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

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

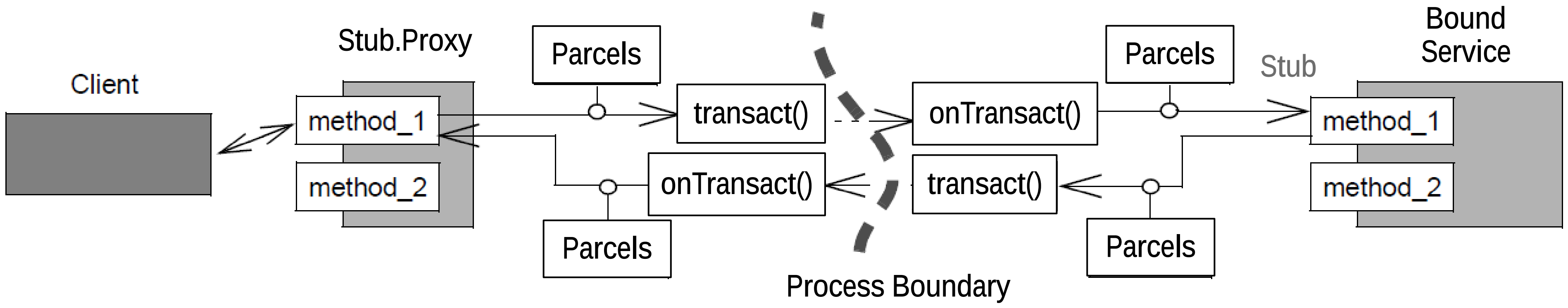
Professor of Computer Science

Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Module

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

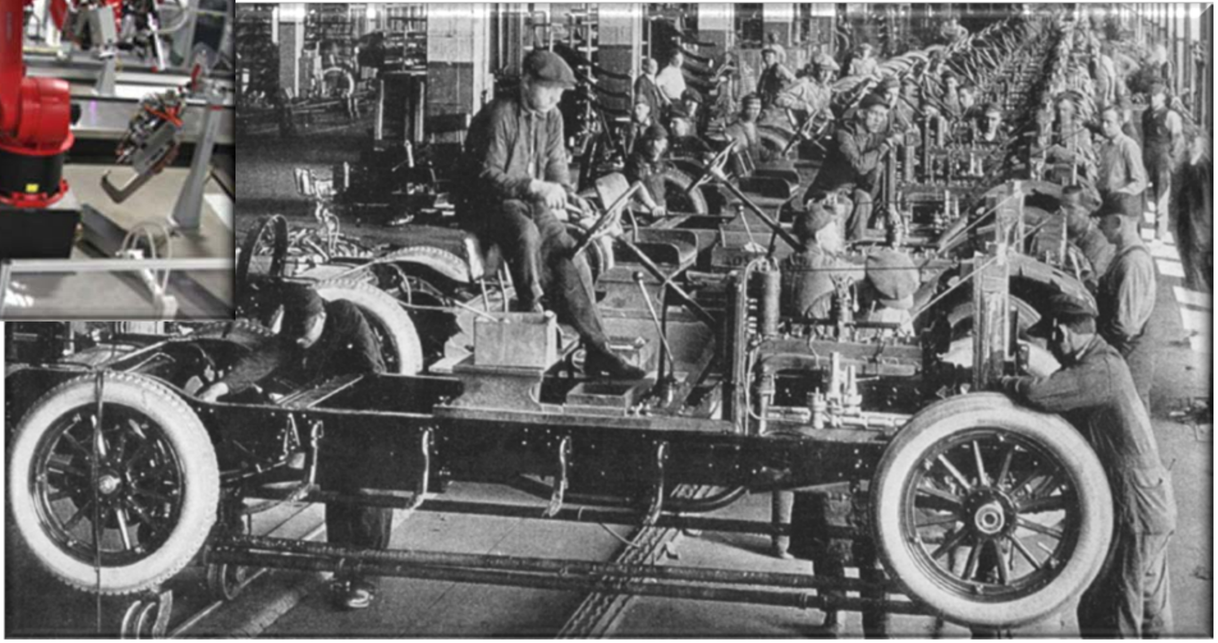


See en.wikipedia.org/wiki/Proxy_pattern for more on *Proxy* pattern

Proxy GoF Object Structural

Implementation

- Auto-generated vs. hand-crafted



Proxy — GoF Object Structural

Implementation

- Auto-generated vs. hand-crafted
- A proxy can cache stable info about the subject to postpone accessing it remotely

