Part 3.3: Comparison of Designs

Part 3.3(a): Comparison with xsocket

How many dispatchers does x-Socket allow? If multiple, how do the dispatchers share workload?

x-Socket allow 2 dispatchers.

In changelog.txt

- [connection] default dispatcher size (num cpu + 1) has been changed to 2

In IoProvider.java, line 407.

Share workload: round-robin approach, in IoSocketDispatcherPool.java, line 105

What is the basic flow of a dispatcher thread?

In file IoSocketDispatcher.java,

1. While current dispatcher is opened
   a) Selector waits for an event, set timeout at 5 seconds.
   b) Perform registered handler tasks
   c) If new event comes, handleReadWriteKey()
2. Deregister all socket handler
3. Close selector

What is the calling sequence until the onData method of EchoHandler (see EchoHandler, EchoServer, and EchoServerTest) is invoked? Please check this link for testing code: http://sourceforge.net/p/xsocket/code/HEAD/tree/xsocket/core/trunk/src/test/java/org/xsocket/connection/

1. EchoServerTest -> new EchoServer-> new Server()-> server set EchoHandler -> acceptor.listen() -> callback.onConnectionAccepted()->new nonBlockingConnection() -> [PerformOnDataTask]run()->[HandlerAdapter]performOnData()->[EchoHandler]onData()

How does x-Socket implement Idle timeout of a connection?

It uses IIdleTimeoutHandler, ConnectionManager to check all the registered connections periodically to see if they are timeout or not.

Please give an example of how the library does testing (seehttp://sourceforge.net/p/xsocket/code/HEAD/tree/xsocket/core/trunk/src/test/java/org/xsocket/connection/EchoServerTest.java for an example). Please describe how you may test your server with idle timeout?

It tests the libarary file by file (module by module).

For example, in file LifeCycleTest.java, it tests the LifeCycle class separately.

Write a test file to test it:

Start the server and use a client connect to it but do not send anything to see if it can kill the idle connection automatically.

Part 3.3(b): Comparison with Netty

- Netty provides multiple event loop implementations. In a typical server channel setting, two event loop groups are created, with one typically called the boss group and the second worker group. What are they? How does Netty achieve synchronization among them?

Answer:
1. The 'boss' group accepts new connections and pass them to worker group. The 'worker' group handles the connections and registers the connection to workers.
2. When an new connection comes, 'boss' group will choose one thread as Acceptor and then pass the SocketChannel to 'worker' group. 'worker' group will then choose thread to deal with network I/O.

- Method calls such as bind return ChannelFuture. Please describe how one may implement the sync method of a future.

Answer:
One may call a blocking function isDone() to wait until it is completed.

- Instead of using ByteBuffer, Netty introduces a data structure called ByteBuf. Please give one key difference between ByteBuffer and ByteBuf.

Answer:
Unlike ByteBuffer, ByteBuf is a reference-counted object which has to be released manually.

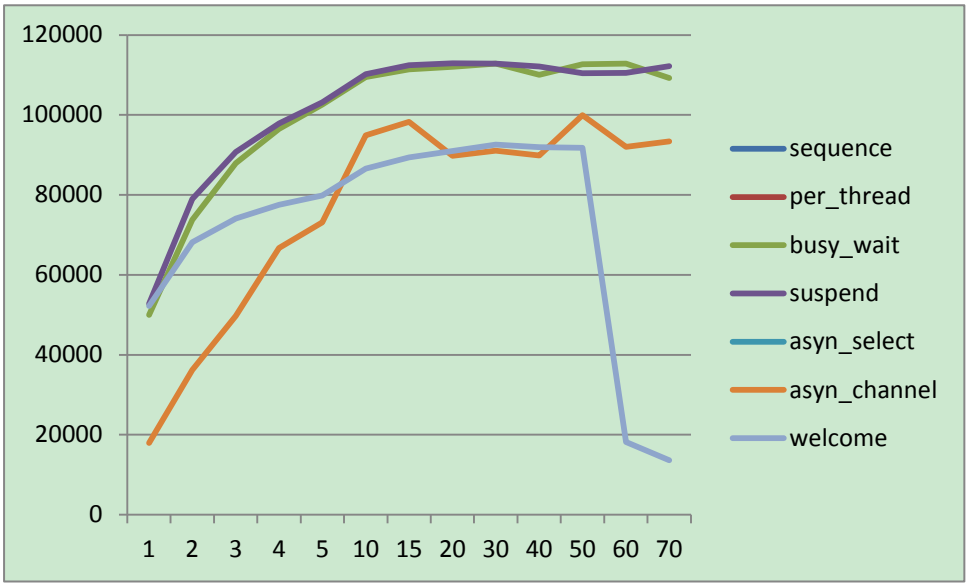
- A major novel, interesting feature of Netty which we did not cover in class is ChannelPipeline. A pipeline may consist of a list of ChannelHander. Compare HTTP Hello World Server and HTTP Snoop Server, what are the handlers that each insert?

Answer:
SnoopClient: HttpClientCodec, HttpContentDecompressor, HttpSnoopClientHandler
SnoopServer: HttpSnoopServerHandler, HttpRequestDecoder, HttpResponseEncoder,
HelloworldServer: HttpServerCodec, HttpHelloWorldServerHandler

- Please scan Netty implementation and give a high-level description of how ChannelPipeline is implemented.

Answer:
I/O Events are handled by ChannelHandler and forwarded to another one.
A Channel contains one ChannelPipeline, which is a list of all ChannelHandlers.
Handlers are managed by DefaultChannelHandlerContext in a linked list.

Part 5: Report



Thread pool with Suspension performs the best.
It's through put is above 100MB/S.
Part 6.

Main class for server:
server.SHTTPTestServer

main class for client:
client.SHTTPTestClient