rus.bce_connected_servers

Alexandr Kirilov (https://github.com/alexandrkirilov)

Blockchain пример для серверных решений.

Данный пример основан на том что было сделано около 10 лет назад по принципу "just for fun". Но это шуточное задание наиболее ярко иллюстрирует применение blockchain в северных и клиент-серверных решениях для обеспечения безопасного обмена данными.

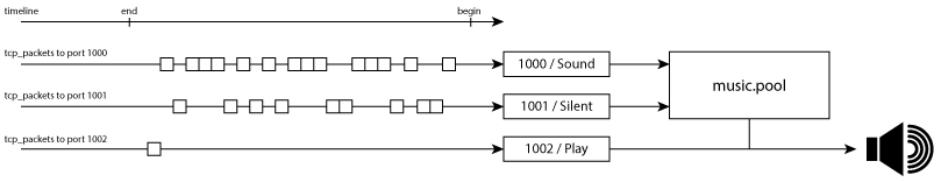
Технически не использовано ничего сложного или чего-то очень нового. Для реализации данного примера нужно:

- установленная FreeBSD начиная с версии 8.0 (это говорит о том что этой технологии 9 лет, релиз этой версии системы состоялся 25 ноября 2009 года, страница в Wikipedia подтверждающая эти даты)
- установленный пакет knockd
- установленный пакет beep
- любой клиент позволяющий посылать последовательности единичных пакетов используя различные протоколы (UDP/TCP), в данном случае использовался стандартны клиент поставляемый вместе со службой knockd

Небольшое отступление о том что делает knockd - данная служба (daemon) позволяет назначать действия (actions/commands) на последовательности (sequence) пакетов получаемых сервером. Это может быть любая комбинация пакетов, начиная от "одиночного выстрела" в один порт до "пулеметной очереди" по различным портам используя различные протоколы. В данный момент это активно используется многими системными администраторами для увеличения уровня безопасности обслуживаемых серверов, примеры использования широко представлены на различных ресурсах в интернете и могут быть найдены без особых трудностей по ключевому словосочетанию для поиска "knockd freebsd".

Задача выполняема пакетом beep очевидна - проигрывание звука определенно длины

Теперь о том что было сделано "just for fun".



Если вкратце - то последовательность пакетов трансформировалась в файл на основании которого происходило формирование параметров для проигрывания звуков на системном бипере. При помощи музыканта была составлена последовательность, которая проигрывалась как "Имперский марш" из фильма "Звездные войны".

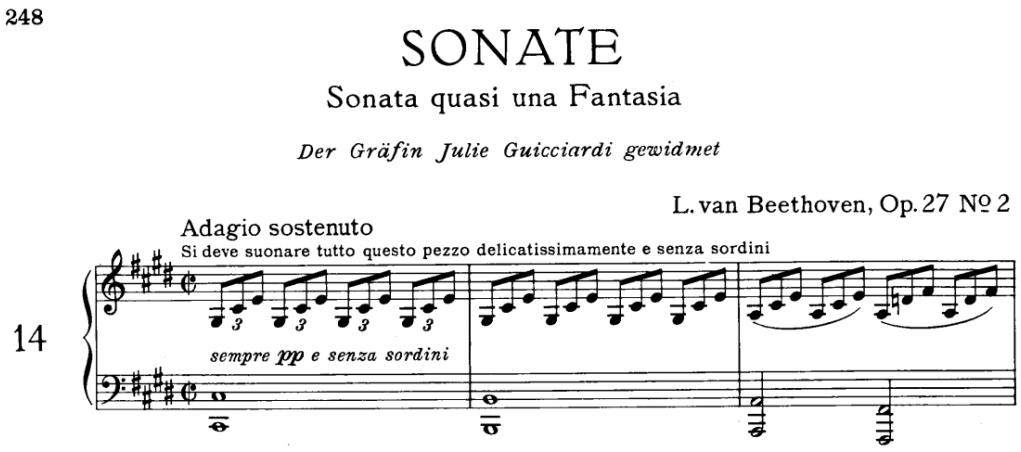
Как это работало?

