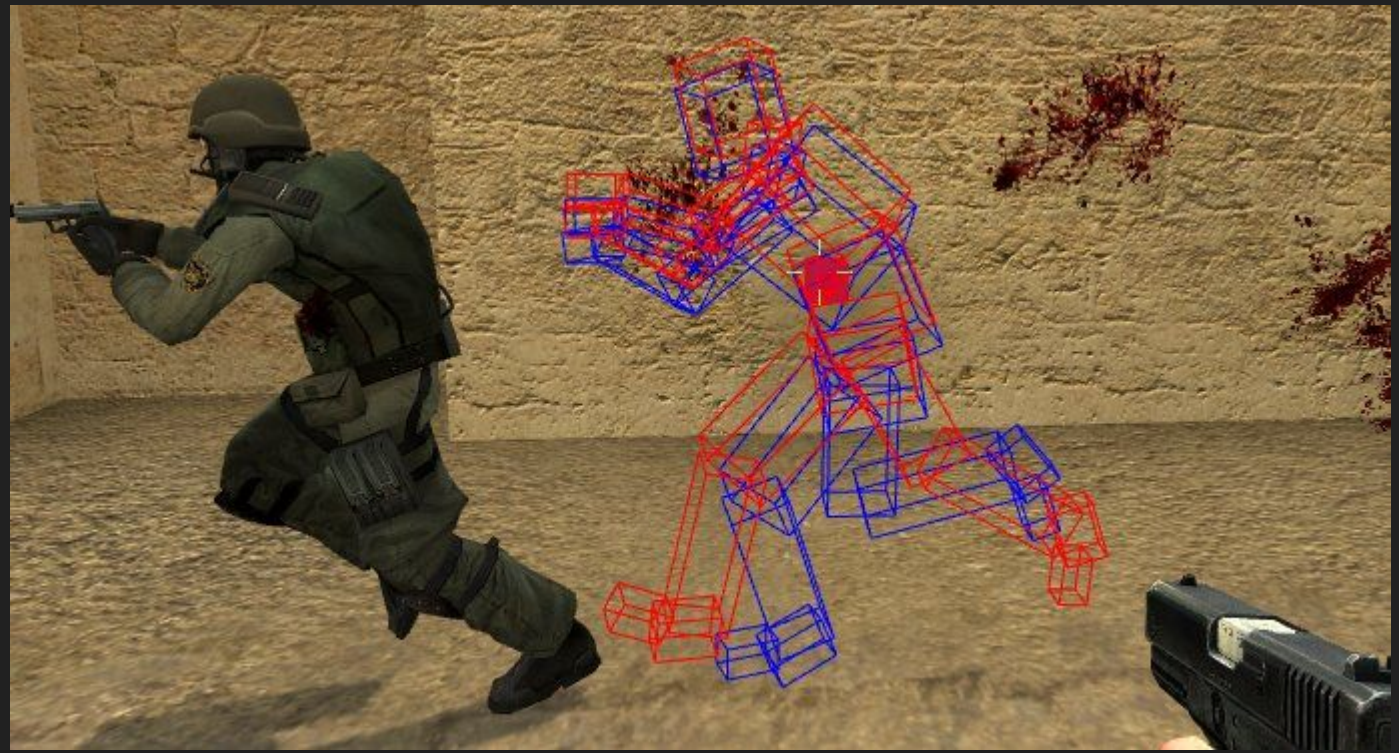
Лагокомпенсация

Храним на сервере историю сущностей за последнее время.
При расчете результата пользовательской команды все сущности перемещаются в положение соответствующее тику, в котором она была сгенерирована.

В случае возникновения событий, существенно меняющих мир (регистрация смерти одного из игроков, например) пересчитывается весь мир до текущего серверного тика, после чего обновленный снапшот рассылается клиентам.

Время исполнения команды = Текущее время сервера - Задержка интерполяции клиента

Лагокомпенсация



Парадоксы лагокомпенсации

Парадоксы, порождаемые лагокомпенсацией, обусловлены временем, затрачиваемым на передачу информации об изменениях.

В реальной жизни мы их не наблюдаем, потому что свет (“пакет с информацией”) путешествует столь быстро, что все вокруг видят мир таким же, как и вы в данный момент.

Q&A

Спасибо за внимание!

www.firstlinesoftware.ru