Android Services & Local IPC: The Broker Pattern (Part 2)

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

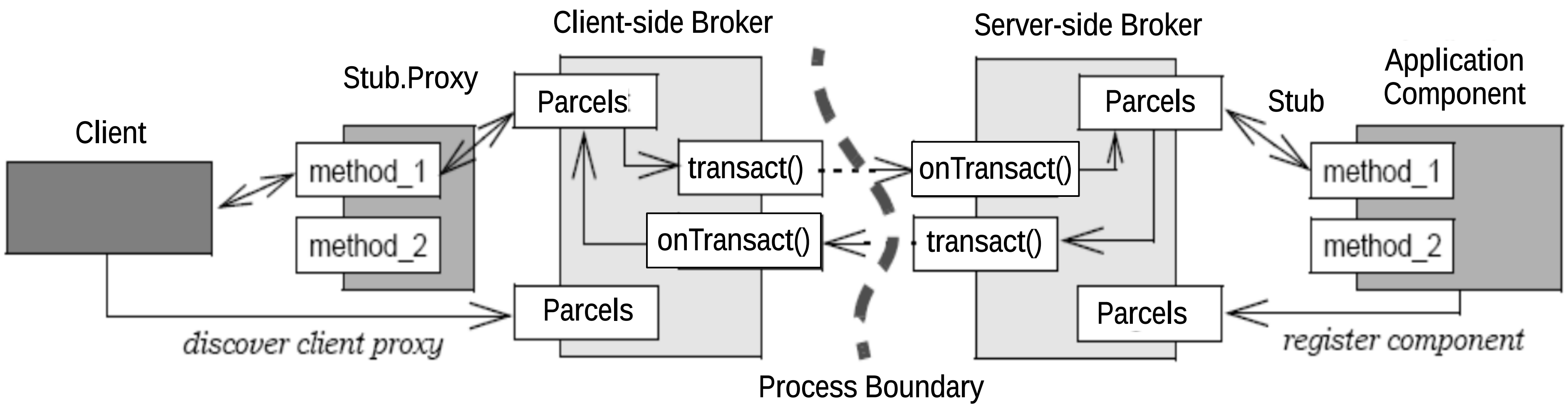
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Module

- Understand how the *Broker* pattern is applied in Android



Broker POSA1 Architectural Pattern

Implementation

- Define an invocation interface
  - Requestor's invocation interface allows clients to construct & send requests

```
public class Binder
       implements IBinder {
  ...
  public final boolean
    transact(int code,
             Parcel data,
             Parcel reply,
             int flags) ... {
    if (data != null)
      data.setDataPosition(0);
    boolean r = onTransact(code,
                           data,
                           reply,
                           flags);
    if (reply != null)
      reply.setDataPosition(0);
    return r;
  }
```

frameworks/base/core/java/android/os/Binder.java has the source code

Broker POSA1 Architectural Pattern

Implementation

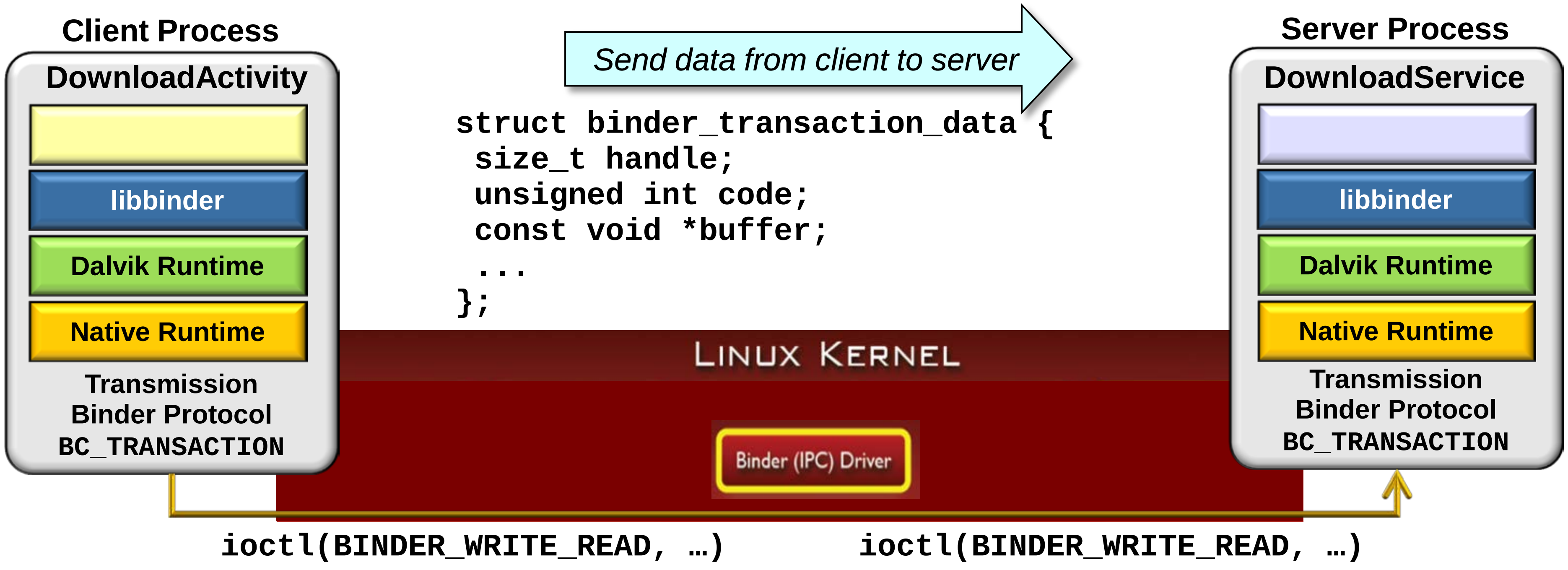
- Define an invocation interface
- Select & implement the marshaler
  - See the *Proxy* discussion for details

```
private static class Proxy
        implements IDownload {
  public String downloadImage(
    String uri) ... {
  android.os.Parcel _data =
    android.os.Parcel.obtain();
  android.os.Parcel _reply =
     android.os.Parcel.obtain();
  _data.writeString(uri);
  mRemote.transact
    (Stub.TRANSACTION_downloadImage,
     _data, _reply, 0);
  _reply.readException();
  java.lang.String _result =
    _reply.readString();
  ...
  return _result;
  ...
```

Broker POSA1 Architectural Pattern

Implementation

- Define an invocation interface
- Select & implement the marshaler
- Select communication protocol
  - e.g., connection-oriented vs. connectionless



```
struct binder_transaction_data {
 size_t handle;
 unsigned int code;
 const void *buffer;
 ...
};
```

