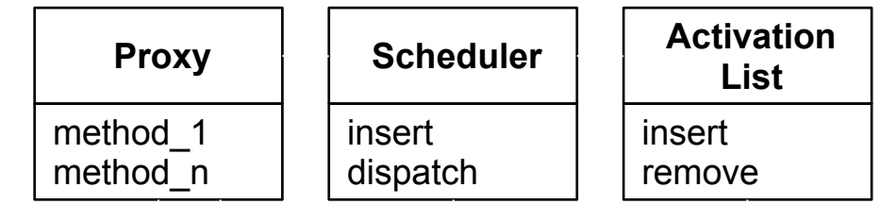
Active Object Design Pattern

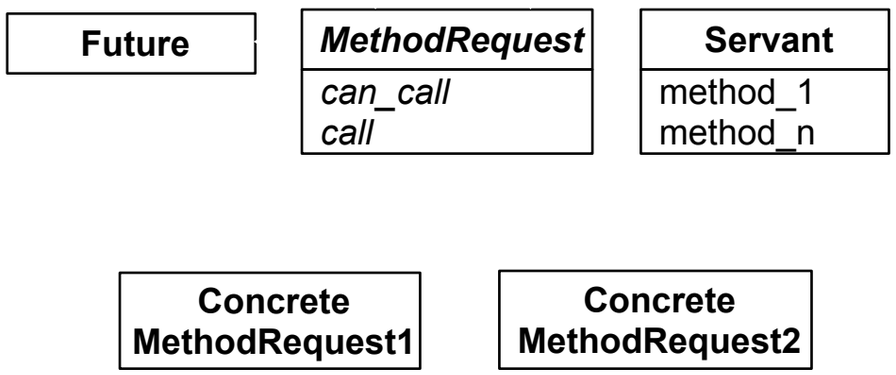
Jeremiah Jordan

Active Object

- Decouples method execution from method invocation
- Simplifies synchronized access to objects that reside in their own threads of control
- Commonly used in distributed systems requiring multi-threaded servers
- Often implemented using message passing
- Similar to the Actor pattern. In the Actor pattern each Actor is an Active Object.

