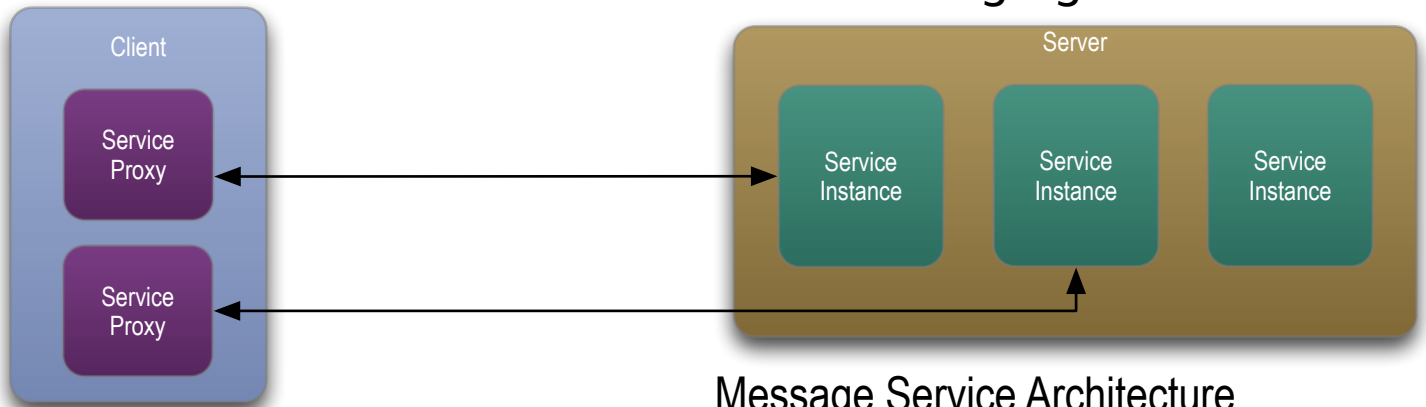
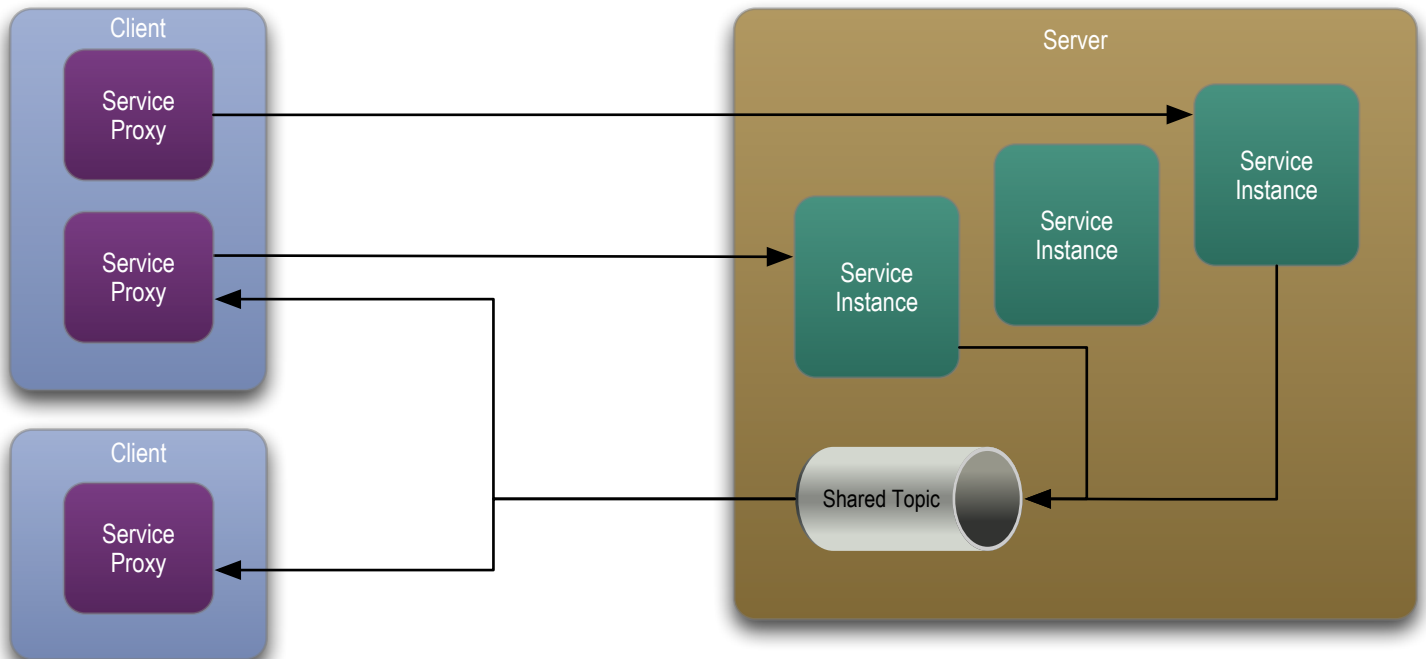


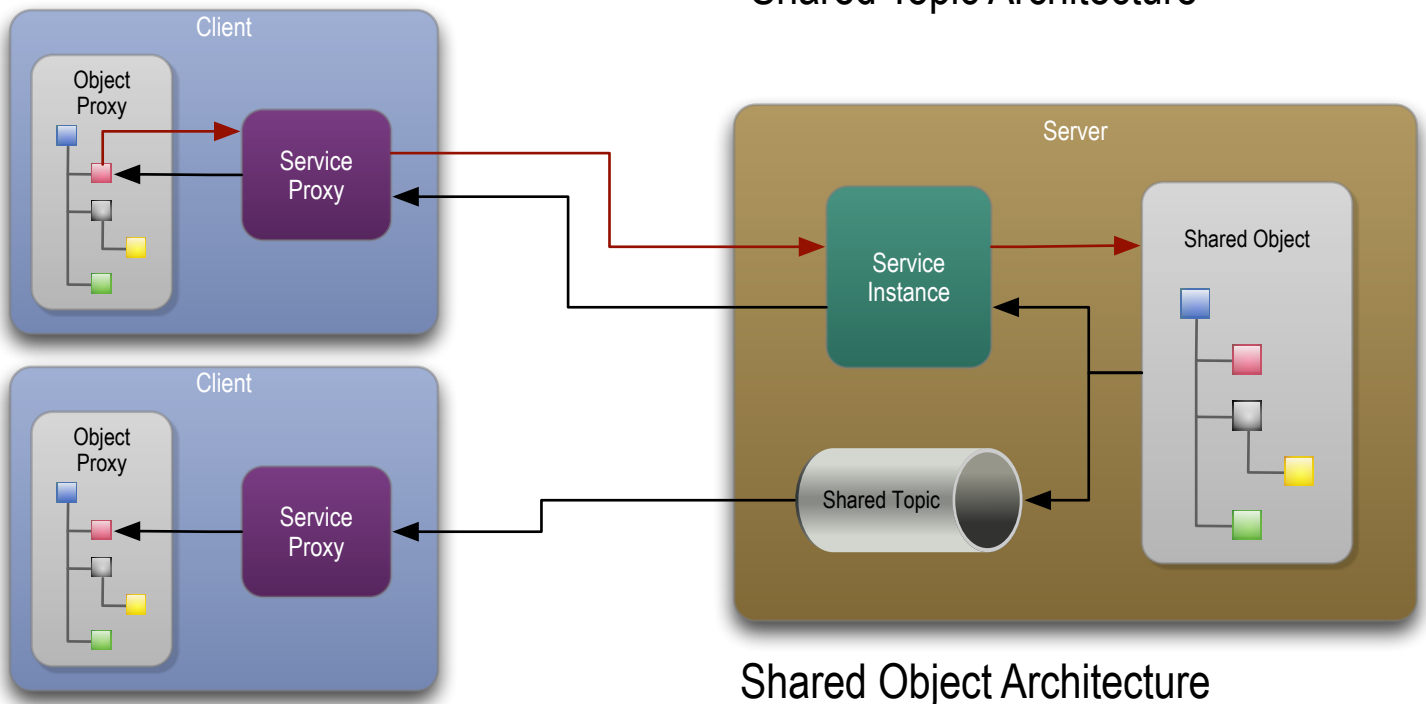
Distributed Messaging Framework



Message Service Architecture

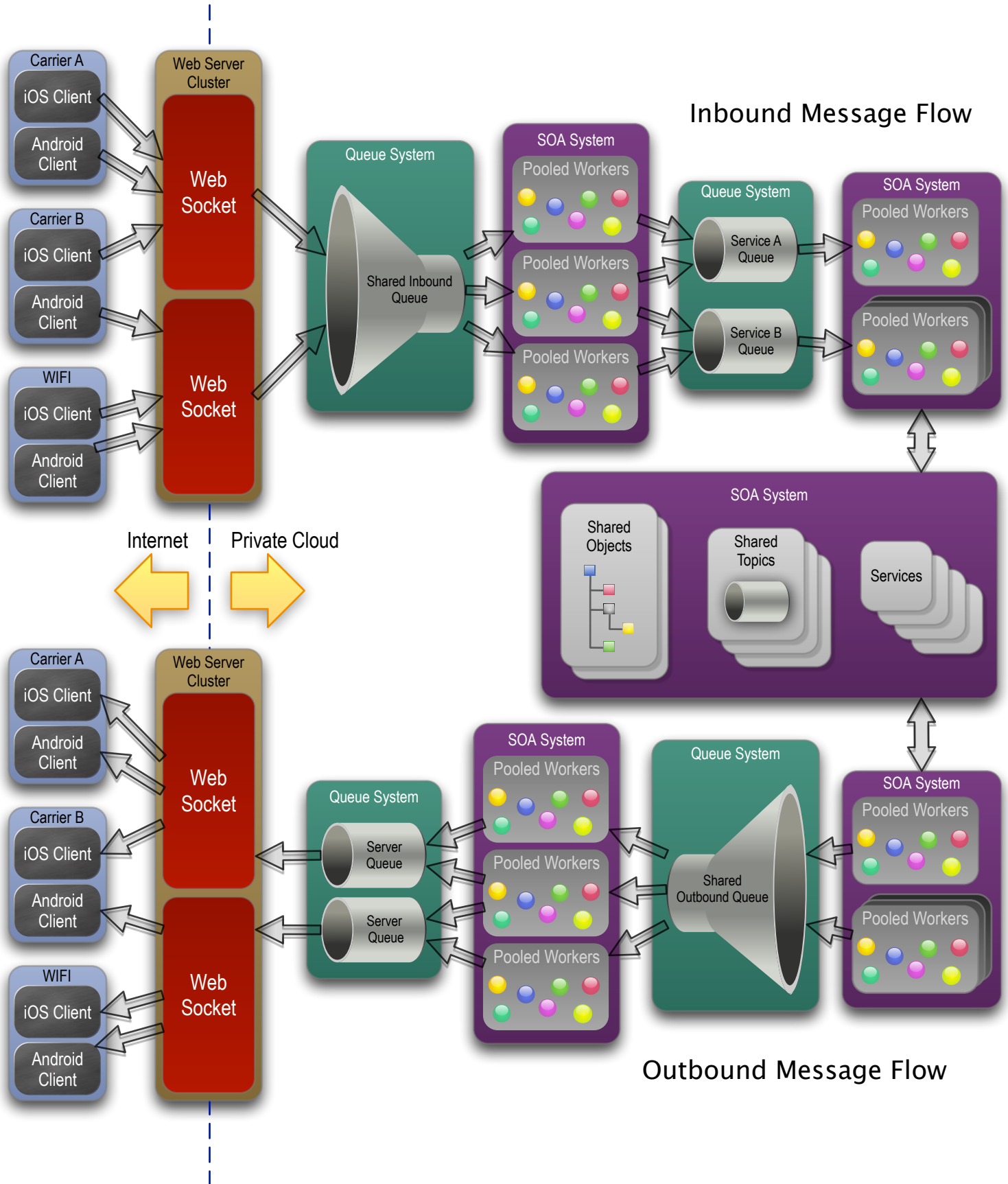