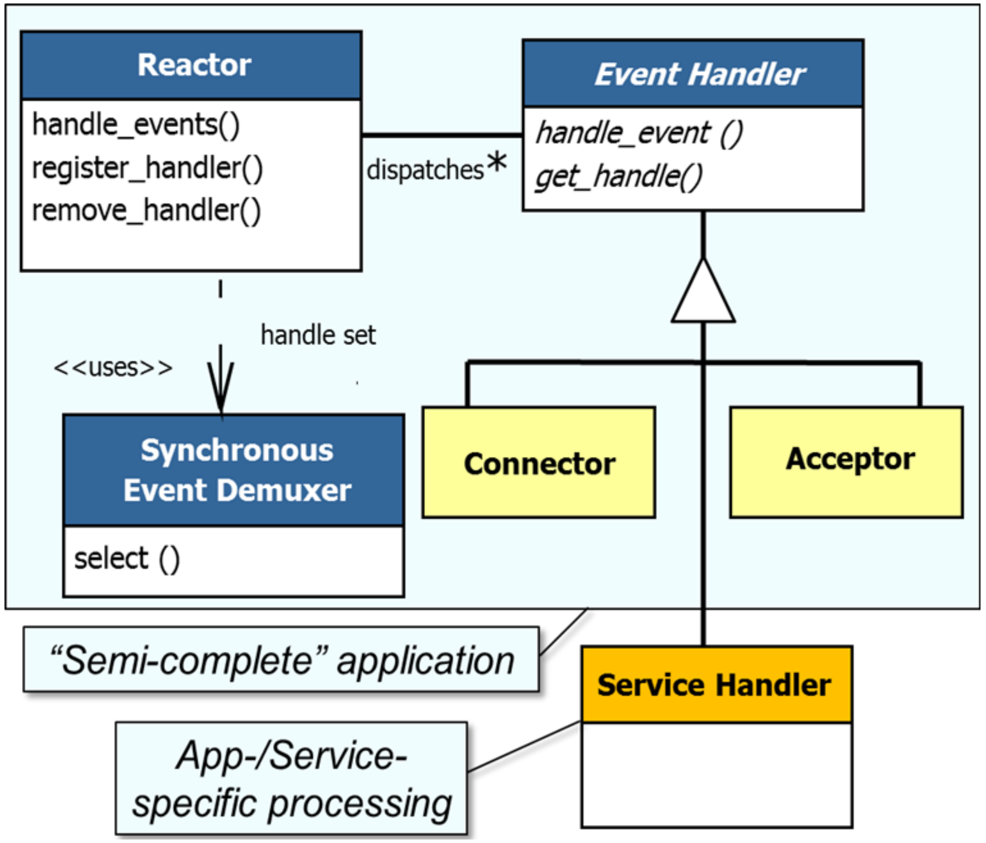
```
ioctl(BINDER_WRITE_READ, …)        ioctl(BINDER_WRITE_READ, …)
```

rts.lab.asu.edu/web_438/project_final/Talk%208%20AndroidArc_Binder.pdf

Broker POSA1 Architectural Pattern

Implementation

- Define an invocation interface
- Select & implement the marshaler
- Select communication protocol
- Implement network communication
  - e.g. use the *Acceptor/Connector* pattern to establish connections between requestor & dispatcher & *Reactor* for demxuing incoming requests & responses

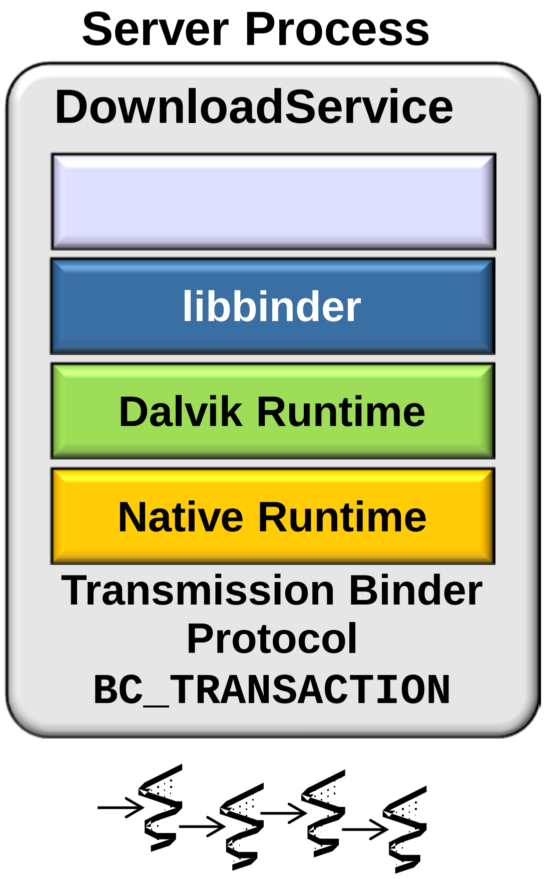


See www.dre.vanderbilt.edu/~schmidt/PDF/TAPOS.pdf for more info

Broker POSA1 Architectural Pattern

Implementation

- Define an invocation interface
- Select & implement the marshaler
- Select communication protocol
- Implement network communication
- Implement resource management
  - Connections between requestors & dispatchers can be reused & shared using the Caching & Pooling pattern, respectively



Broker POSA1 Architectural Pattern

Implementation

- Define an invocation interface
- Select & implement the marshaler
- Select communication protocol
- Implement network communication
- Implement resource management
- Define an registration interface
  - Provided by the dispatcher for the registration & unregistration of servants

```
public class Binder
       implements IBinder {
  ...
  public void attachInterface
               (IInterface owner,
               String descriptor)
    {
      mOwner = owner;
      mDescriptor = descriptor;
    }
  ...
```

frameworks/base/core/java/android/os/Binder.java has the source code

Broker POSA1 Architectural Pattern

Implementation

- Define an invocation interface
- Select & implement the marshaler
- Select communication protocol
- Implement network communication
- Implement resource management
- Define an registration interface
- Provide a mechanism to reference servants
  - To perform requests on remote objects, represented by servants, the clients have to obtain references to those remote objects

```
public class Service extends
                  ... {
  ...
   public abstract IBinder
      onBind(Intent intent);
   ...
}
```

Factory method that returns a reference to a Binder object

frameworks/base/core/java/android/app/Service.java

Broker POSA1 Architectural Pattern

Implementation

- Define an invocation interface
- Select & implement the marshaler
- Select communication protocol
- Implement network communication
- Implement resource management
- Define an registration interface
- Provide a mechanism to reference servants
  - To perform requests on remote objects, represented by servants, the clients have to obtain references to those remote objects

```
public class Service extends
                  ... {
  ...
   public abstract IBinder
      onBind(Intent intent);
   ...
}
```

```
interface ServiceConnection {
  public void
    onServiceConnected
           (ComponentName name,
            IBinder service);
  ...
}
```

Hook method to pass Binder reference back to client

frameworks/base/core/java/android/content/ServiceConnection.java

Broker POSA1 Architectural Pattern

Implementation

- Define an invocation interface
- Select & implement the marshaler
- Select communication protocol
- Implement network communication
- Implement resource management
- Define an registration interface
- Provide a mechanism to reference servants
- Implement the mechanism to transform request messages into upcalls on servants

```
public static abstract class Stub
       extends android.os.Binder
       implements IDownload {
  public boolean onTransact
           (int code,
            android.os.Parcel data,
            android.os.Parcel reply,
            int flags) ... {
    switch (code) {
    case TRANSACTION_downloadImage:
      ...
      java.lang.String _arg0 =
        data.readString();
      java.lang.String _result =
        this.downloadImage(_arg0);
      ...
```

Broker POSA1 Architectural Pattern

Implementation

- Define an invocation interface
- Select & implement the marshaler
- Select communication protocol
- Implement network communication
- Implement resource management
- Define an registration interface
- Provide a mechanism to reference servants
- Implement the mechanism to transform request messages into upcalls on servants
- Decide if/how to support asynchrony

```
interface IDownload {
  oneway void setCallback
    (in IDownloadCallback
        callback);
}

interface IDownloadCallback {
  oneway void sendPath
    (in String path);
}
```

Broker POSA1 Architectural Pattern

Implementation

- Define an invocation interface
- Select & implement the marshaler
- Select communication protocol
- Implement network communication
- Implement resource management
- Define an registration interface
- Provide a mechanism to reference servants
- Implement the mechanism to transform request messages into upcalls on servants
- Decide if/how to support asynchrony
- Optimize local invocations