Для начала была выбрана минимальная длительность проигрывания, своего рода "базовая единица времени". Она была выбрана 0.1 секунды - это было базовое значение установленное как настройка на сервере. Были назначены порты и действия к ним:

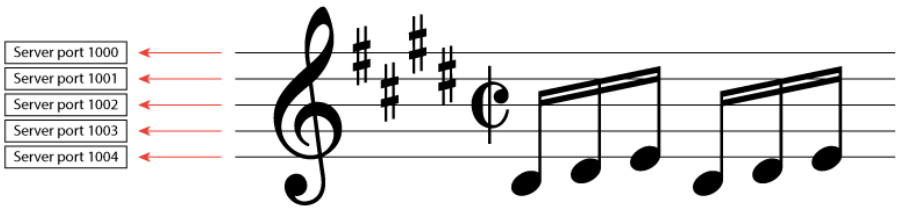
- 1000 для состояния "Beep" и к этому порту было привязано действие - записать в файл команду "играть одну единицу времени" при получении пакета на этот порт
- 1001 для состояния "Silent" и к этому порту было так же привязано действие которое записывало в файл команду, только теперь она выглядит так - "молчать одну единицу времени"
- 1002 для начала проигрывания, при получении пакета на этот порт происходило формирование параметров для запуска пакета beep и проигрывание музыки с последующей очисткой фала указанного на изображении как music.pool

После установки всех настроек, клиентское приложение посылало последовательность TCP пакетов на основании которых формировался текстовый файл который при получении пакета на порт 1002 парился и трансформировался в тройку параметров для beep.

С таким же успехом при помощи такой технологии можно передавать целые музыкальные произведения.



При первом взгляде связь между нотным станом и последовательностями пакетов на порты сервера может показаться не очевидной. Следующее изображение может проиллюстрировать структурное подобие на уровне информации.



Теперь о том что в этом примере является цепью (chain) и что является (block).

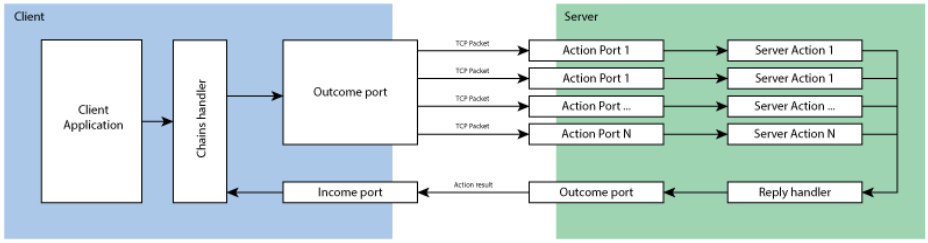
Для человека не изучавшего ноты это просто бессмысленные закорючки. Человек который изучал нотный стан сразу представит в голове звук который будет исходить от инструмента при исполнении музыкального инструмента.

С точки зрения информации музыка - это повторяющаяся активация источника звука (в данном случае фортепиано путем удара кулачка о струны) в определенной последовательности при соотношении частоты издаваемого звука к длительности звука и длительности паузы.

Иными словами последовательность активации источника звука - это цепь (chain) состояний источника звука "beep", где блоком (block) является язык при помощи которого описывается свойства звука издаваемого источником: частота и длительность. В данном случае нет жесткой последовательности того, что должно проиграть, есть язык - ноты или блок для проигрывания определенного звука с определенной частотой с определенной длительностью и на основании которой

создается другая последовательность - музыкальное произведение которое в свою очередь является цепью (chain) из состоянии источника звука, где блок (block) это произведение целиком, созданное автором и в этом состоянии заблокированное. Если в приведенном примере измените ноты местами это перестанет быть "Лунной Сонатой" Бетховена.

Теперь о том как это связано с взаимодействием между серверами или между клиентами и серверами. Связаны они между собой понятием - язык. В случае с музыкой - набор нот на нотном стане, в случае с серверами - последовательности пакетов связанных с выполняемыми действиями в контексте задач решаемых приложением на сервере.



Наиболее привычный подход один порт - одно приложение, в редких случаях приложения используют несколько портов. Во время тестирования было реализовано REST API через отсылку пакетов, где номер порта был привязан к определенному действию. При использовании данного подхода снижался трафик в несколько раз, а иногда и в десятки раз.

Технически это выглядело так - клиент отсылал на сервер пакет нулевой длины на определенный порт, при получении этого пакета на сервере выполнялось некое действие и возвращался на порт клиента определенный заранее.

В ключе политики безопасности "запрещено все, кроме того что разрешено", может быть хорошим решением удовлетворяющим требованиям.

Для реализации blockchain в контексте этой задачи нужно определить язык понятный для клиента и для сервера. Для примера можно назначить порты соответствующие символам UTF-8 и через отсылку последовательности пакетов формировать строку на стороне сервера без отсылки самой строки. Для увеличения безопасности справочник соответствий символов портам может формироваться динамически с определенным периодом времени и быть дополнен "пустышками" не позволяющими распознать язык по аналогии, где количество пустышек так же может формироваться динамически.

В данном случае цепь (chain) - это последовательность пакетов, а блок (block) - это язык понятный только клиенту и серверу с динамическим изменением языка в зависимости от необходимости.