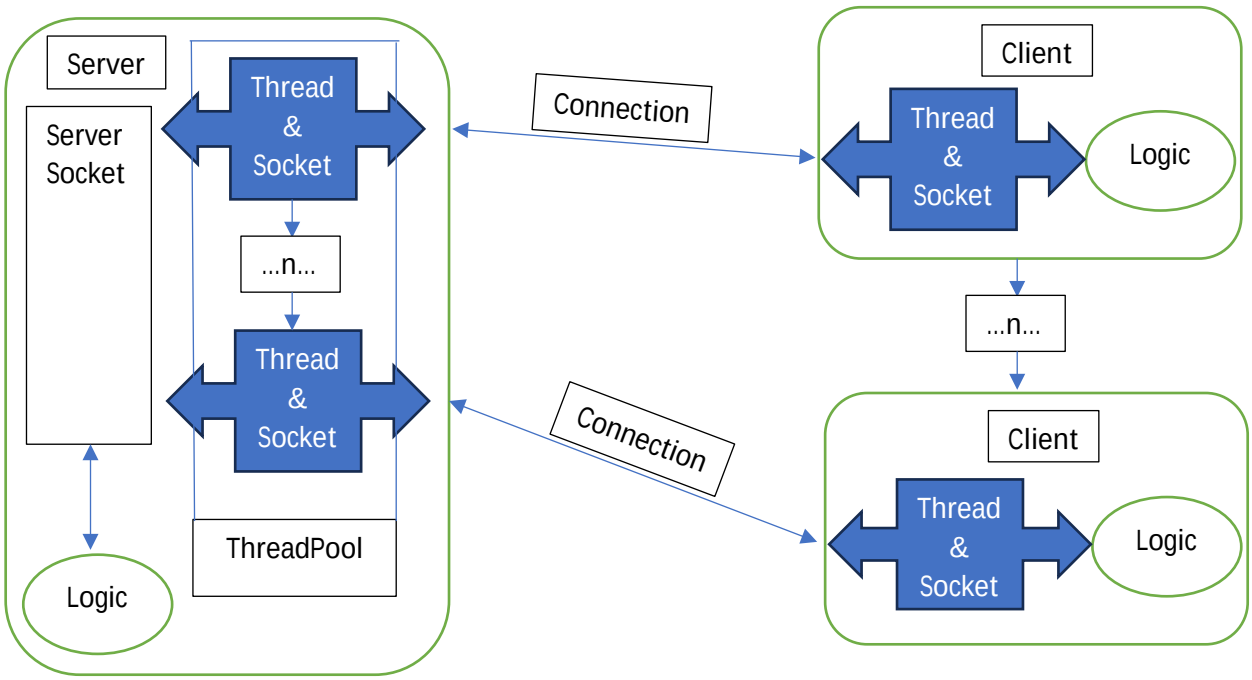
О проекте

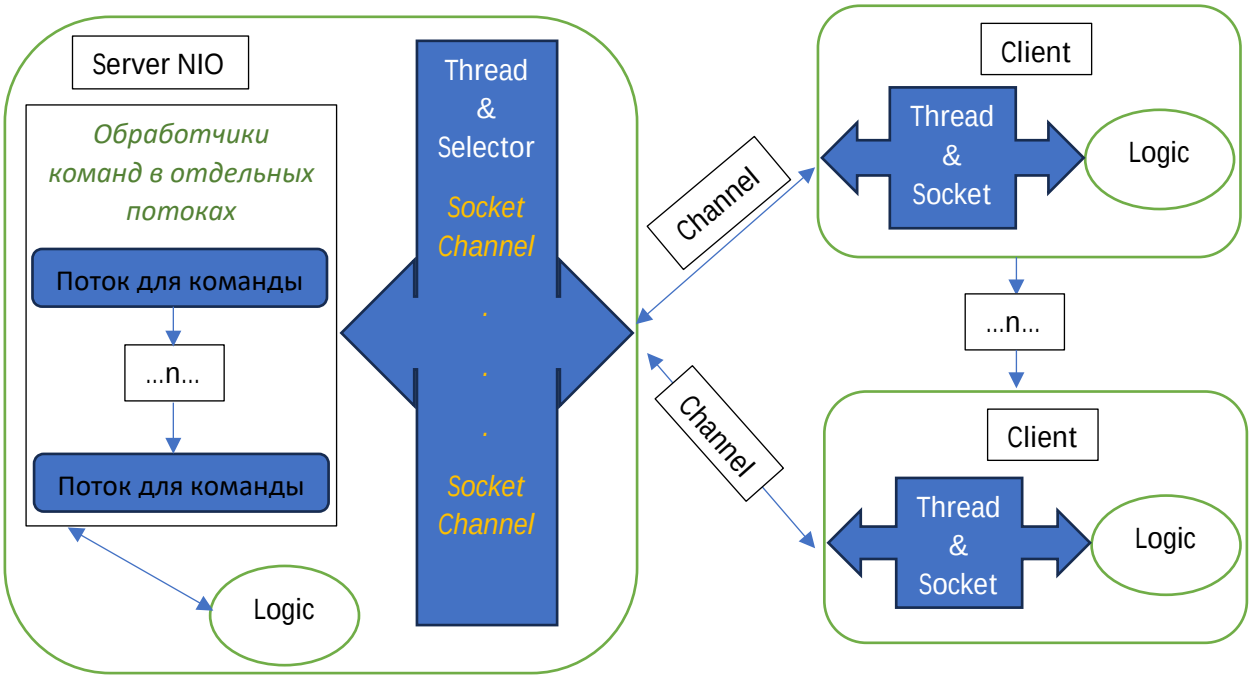
Стандартный подход к реализации клиент-серверных приложений на Java, предусматривает использование пакета I/O API с блокирующими потоками ввода/вывода, в т.ч. блокирующих socket's. Блокирующий сокет блокирует поток до тех пор, пока не получен ответ на отправленный вызов, а неблокирующие сокеты просто отправляют запрос на другой конец, не блокируя поток, и сразу могут выполнять следующею инструкцию. Использование неблокирующих socket's стало возможно благодаря Java NIO или Java New IO. Неблокирующий режим Java NIO позволяет запрашивать считанные данные из канала (channel) и получать только то, что доступно на данный момент, или вообще ничего, если доступных данных пока нет. Вместо того, чтобы оставаться заблокированным пока данные не станут доступными для считывания, поток выполнения может заняться чем-то другим. Тоже самое справедливо и для неблокирующего вывода. Поток выполнения может запросить запись в канал некоторых данных, но не дожидаться при этом пока они не будут полностью записаны. Таким образом неблокирующий режим Java NIO позволяет использовать один поток выполнения для решения нескольких задач вместо пустого прожигания времени на ожидание в заблокированном состояний. Наиболее частой практикой является использование сэкономленного времени работы потока выполнения на обслуживание операций ввода/вывода в другом или других каналах. Java NIO позволяет управлять несколькими каналами (сетевыми соединениями или файлами) используя минимальное число потоков выполнения.