Proxy GoF Object Structural

Implementation

- Auto-generated vs. hand-crafted
- A proxy can cache stable info about the subject to postpone accessing it remotely
- Overloading operator–> in C++

```
template <class TYPE> class ACE_TSS {
  TYPE *operator->() const {
    TYPE *tss_data = 0;
    if (!once_) {
      ACE_Guard<ACE_Thread_Mutex>
        g (keylock_);
      if (!once_) {
        ACE_OS::thr_keycreate
          (&key_, &cleanup_hook);
        once_ = true;
      }
    }
    ACE_OS::thr_getspecific
      (key_, (void **) &tss_data);
    if (tss_data == 0) {
      tss_data = new TYPE;
      ACE_OS::thr_setspecific
        (key_, (void *) tss_data);
    }
    return tss_data;
  }
```

www.dre.vanderbilt.edu/~schmidt/PDF/TSS-pattern.pdf

Proxy GoF Object Structural

Applying the Proxy Pattern in Android

```
public interface IDownload extends android.os.Iinterface {
  public static abstract class Stub extends android.os.Binder
                                   implements IDownload {
```

Local-side IPC implementation class

```
    public Stub() {
      this.attachInterface(this, DESCRIPTOR);
    }

    ...
```

Construct the stub & attach it to the interface

Proxy GoF Object Structural

Applying the Proxy Pattern in Android

```
public interface IDownload extends android.os.Iinterface {
  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);
    }

    ...
```

Cast an IBinder object into an IDownload interface, generating a proxy if needed

Proxy GoF Object Structural

Applying the Proxy Pattern in Android

```
public interface IDownload extends android.os.Iinterface {
  public static abstract class Stub ... {
    private static class Proxy implements IDownload {
```

Used by a client to call a remote method

```
      private android.os.IBinder mRemote;

      Proxy(android.os.IBinder remote) {
        mRemote = remote;
      }
      ...
```

Cache Binder for subsequent use by Proxy

This code fragment has been simplified a bit to fit onto the slide

Proxy GoF Object Structural

Applying the Proxy Pattern in Android

```
public interface IDownload extends android.os.Iinterface {
  public static abstract class Stub ... {
    private static class Proxy implements IDownload {
      ...
```

Marshal the parameter, transmit to the remote object, & demarshal the result

```
      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;
   ...
```

This code fragment has been simplified a bit to fit onto the slide

Proxy GoF Object Structural

Applying the Proxy Pattern in Android

```
public interface IDownload extends android.os.Iinterface {
  public static abstract class Stub extends android.os.Binder
                                    implements IDownload {
```

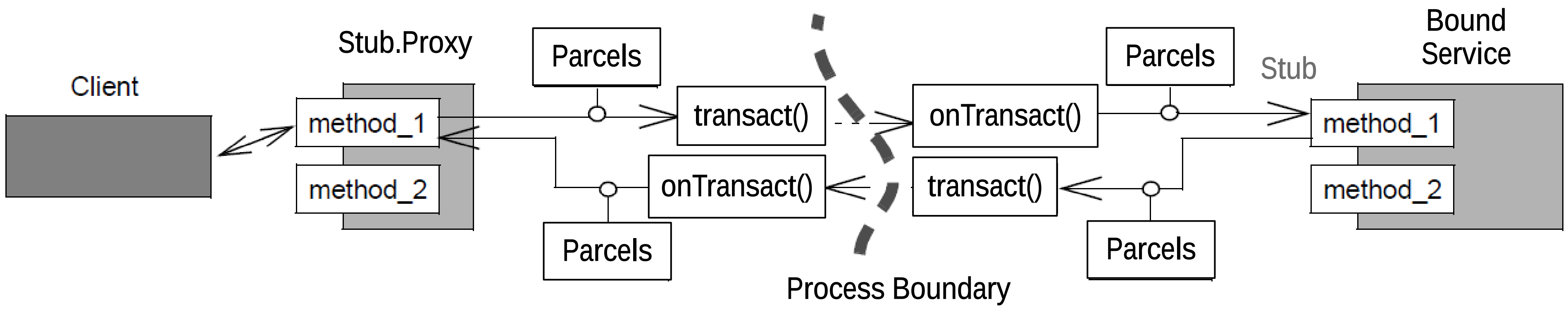
This method is dispatched by Binder RPC to trigger a callback on our downloadImage()

```
    public boolean onTransact(int code, android.os.Parcel data,
                       android.os.Parcel reply, int flags) ... {
      switch (code) {
      case TRANSACTION_downloadImage:
        data.enforceInterface(DESCRIPTOR);
        java.lang.String _arg0 = data.readString();
        java.lang.String _result = this.downloadImage(_arg0);
        reply.writeNoException();
        reply.writeString(_result);
        return true;
        ...
```