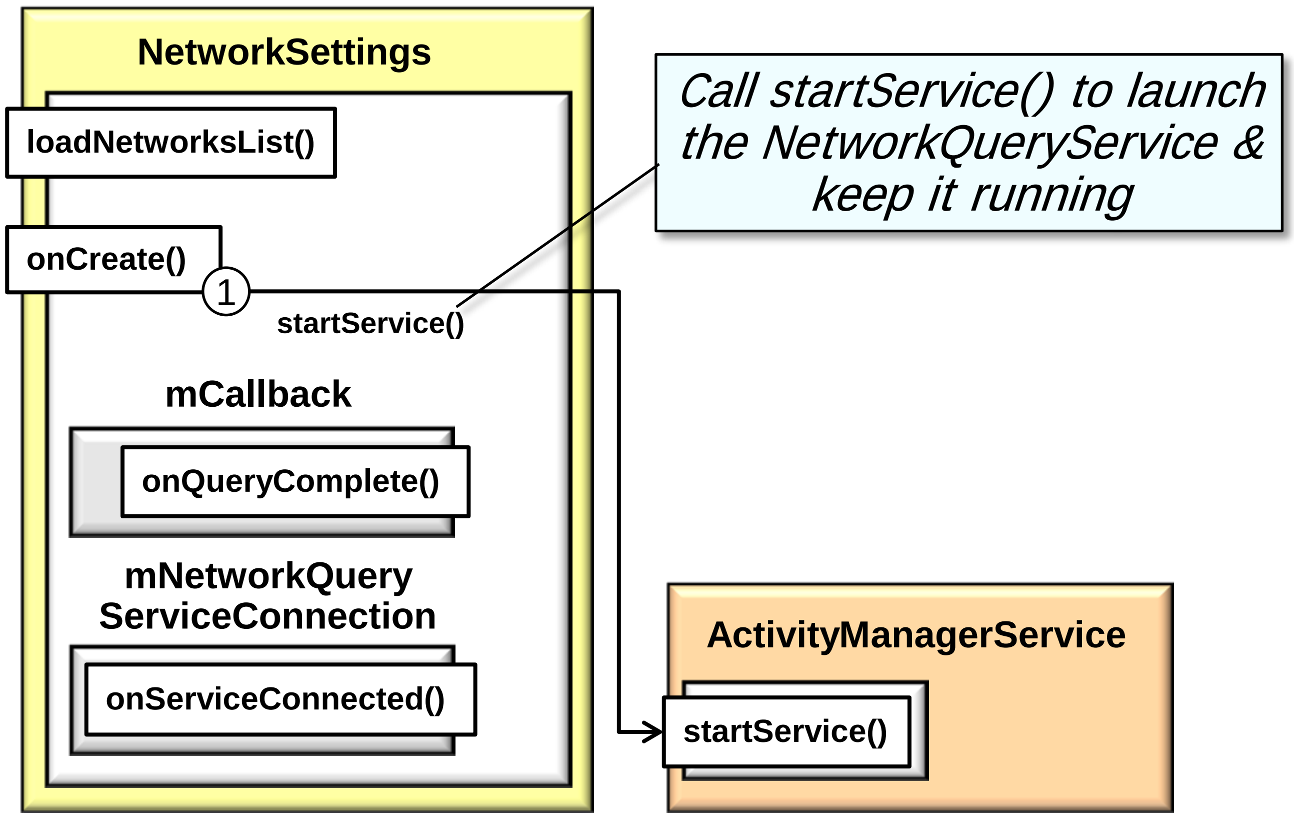
```
public static abstract class Stub
       extends android.os.Binder
       implements IDownload {
  ...
  public static IDownload
    asInterface
     (android.os.IBinder obj) {
    if ((obj==null)) return null;
    android.os.IInterface iin =
     (android.os.IInterface)
    obj.queryLocalInterface
                    (DESCRIPTOR);
    if(((iin != null) &&
      (iin instanceof IDownload)))
      return ((IDownload)iin);
    return new IDownload.Stub.
                 Proxy(obj);
  }
```

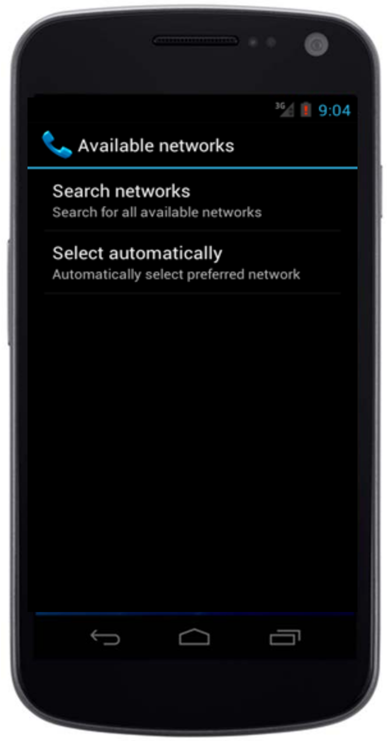
www.dre.vanderbilt.edu/~schmidt/PDF/COOTS-99.pdf has more info

Broker POSA1 Architectural Pattern

Applying the Broker pattern in Android

- The NetworkSettings Activity uses the *Activator* pattern to launch the NetworkQueryService to assist in querying the network for service availability

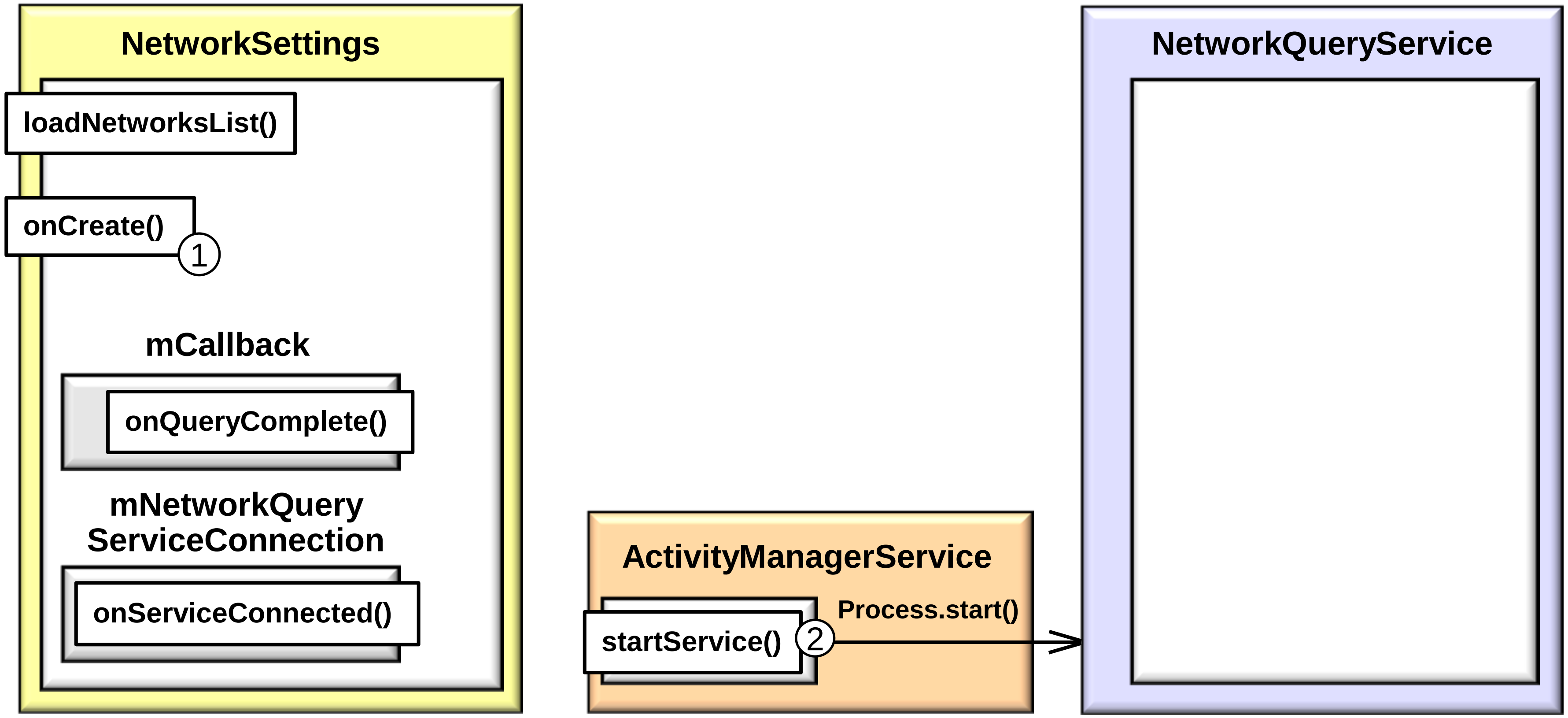



9:04
Available networks
Search networks
Search for all available networks
Select automatically
Automatically select preferred network

