NIO Server vs TCP Server

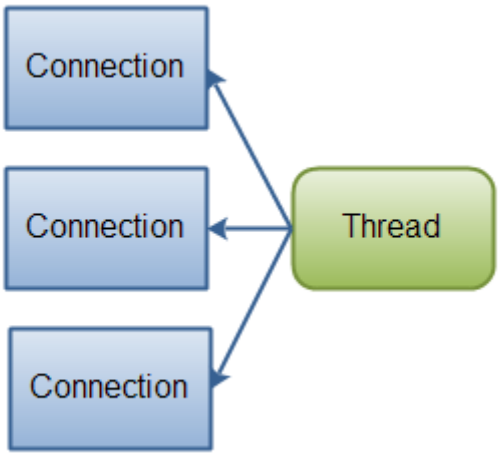
NIO Server:

Serwer dostępny w pakiecie JGroups. Służy do wysyłania i odbierania wiadomości przez kanały NIO (Non-blocking I/O). Używa pojedynczego wątku do wykonywania operacji: accept, connect, write, read connections. Odczytane wiadomości są przekazywane do receiver'a, który domyślnie używa puli wątków w celu przetworzenia wiadomości.

Wiadomości przesyłane mogą być protokołem TCP.

Cechy rozwiązania NIO Servera w kontekście IO:

- Buffer oriented
- Selectors
- Non blocking IO



Rysunek 1. Java NIO: Pojedynczy wątek zarządza wieloma połączeniami.

TCP Server:

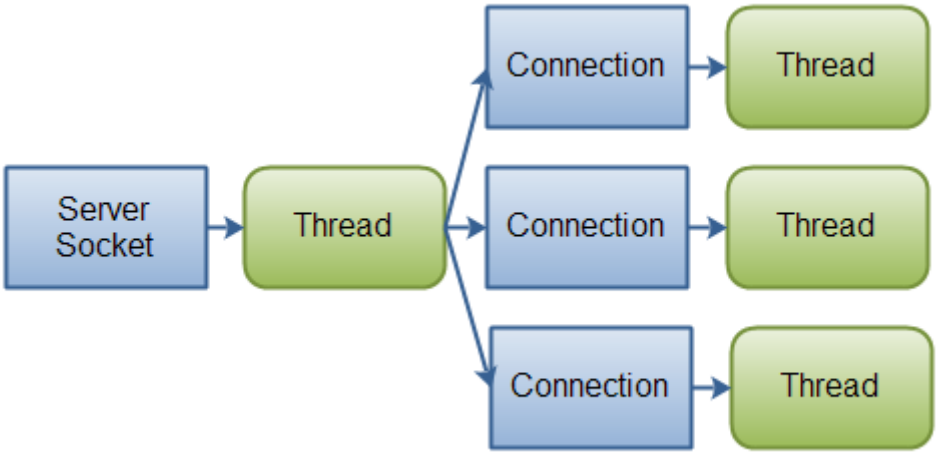
Serwer dostępny w pakiecie JGroups. Pozwala na wysyłanie i odbierania wiadomości 'via TCP sockets'. Używa 1 wątku per połączenie do odczytywania wiadomości. W przeciwieństwie do NIO Server, ten realizuję politykę BIO (Blocking I/O).

Wiadomości przesyłane są protokołem TCP.

Cechy rozwiązania TCP Servera w kontekście IO:

- Stream oriented

- Blocking IO



Rysunek 2. Java IO: Klasyczny schemat serwera opartego o podejście BIO: Każde połączenie obsługiwane jest przez pojedynczy (dedykowany) wątek.

Stream Oriented vs Buffer Oriented:

Poniższe cytaty są porównaniem tych dwóch podejść dla języka Java. Nie odnoszą się one bezpośrednio do JGroups.

Za Jenkov:

> *„Java IO being stream oriented means that you read one or more bytes at a time, from a stream. What you do with the read bytes is up to you. They are not cached anywhere. Furthermore, you cannot move forth and back in the data in a stream. If you need to move forth and back in the data read from a stream, you will need to cache it in a buffer first.*
>
> *Java NIO's buffer oriented approach is slightly different. Data is read into a buffer from which it is later processed. You can move forth and back in the buffer as you need to. This gives you a bit more flexibility during processing. However, you also need to check if the buffer contains all the data you need in order to fully process it. And, you need to make sure that when reading more data into the buffer, you do not overwrite data in the buffer you have not yet processed. „*

Za dokumentacją Oracle:

> *„NIO construction makes I/O faster than traditional I/O. In a program where the I/O operations constitute a significant amount of the processing, expect to see some difference. For example if an application has to copy files or transfer bytes using*

sockets, using Nio is possible to obtain a faster performance because it is closer to the OS than the I/O API. Increasing the byte size, the difference becomes more appreciable. Nio also provides other features not in io API, for streaming operations. However, it is not possible to substitute IO with NIO because NIO API adds functionalities to the java.io. NIO extends the native IO API introducing new possibilities for the developer to manipulate stream data in a powerful way."

Blocking vs Non-blocking IO:

Poniższe cytaty są porównaniem tych dwóch podejść dla języka Java. Nie odnoszą się one bezpośrednio do JGroups.

Za Jenkov:

„Java IO's various streams are blocking. That means, that when a thread invokes a `read()` or `write()`, that thread is blocked until there is some data to read, or the data is fully written. The thread can do nothing else in the meantime.

Java NIO's non-blocking mode enables a thread to request reading data from a channel, and only get what is currently available, or nothing at all, if no data is currently available. Rather than remain blocked until data becomes available for reading, the thread can go on with something else.

The same is true for non-blocking writing. A thread can request that some data be written to a channel, but not wait for it to be fully written. The thread can then go on and do something else in the mean time.

What threads spend their idle time on when not blocked in IO calls, is usually performing IO on other channels in the meantime. That is, a single thread can now manage multiple channels of input and output."

Opracowano na podstawie:

- *JavaDocs dla JGroups.*
- http://tutorials.jenkov.com/java-nio/nio-vs-io.html
- http://stackoverflow.com/questions/18677258/stream-oriented-io-vs-block-oriented-io
- https://blogs.oracle.com/slc/entry/javanio_vs_javaio

Powyższy dokument zawiera jedynie dokumentację techniczną związaną z komunikacją między aplikacjami poprzez toolkit JGroups. Ma on na celu zaznajomienie zainteresowanych z architekturą tworzonego systemu do monitorowania natężenia ruchu miejskiego w Krakowie.