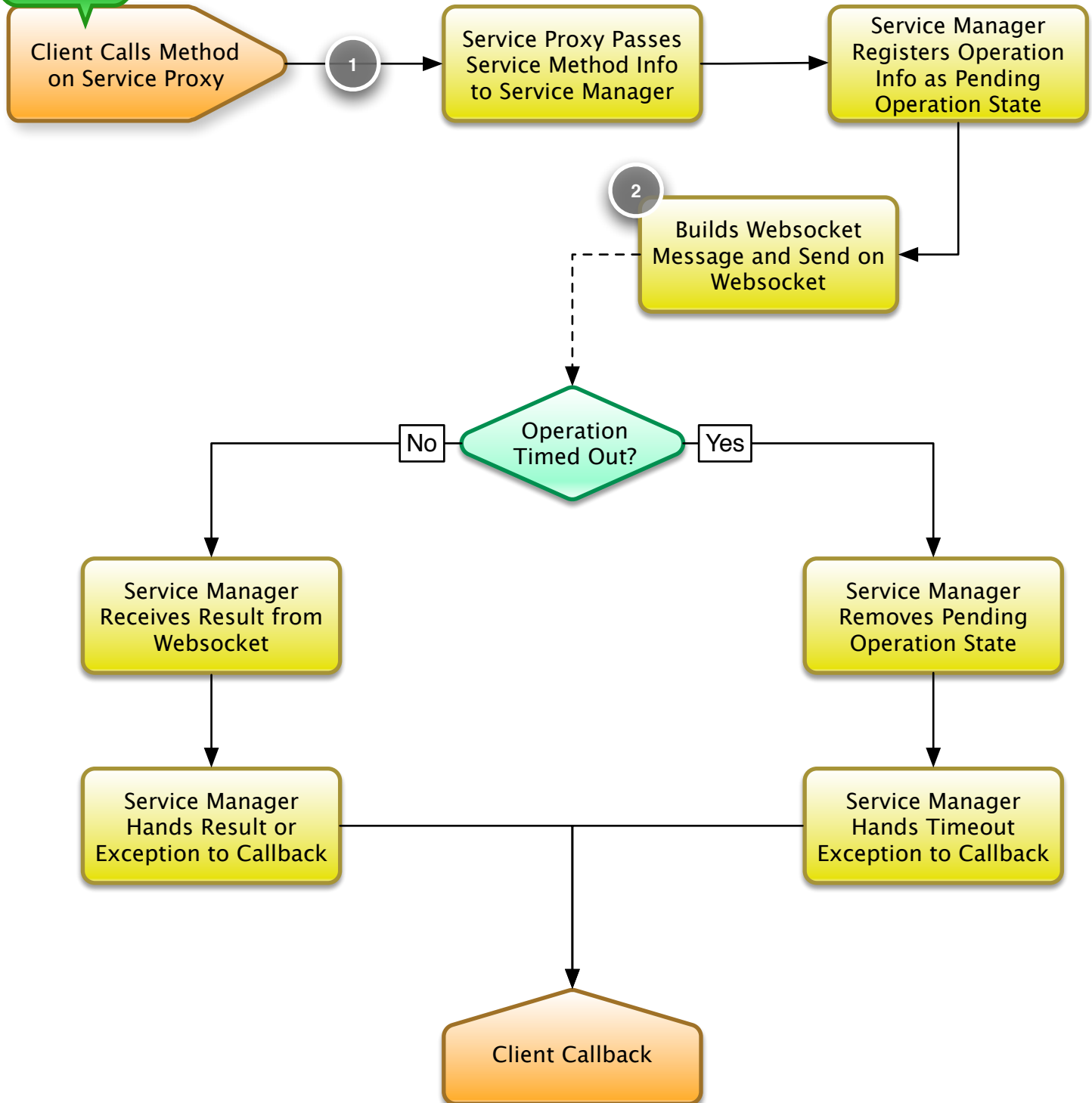


# Distributed Messaging Service: Client

Start Here



1 Every service proxy methods returns void and provides an additional argument for the caller to specify a callback. These callbacks handle successes & failures.

2 Message is 3 part binary message:

1. Length of header in bytes (32 bit int)
2. Header (byte array, UTF8 JSON or XML)
3. Body (byte array, UTF8 JSON or XML)

# Distributed Messaging Service: Server

Start Here

Websocket Message

1

Parse Message

Push Header Object  
into Shared Queue

Push Body Bytes  
into Shared Map

2

Pull Body from Shared  
Map and Call  
Specified Service  
Method Using Body  
as Arguments

Execute Service  
Method and Return  
Result or Throw  
Exception

2

Route Header Object  
to Specific Service  
Queue Based on  
Header

Create Full Message  
for Return to Client  
and Push Message  
into Client Queue