Broker POSA1 Architectural Pattern

Applying the Broker pattern in Android

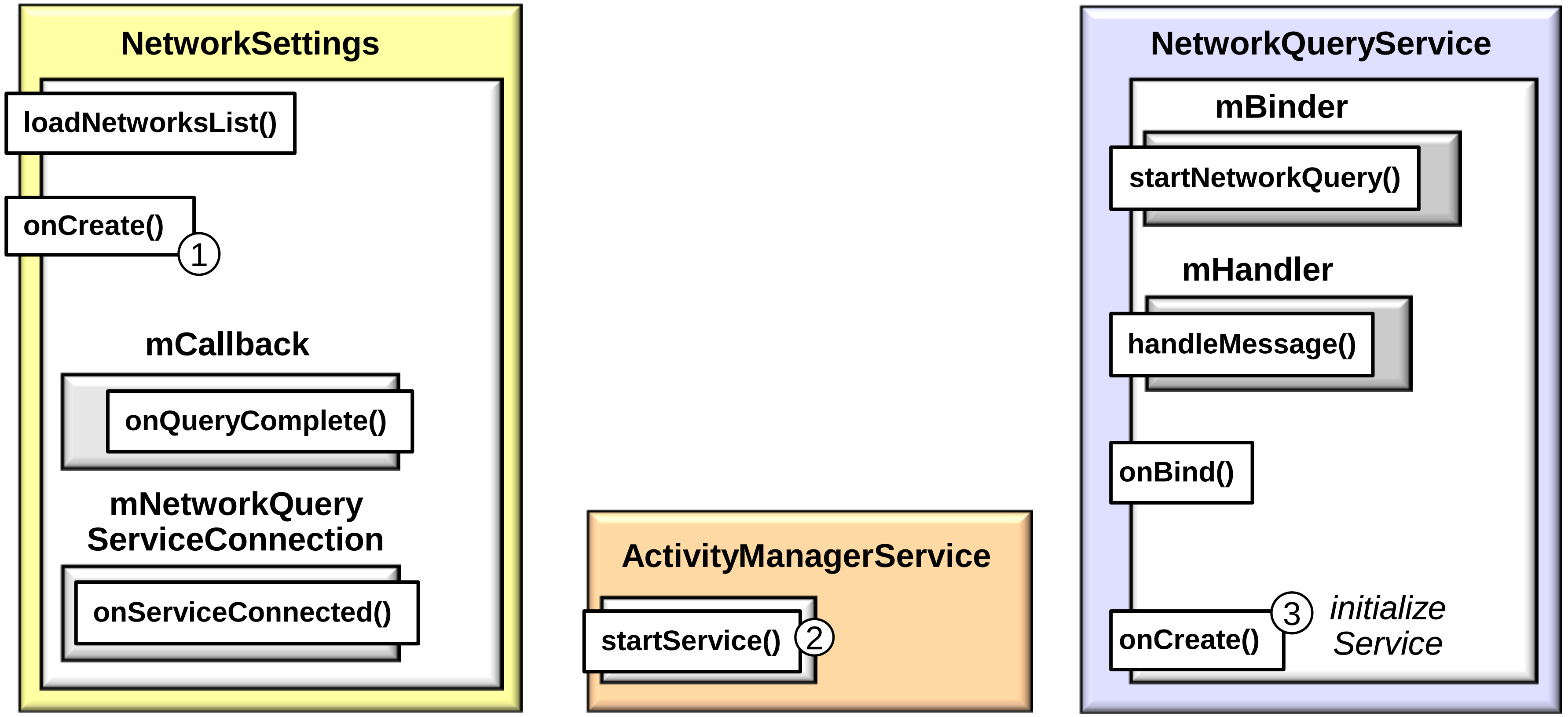
- The NetworkSettings Activity uses the *Activator* pattern to launch the NetworkQueryService to assist in querying the network for service availability



Broker POSA1 Architectural Pattern

Applying the Broker pattern in Android

- The NetworkSettings Activity uses the *Activator* pattern to launch the NetworkQueryService to assist in querying the network for service availability

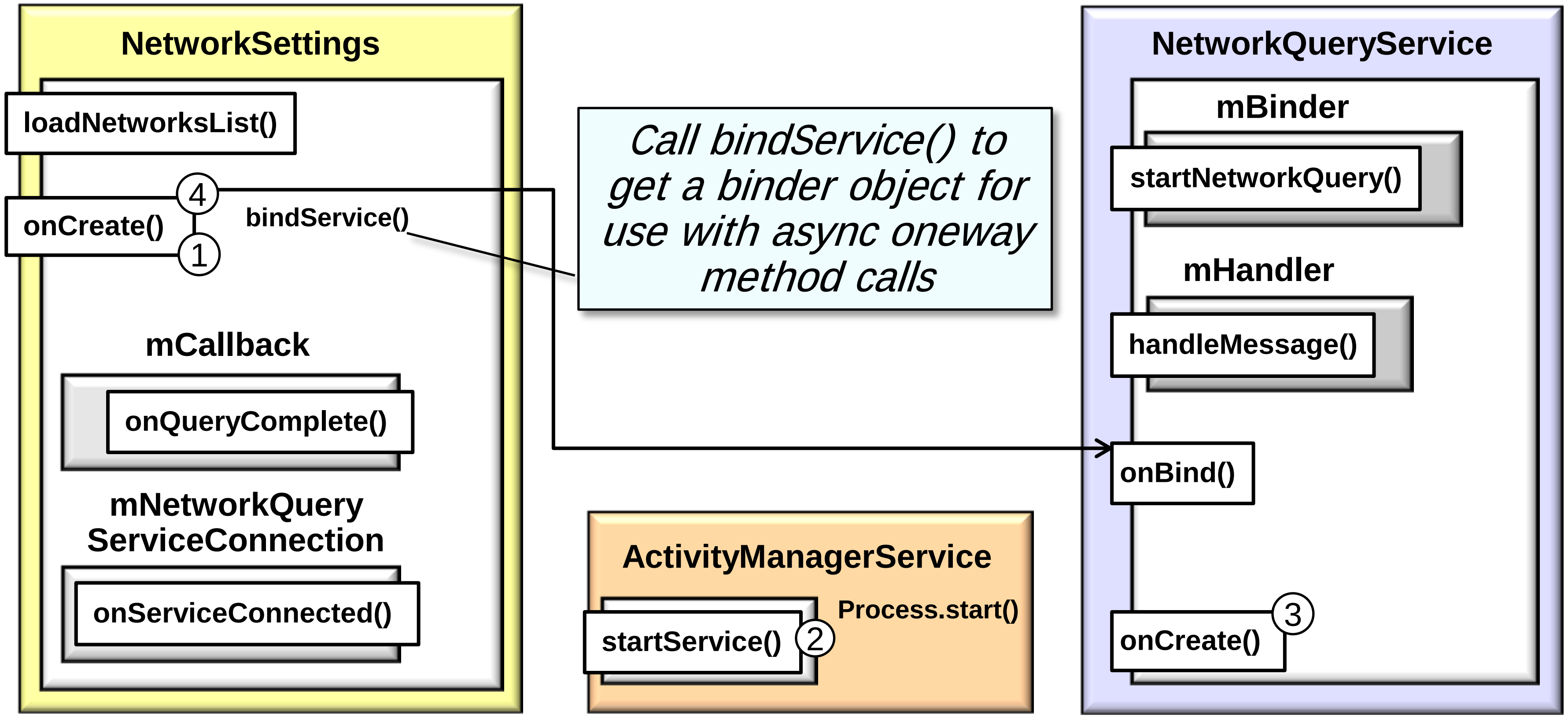


packages/apps/Phone/src/com/android/phone/NetworkQueryService.java has source

Broker POSA1 Architectural Pattern

Applying the Broker pattern in Android

- The NetworkSettings Activity uses the *Activator* pattern to launch the NetworkQueryService to assist in querying the network for service availability