Shared Topic Architecture



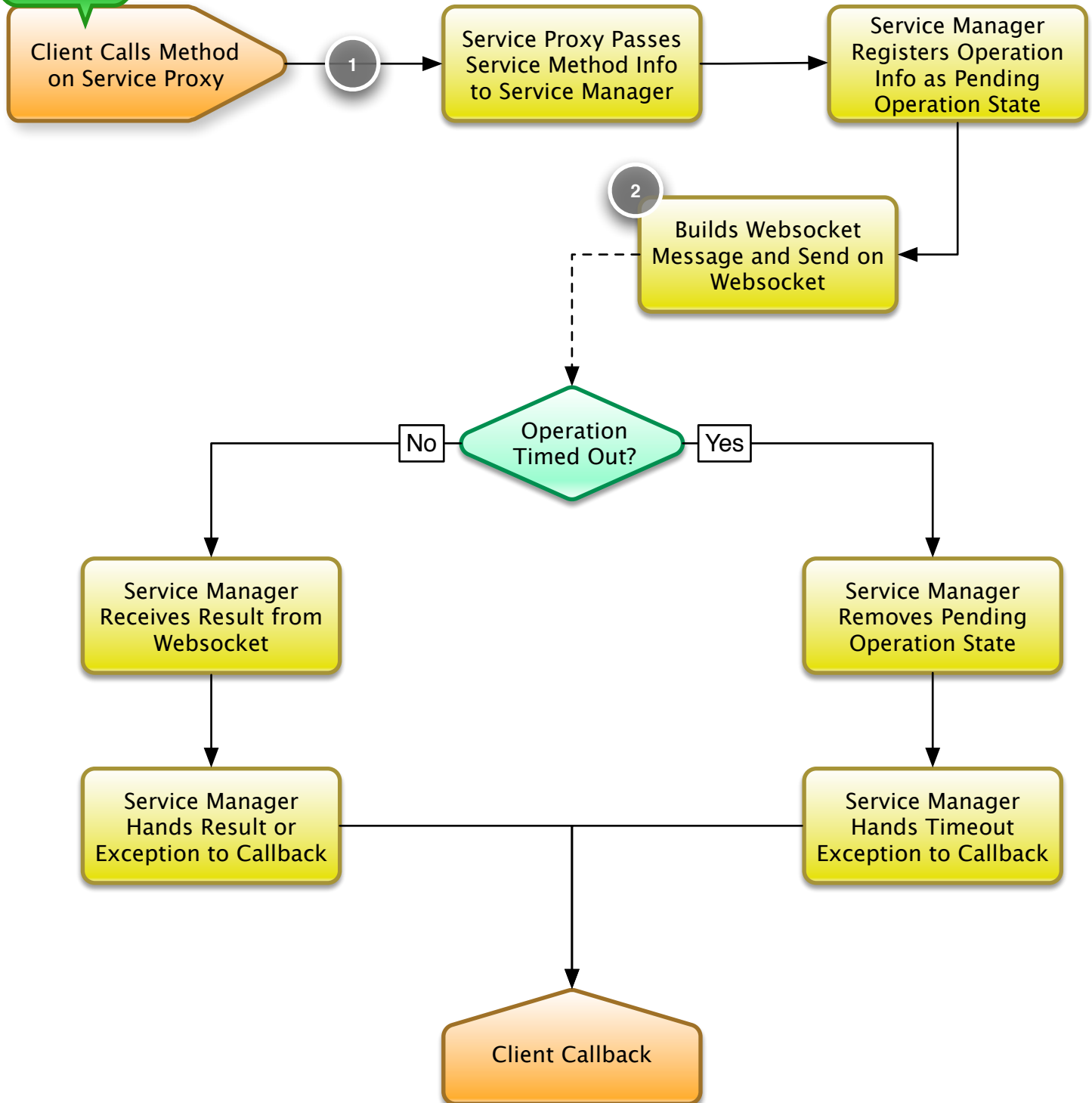
Shared Object Architecture

Distributed Messaging Service: Topology



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 4 part binary message:
1. Length of header in bytes (16 bit int)