Websocket Message

1

Pull Message from  
Queue and Send on  
Websocket

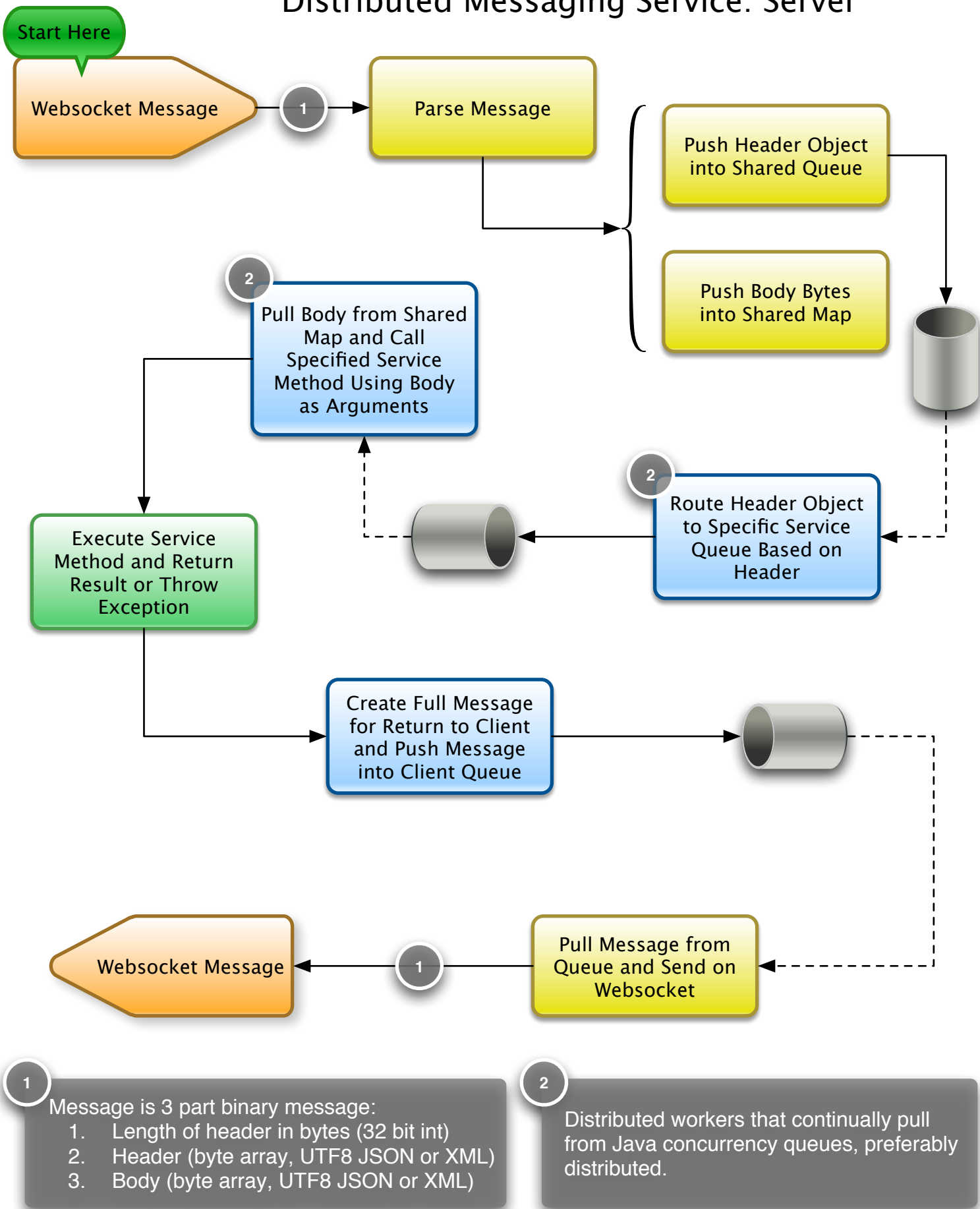
1

Message is 3 part binary message:

1. Length of header in bytes (32 bit int)
2. Header (byte array, UTF8 JSON or XML)
3. Body (byte array, UTF8 JSON or XML)

2

Distributed workers that continually pull from Java concurrency queues, preferably distributed.



# Distributed Messaging Service: Generators

## Java Service Interface

```
@MessageService
public interface MyMessageService
{
    @RemoteMessage
    public String sayHello(String aMyName)
    {
        ...
    }

    public String sayGoodbye(String aMyName)
    {
        ...
    }
}
```

Annotation Processor

## Generated

Service Proxy

Service Queue  
Workers

## Service Proxy

### @implementation MyMessageService

```
#pragma mark Message Service
```

```
- (void) sayHelloWithCallback:(Callback*) aCallback myName:(NSString*) aMyName
{
    MethodInfo* info = [MethodInfo info];
    info.channel = @"MyMessageService";
    info.destination = @"sayHello#java.lang.String";
    info.args = [NSArray arrayWithObjects:aMyName];
    info.callback = aCallback;
    [self.manager callWithInfo:info];
}
```

```
@end
```

## Service Queue Workers

```
public class MessageServiceWorker extends ServiceWorker
{
}
```

# Distributed Messaging Service: Topology