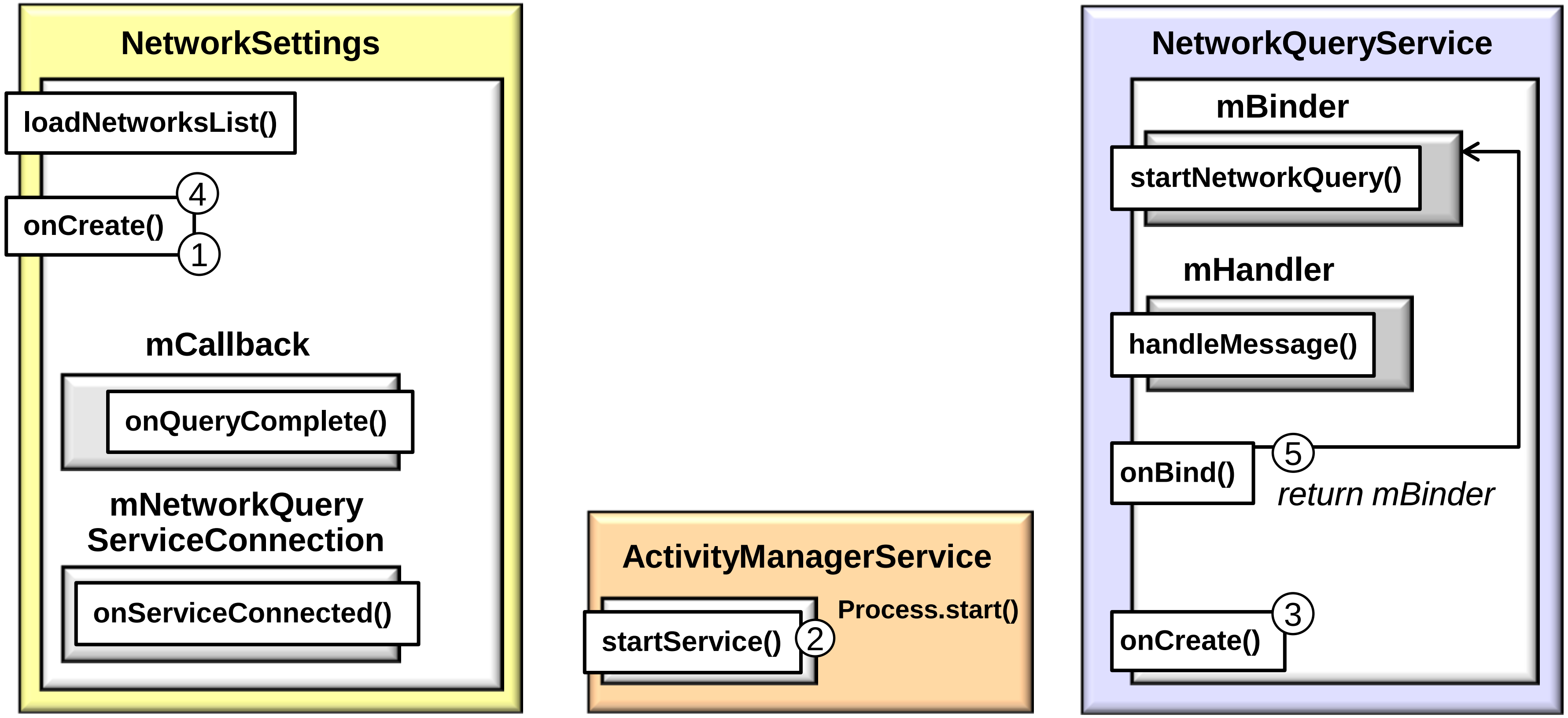


packages/apps/Phone/src/com/android/phone/NetworkSetting.java has source code

Broker POSA1 Architectural Pattern

Applying the Broker pattern in Android

- The NetworkSettings Activity uses the *Activator* pattern to launch the NetworkQueryService to assist in querying the network for service availability