2. Type of Data Encoding/Serialization
3. Header (byte array, UTF8 JSON or XML)
4. 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 4 part binary message:
1. Length of header in bytes (16 bit int)
 2. Type of Data Encoding/Serialization
 3. Header (byte array, UTF8 JSON or XML)
 4. 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 Proxy

@implementation MyMessageService

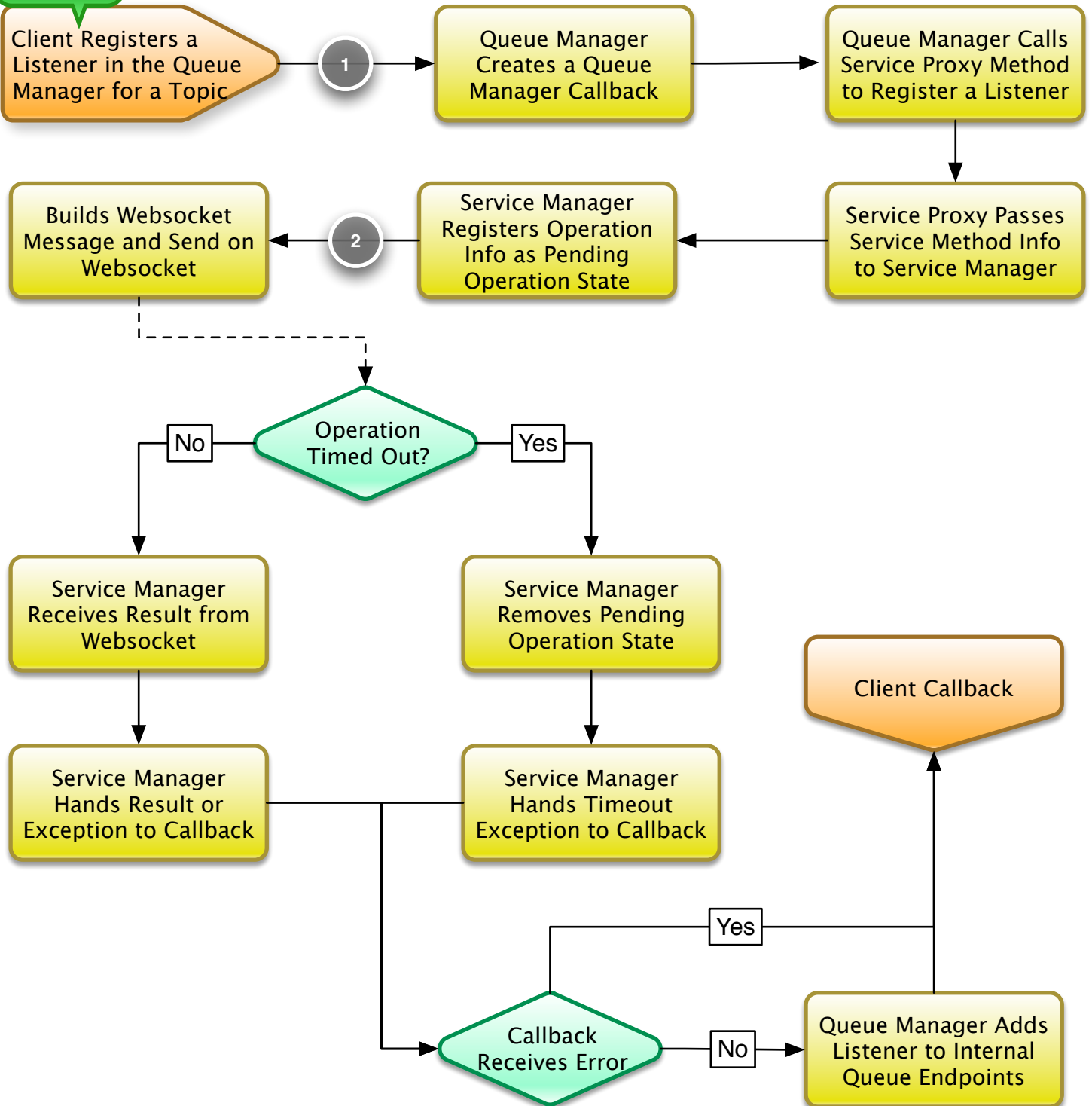
#pragma mark Message Service

```
- (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

Shared Topics: Client Registers a Listener

Start Here



1 Listener registration methods return void and provide an additional argument for the caller to specify a callback to handle registration successes & failures.