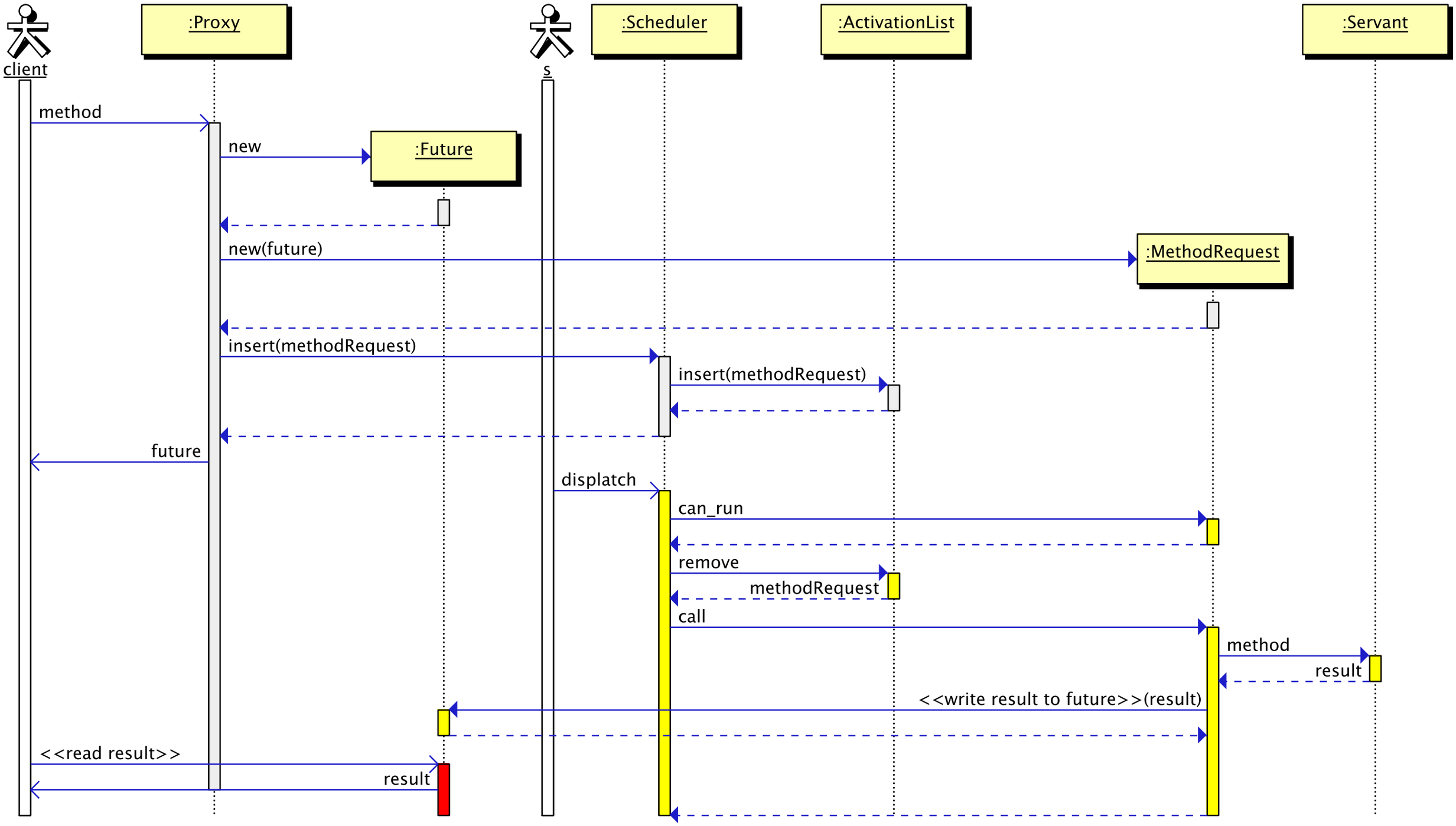
Example Class Diagram



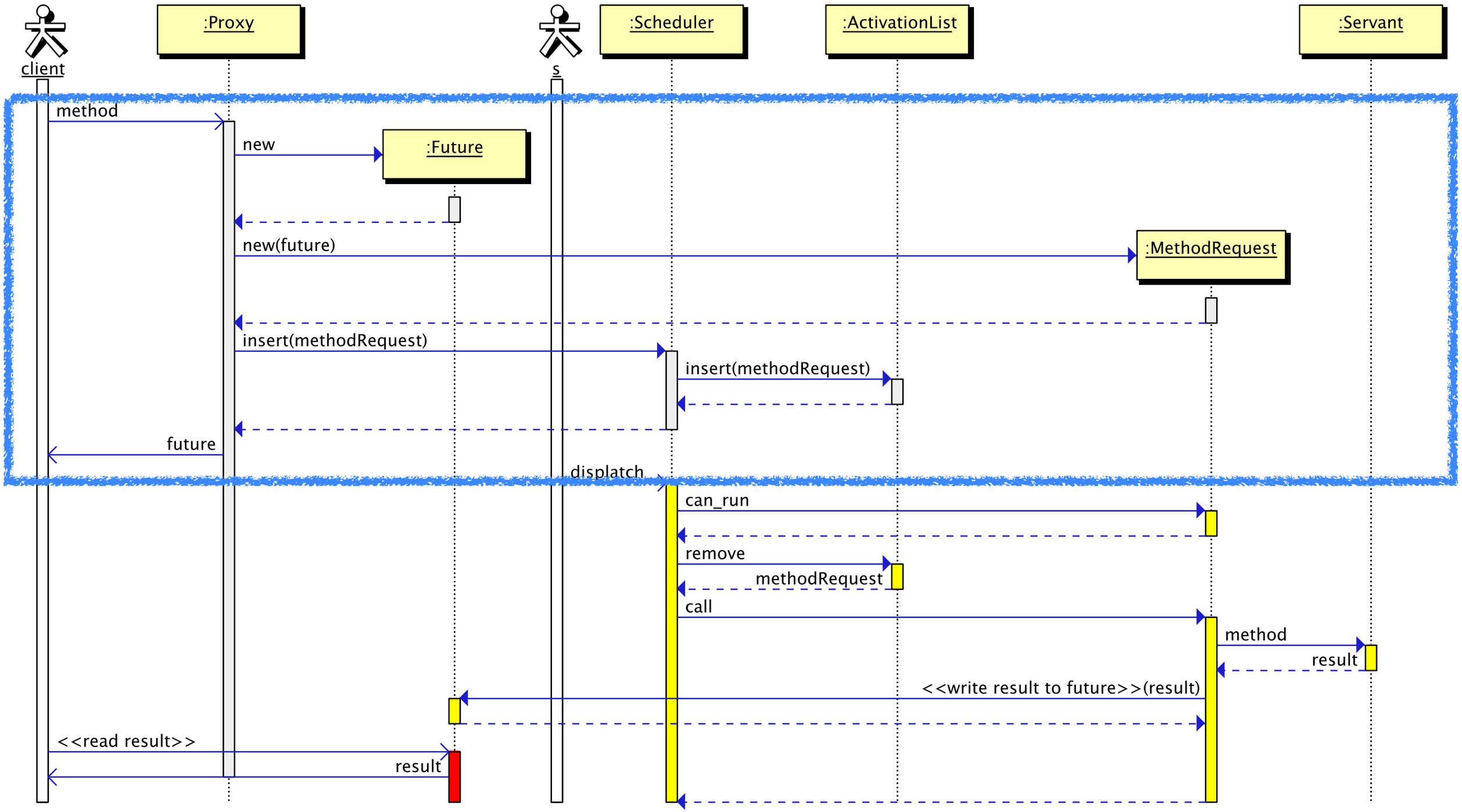


- A proxy provides an interface that allows clients to access methods of an object
- A concrete method request is created for every method invoked on the proxy
- A scheduler receives the method requests & dispatches them on the servant when they become runnable
- An activation list maintains pending method requests
- A servant implements the methods
- A future allows clients to access the results of a method call on the proxy