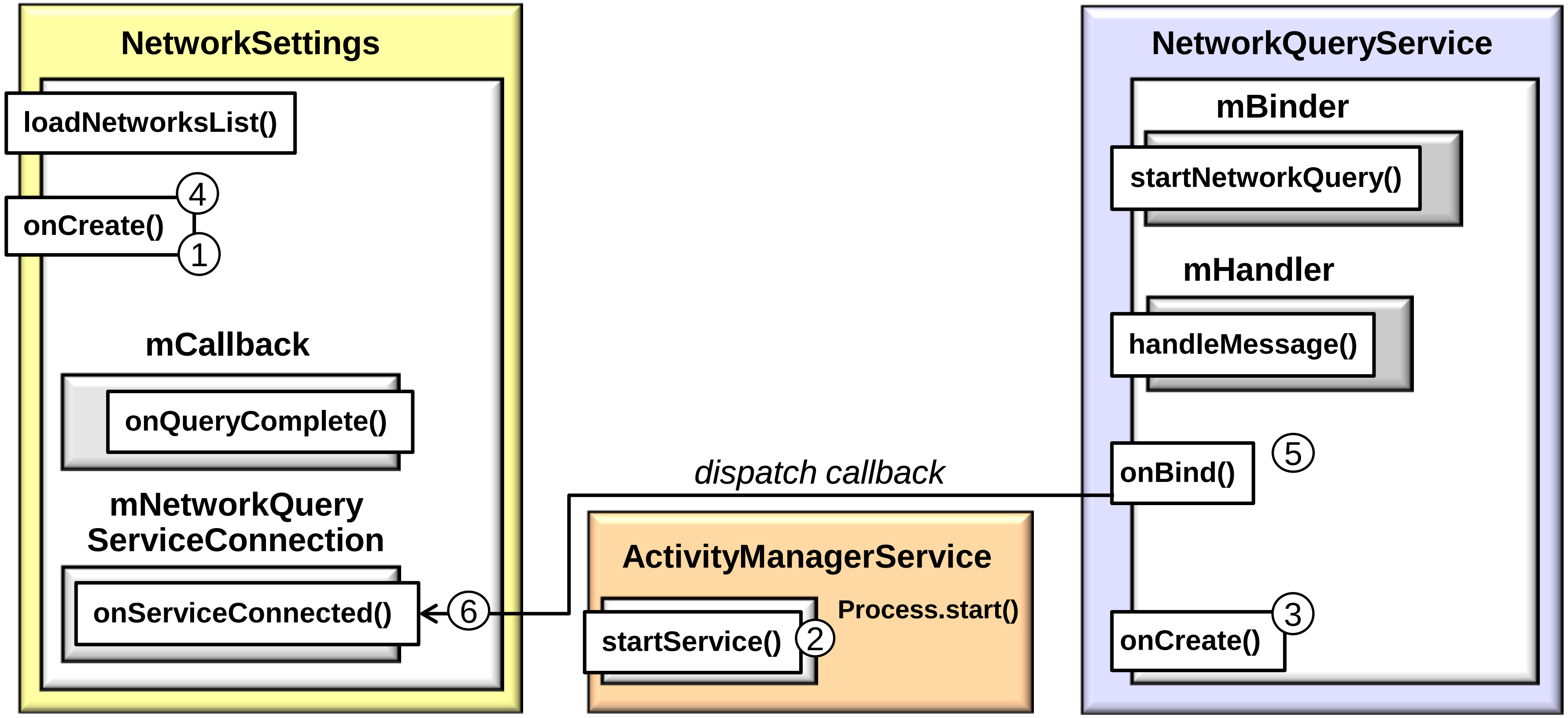


packages/apps/Phone/src/com/android/phone/NetworkQueryService.java has source

Broker POSA1 Architectural Pattern

Applying the Broker pattern in Android

- The NetworkSettings Activity uses the *Activator* pattern to launch the NetworkQueryService to assist in querying the network for service availability

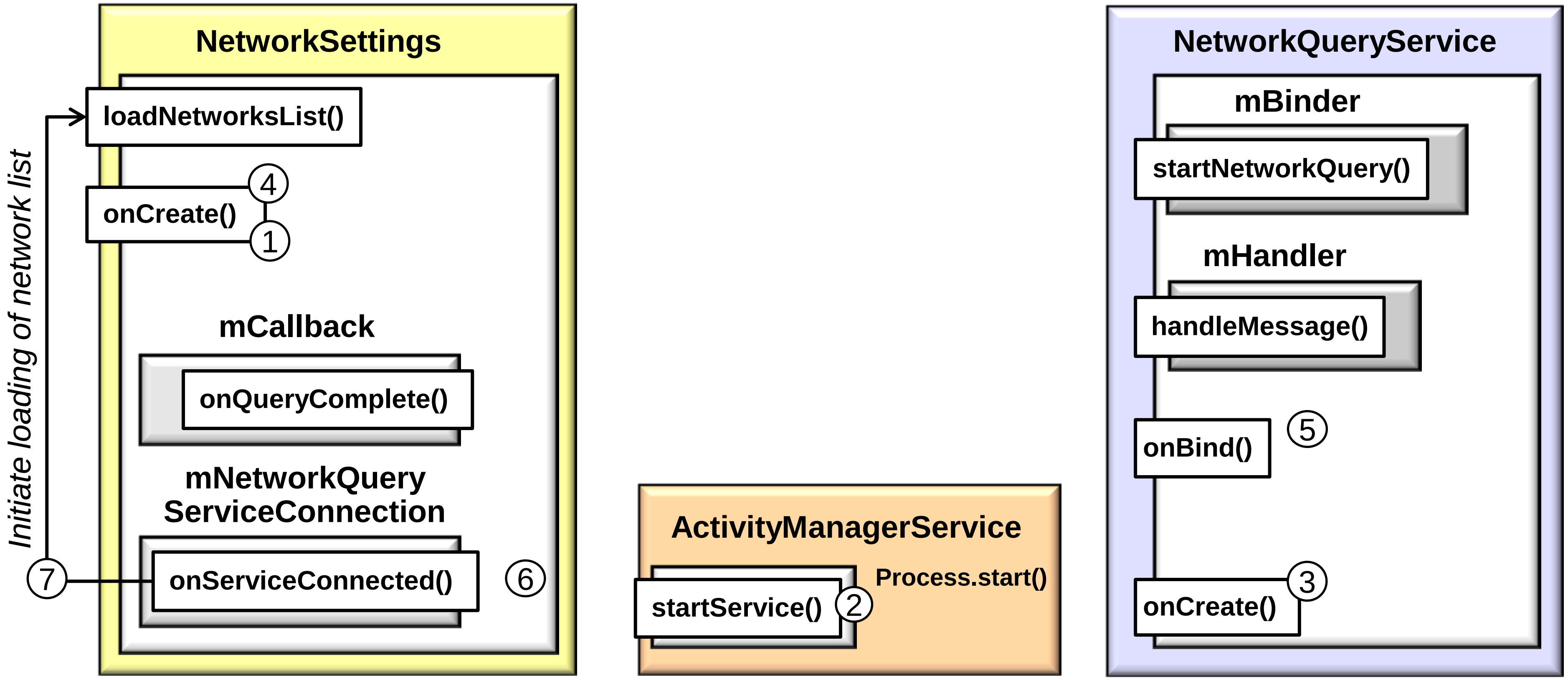


packages/apps/Phone/src/com/android/phone/NetworkSetting.java has source code

Broker POSA1 Architectural Pattern

Applying the Broker pattern in Android

- The NetworkSettings Activity uses the *Activator* pattern to launch the NetworkQueryService to assist in querying the network for service availability

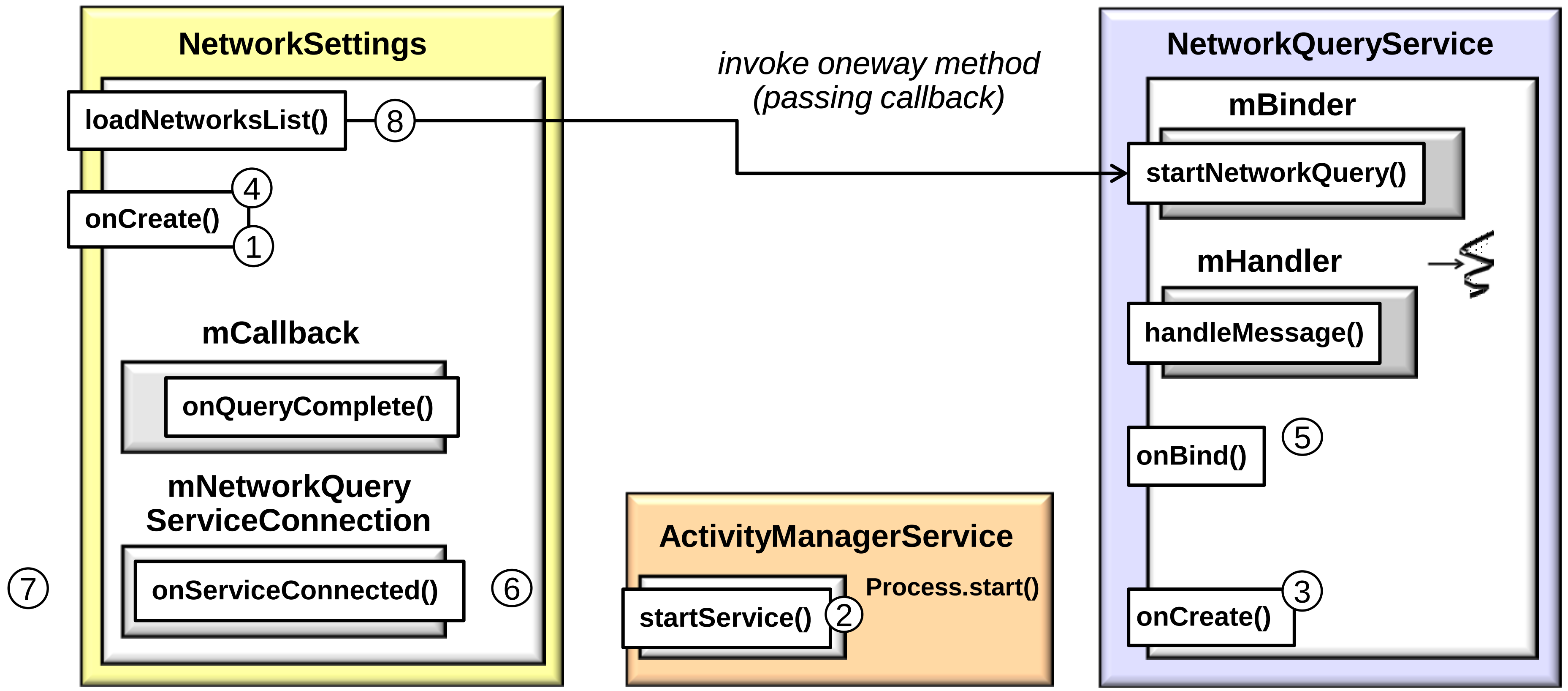


packages/apps/Phone/src/com/android/phone/NetworkSetting.java has source code

Broker POSA1 Architectural Pattern

Applying the Broker pattern in Android

- The NetworkSettings Activity uses the *Activator* pattern to launch the NetworkQueryService to assist in querying the network for service availability