В случае использования в нашей задаче данной устаревшей модели схема клиент-серверного приложения выглядела бы следующим образом:



Данный подход больше подходит для малого числа клиентских соединений, в сочетании с объемными передаваемыми данными. В случае же большого количества соединений (например общий чат в многопользовательской игре), количество потоков сильно возрастает (линейно), растут и сопутствующие этому издержки времени, обусловленные блокирующими методами и аппаратными ограничениями.

В связи с указанным, мной принято решение использовать для решения задачи фреймворк Netty, разработанный на базе Java NIO. В общем виде, схема клиент-серверного приложения на NIO выглядит следующим образом:



Разработанное приложение состоит из шести модулей:

- ChatServer – конфигурирует сервер и регистрирует обработчики данных, в т.ч. модуль ответственный за логирование событий в чате;

- ChatServerHandler – обработчик данных на стороне сервера;

- ChatClient – конфигурирует клиента, регистрирует обработчики данных, принимает дынные ввода из консоли, отвечает за прекращение работы клиента в случае получения соответствующей команды;

- ChatServerHandler - обработчик данных на стороне клиента (выводит сообщения полученные сервером от клиентов в консоль);

- LogToFile - обработчик данных на стороне сервера, ответственный за логирование событий в чате;

- FileSettingsReader – модуль, ответственный за чтение сетевых настроек сервера и клиентов и их начальную валидацию.