Sequence Diagram



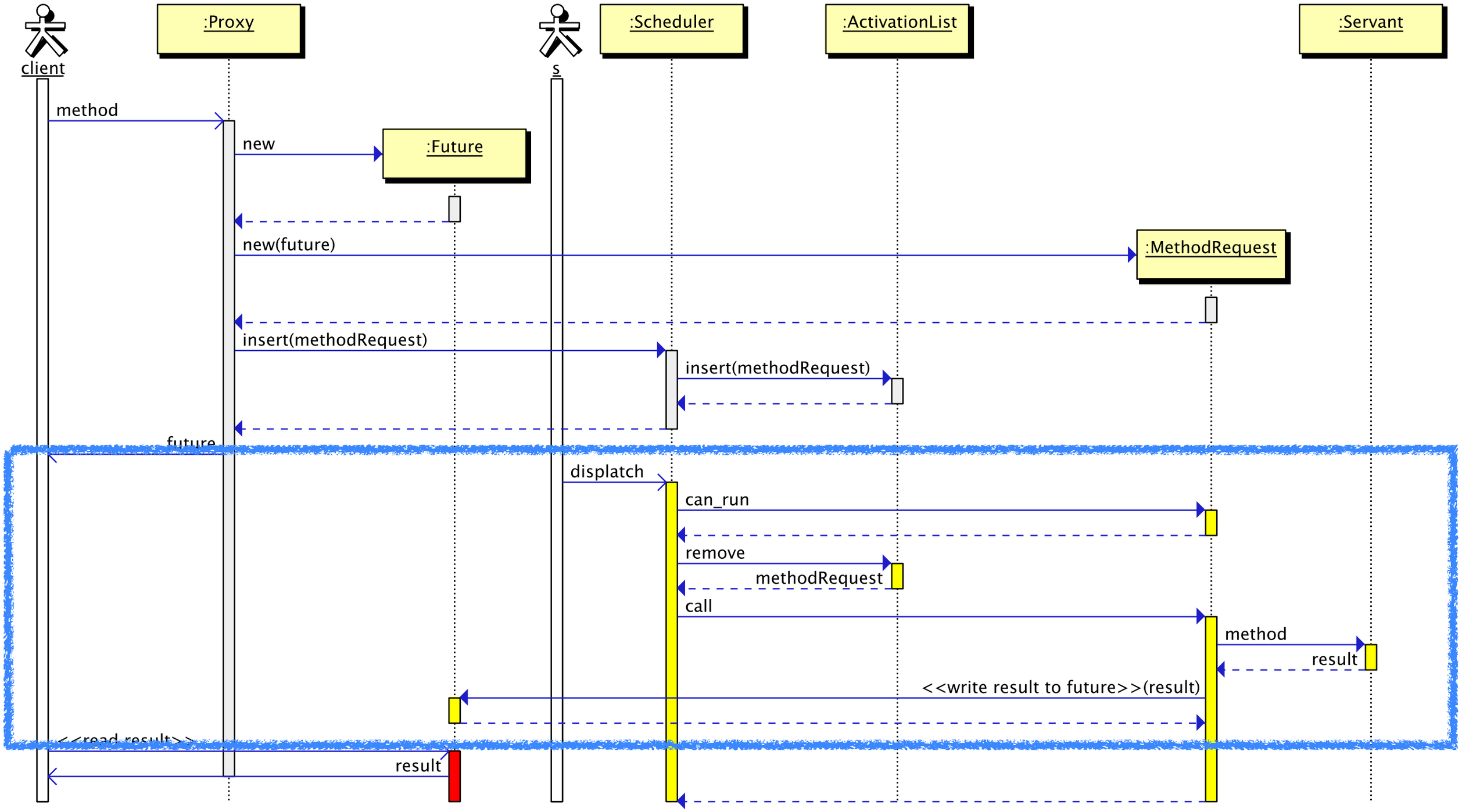
Method Scheduling



Method Scheduling (cont)

- A client calls a function on a proxy object
- The proxy creates a future which will be returned to the client (If the method returns a result)
- The proxy creates a method request which is told about the future
- The method request is then sent to the scheduler which places it in the activation list
- If the method is to return a result, the future is returned to the client