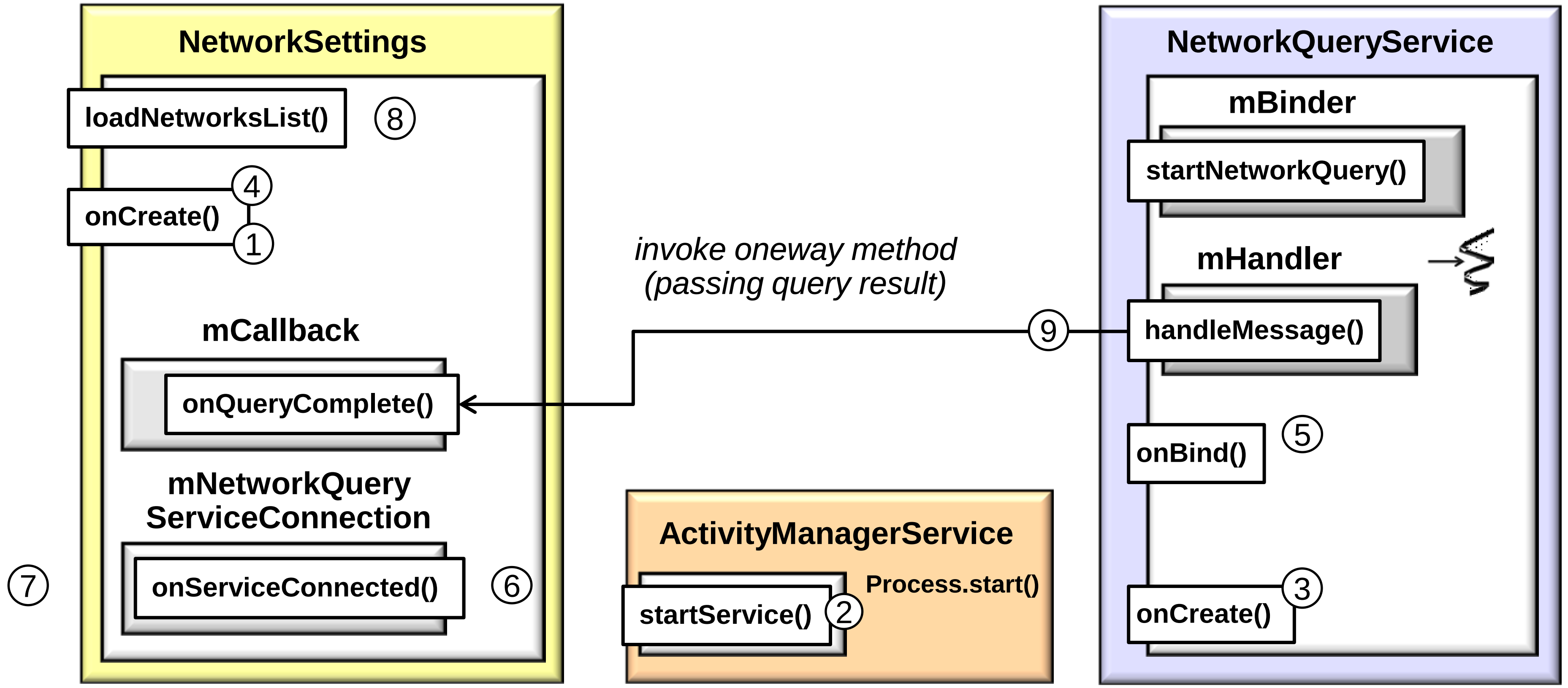


packages/apps/Phone/src/com/android/phone/NetworkQueryService.java has source

Broker POSA1 Architectural Pattern

Applying the Broker pattern in Android

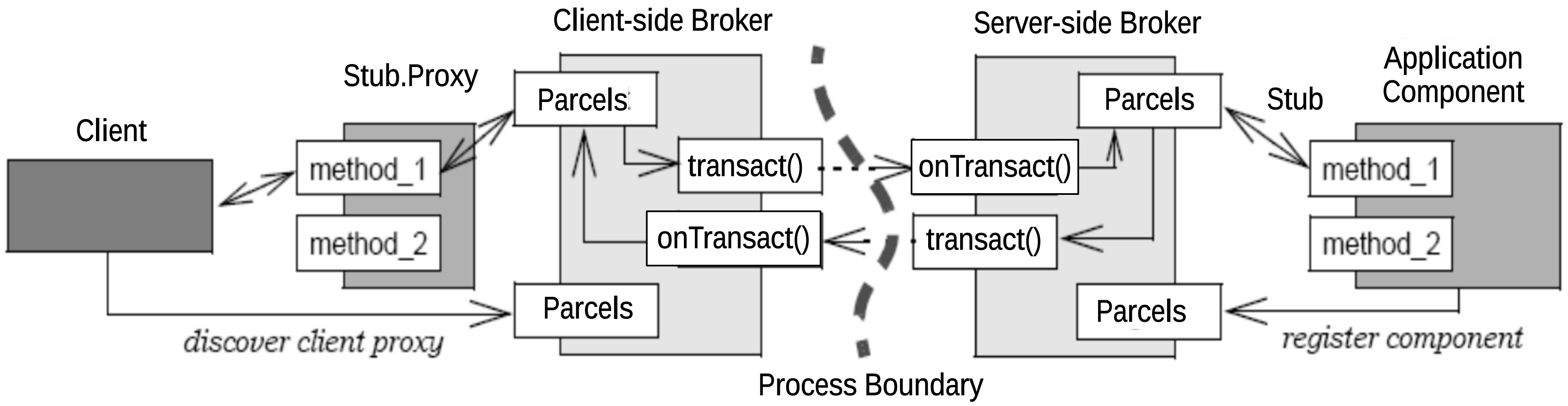
- The NetworkSettings Activity uses the *Activator* pattern to launch the NetworkQueryService to assist in querying the network for service availability