2 Message is 4 part binary message:
1. Length of header in bytes (16 bit int)
2. Type of Data Encoding/Serialization
3. Header (byte array, UTF8 JSON or XML)
4. Body (byte array, UTF8 JSON or XML)

Shared Topic: Server Registers a Listener

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

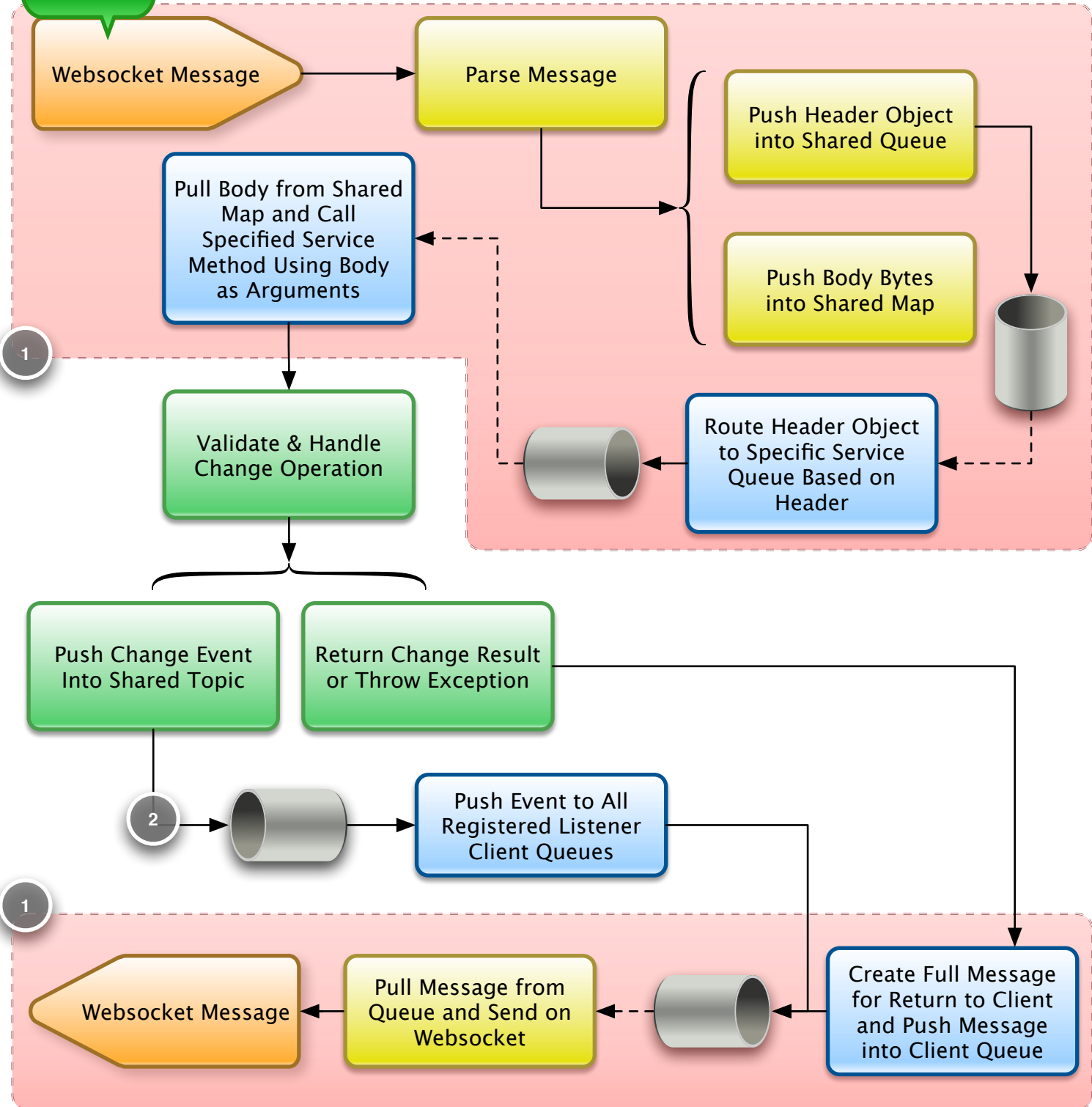
Pull Message from
Queue and Send on
Websocket

- 1 Message is 4 part binary message:
1. Length of header in bytes (16 bit int)
 2. Type of Data Encoding/Serialization
 3. Header (byte array, UTF8 JSON or XML)
 4. Body (byte array, UTF8 JSON or XML)

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

Shared Topic: Server Sends an Event

Start Here



1 System uses a service to manage a shared object. The service controls all access and changes to the object. This service uses the existing messaging service framework.

2 Changes to a shared object are returned as a part of the result and as a separate event sent through a shared topic to the remaining listeners.

Shared Objects: Client Changes a Property

Start Here

Client Calls Mutator
on Object Proxy

1

Object Proxy Calls
Change Method on
Service Proxy

2

Service Proxy Passes
Service Method Info
to Service Manager

Builds Websocket
Message and Send on
Websocket

Service Manager
Registers Operation
Info as Pending
Operation State

Operation
Timed Out?

No

Yes

Service Manager
Receives Result from
Websocket

Service Manager
Removes Pending
Operation State

Service Manager
Hands Result or
Exception to Callback

Service Manager
Hands Timeout
Exception to Callback

Service Callback
Delegates Response
to Object Proxy

Object Proxy Callback

Object Proxy Hands
Result or Exception
to Callback

Object Proxy Fires
Change Event to All
Appropriate Listeners

Object Proxy Updates
Property to Server
Value and Version

Property Change
Listener

1

Every object proxy mutator method returns void and provides an additional argument for the caller to specify a callback. These callbacks handle successes & failures.