Demarshal the parameter, dispatch the upcall, & marshal the result

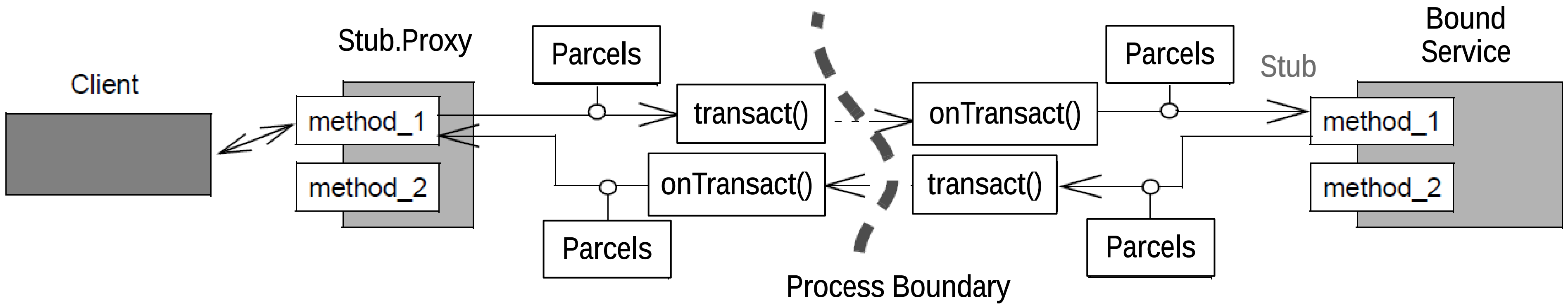
This code fragment has been simplified a bit to fit onto the slide

Summary



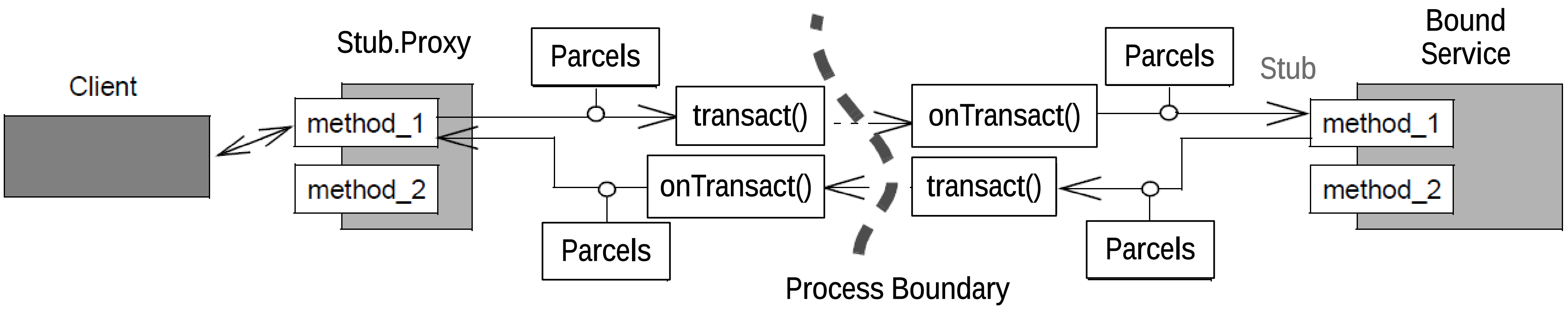
- The Android generated AIDL proxies implement the *Proxy* pattern

Summary



- The Android generated AIDL proxies implement the *Proxy* pattern
- Proxies support a remote method invocation style of IPC
    - As a result, there is no API difference between a call to a local or a remote component, which enhances location-independent communication within an Android App

Summary



- The Android generated AIDL proxies implement the *Proxy* pattern
- Proxies support a remote method invocation style of IPC
- In addition, a proxy can shield its clients from changes in the represented component's 'real' interfaces, which avoids rippling effects in case of component evolution