packages/apps/Phone/src/com/android/phone/NetworkSetting.java has source code

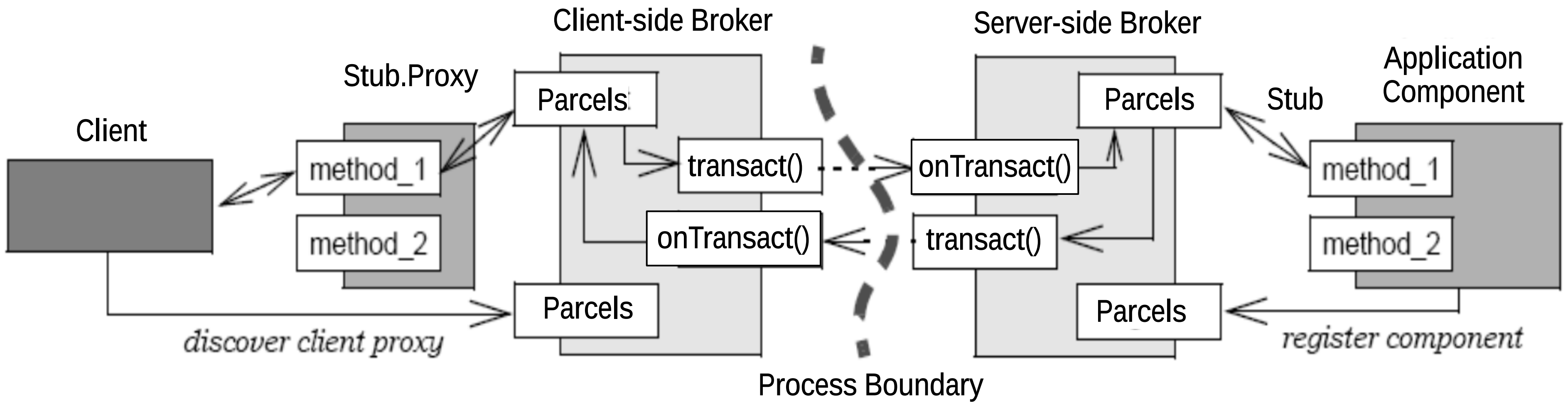
Summary

- Android Bound Services uses *Broker* to invoke methods across processes

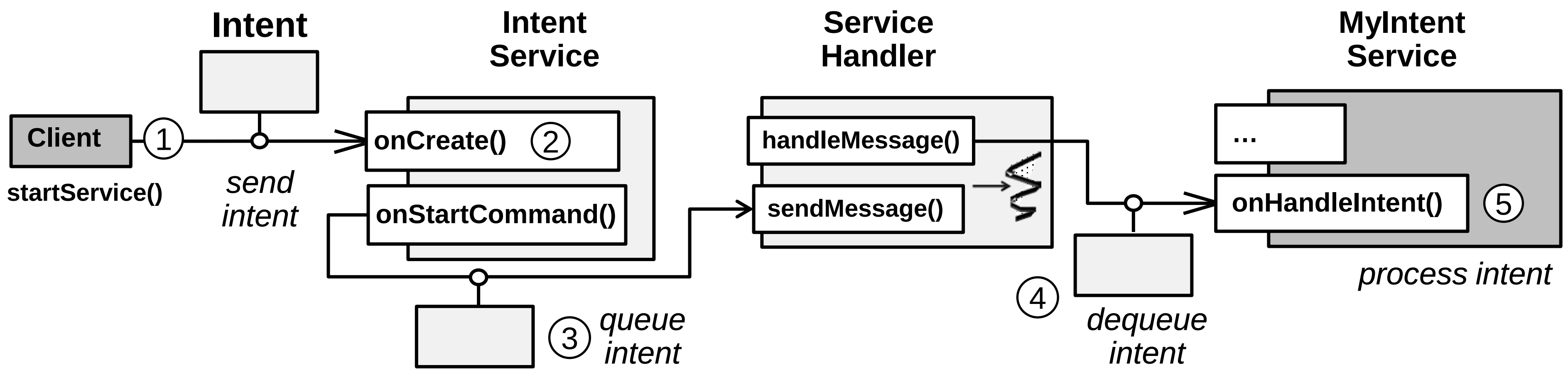


Summary

- Android Bound Services uses *Broker* to invoke methods across processes



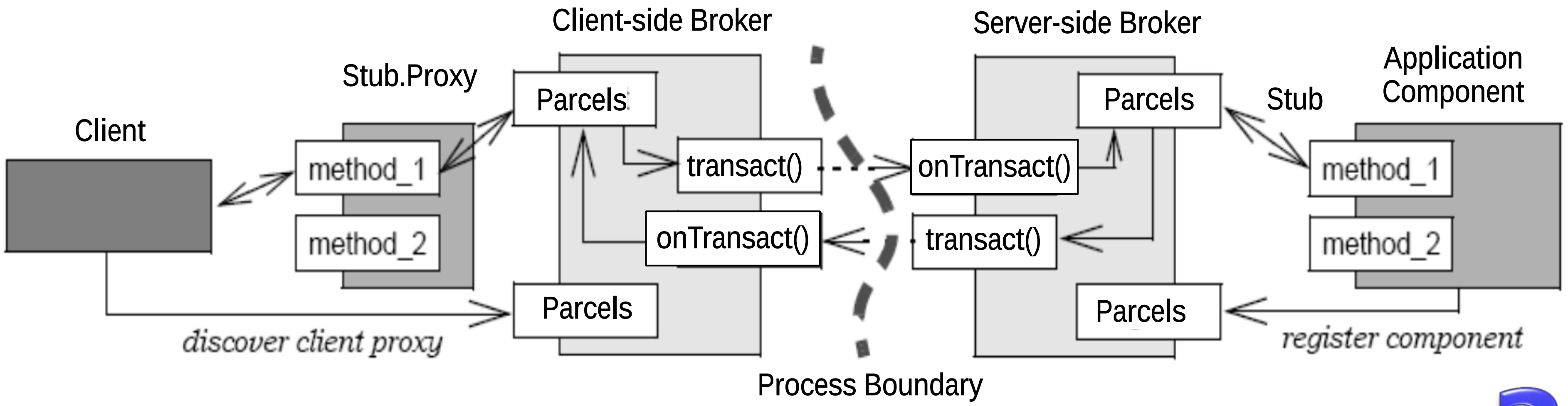
- Android Started Services use *Command Processor* to pass messages



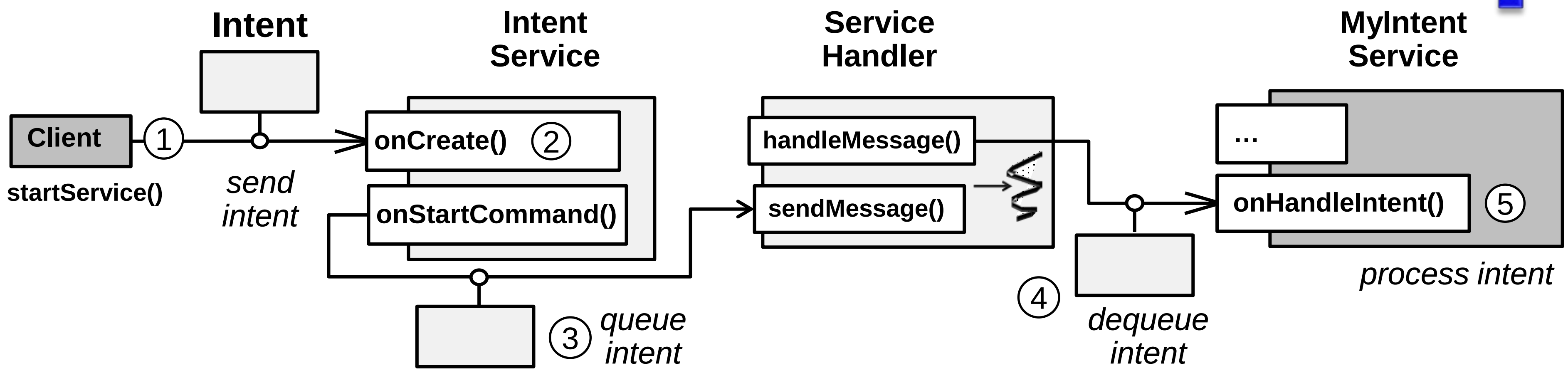
Command Processor & *Broker* are "pattern complements"

Summary

- Android Bound Services uses *Broker* to invoke methods across processes



- Android Started Services use *Command Processor* to pass messages



- Software architects must understand the trade-offs between these patterns