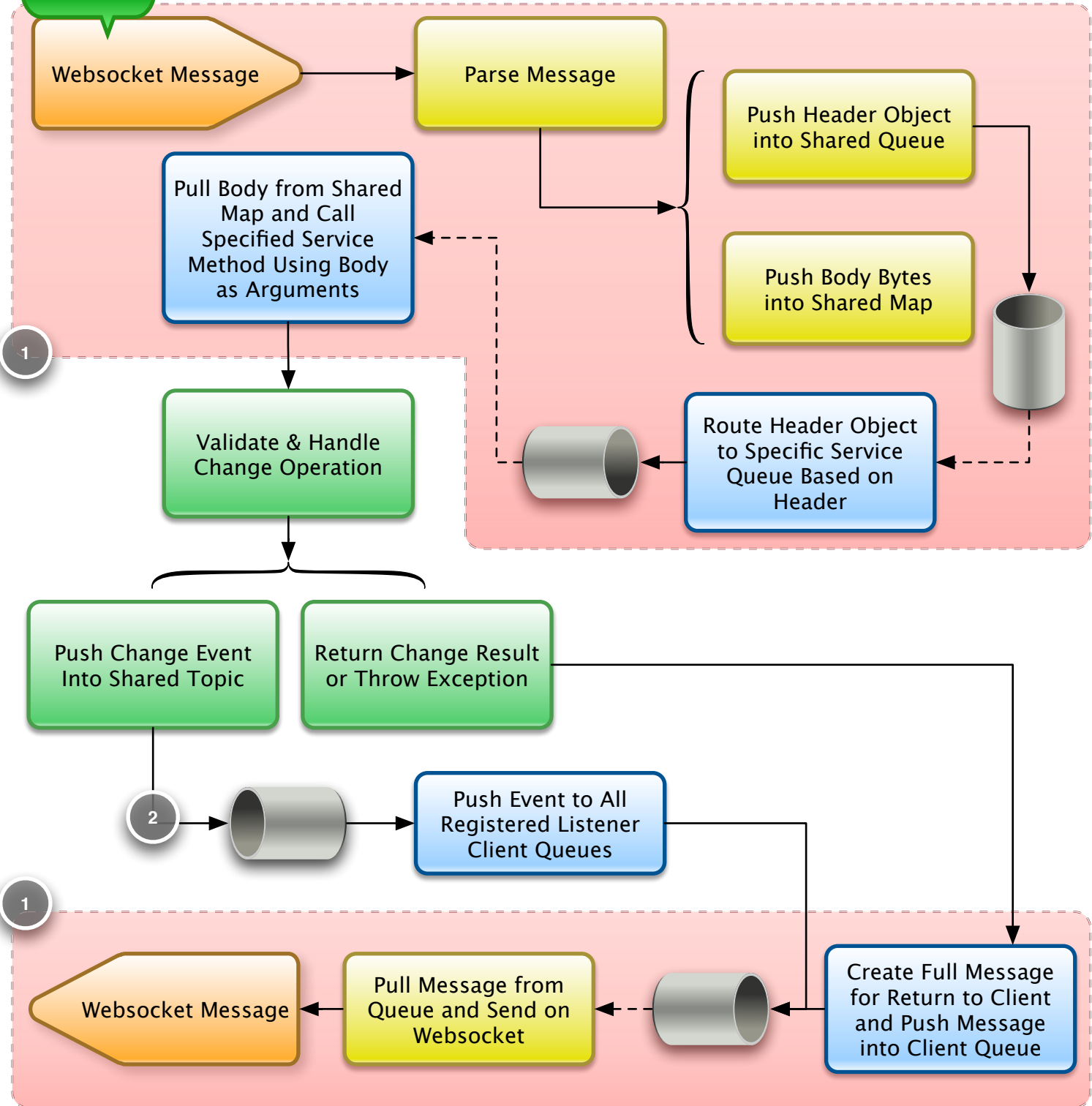
2

Property Change Method Requires:

1. Property Version Number
2. New Value
3. Shared Object UUID

Shared Objects: Server Changes a Property

Start Here



1 System uses a service to manage a shared object. The service controls all access and changes to the object. This service uses the existing messaging service framework.

2 Changes to a shared object are returned as a part of the result and as a separate event sent through a shared topic to the remaining listeners.

Shared Object: Generators

Java Service Interface

```
@SharedObject  
public interface MySharedObject  
{  
    public String getName();  
    public void setName(String aName);  
  
    ...  
}
```

Annotation Processor

Generated

Object Proxy

Object Proxy

@implementation MyMessageService

```
#pragma mark My Shared Object  
- (void) setName:(Callback*) aCallback value:(NSString*) aValue  
{  
    [serviceProxy setProperty:@"name" value:aValue callback:aCallback];  
}  
  
@end
```