Method Dispatch



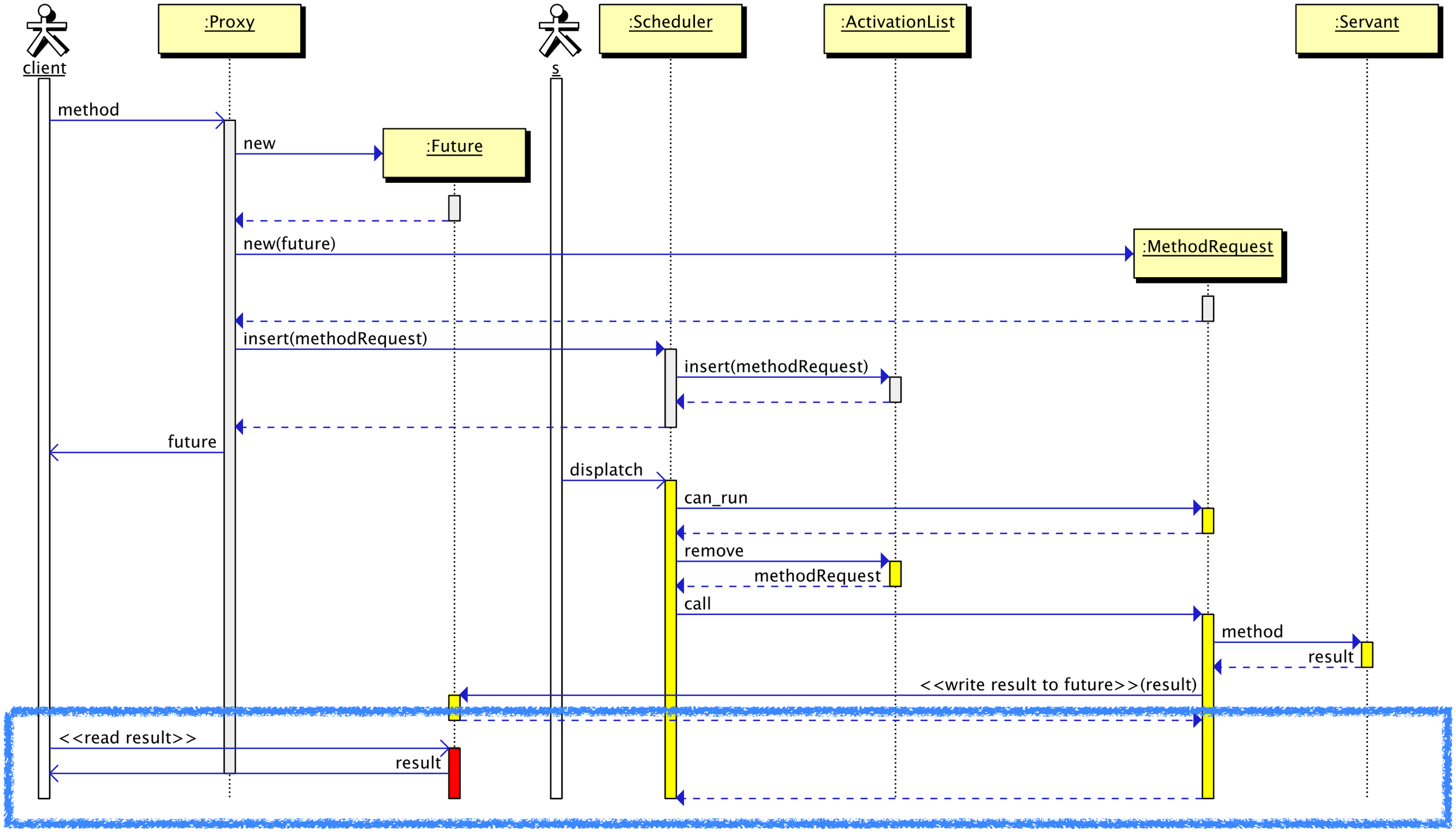
Method Dispatch (cont)

- The scheduler checks if the method request can be run
- If the method request can be run it is removed from the activation list and called
- The method request calls the method on the servant.
  - If the method has a result, the result is written to the associated future

Method Dispatch and Guards

- Methods have guards which can delay them from being called.
  - Before the scheduler can have the servant run a method, it checks if the method can be called
  - If a method cannot be called, the scheduler goes on to the next method to be run
- Example: Queue implemented as an Active Object
  - Enqueue cannot be called if the queue is full
  - Dequeue cannot be called if the queue is empty

Retrieving Results



Retrieving Results (cont)

- After the result has been written to the future the client can then retrieve it
- Reading the result is asynchronous of calling the method

Example Implementations / References

- Actor implementation in Python:
  http://jodal.github.com/pykka/
- Actors for Scala/Java:
  http://akka.io/
- http://www.erlang.org/
- "Prefer Using Active Objects Instead of Naked Threads" by Herb Sutter
  http://drdobbs.com/high-performance-computing/225700095
- "Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects" by Douglas C. Schmidt, et al.
  www1.cse.wustl.edu/~schmidt/POSA/POSA2/
  www.cs.wustl.edu/~schmidt/posa2.ppt
- http://en.wikipedia.org/wiki/Actor_model