Working with Services

Santa Clara University

Center for Science, Technology, and Society

Working with Services

- A Service (android.app.Service) can be thought of as a component that has no UI.
- An Android Service can be one of two things, or both:
  - It can be used to perform lengthy operations beyond the scope of a single Activity.
  - It can be the server of a client/server for providing functionality through remote invocation via interprocess communication.
- A Service is often used to control long-running server operations.
- Generally, use a Service when no input is required.

Android Tasks Types

There are Three different types of tasks that you can do in Android:

- Foreground
  - e.g. audio app, doesn't require interaction
- Background
  - next slide
- Bound
  - Client-server interface, e.g., send, receive requests

Background Tasks

- Scheduled jobs: to refresh your app's cached data
- Services: a music app, you would want to define a service to handle the audio playback.
- Alarms: to start your app up at a specific time.
- Broadcasts: register to receive system broadcasts, and perform actions when it receives those broadcasts.

https://developer.android.com/guide/background/

Working with Services (2)

- **A service is NOT a separate process**
- **A Service is NOT a thread.**

- A facility for the application to tell the system *about* something it wants to be doing in the background
  - Context.startService(): ask the system to schedule work for the service, to be run until the service or someone else explicitly stop it.

- A facility for an application to expose some of its functionality to other applications.
  - Context.bindService(): allows a long-standing connection to be made to the service in order to interact with it.

Working with Services (3)

Examples of when to implement a Service:

- A weather, email, or social network app
  - Routinely check for updates on the network
- A game
  - Downloading and processing content for the next level before the user needs it
- A photo or media app
  - To keep data in sync online
  - To package and upload new content in the background
- A video-editing app
  - To offload heavy processing to a queue on a server in order to avoid affecting system performance
- A news application
  - For “preloading” content by downloading news stories in advance, to improve performance and responsiveness

Adding Service to Manifest

You need to add a Service to AndroidManifest.xml file

- Add a <service> element as a child of the <application> element & provide android:name to reference your Service class

- Use android:process = ":myProcess" to run the service in its own process

```
<manifest…>
 …
 <application…>
   <service android:name=".MyService"/>
    …
 </application>
</manifest>
```

Adding Service to Manifest

SMS Service

```
<service
  android:name=".transaction.SmsReceiverService"
  android:exported="true"/>
```

Music Service

```
<service
  android:name="com.android.music.MediaPlaybackService"
  android:exported="true"/>
```

Service Syntax

```
<service
        android:description="string resource"
        android:directBootAware=["true"|"false"]
        android:enabled=["true"|"false"]
        android:exported=["true"|"false"]
        android:icon="drawable resource"
        android:isolatedProcess=["true"|"false"]
        android:label="string resource"
        android:name="string"
        android:permission="string"
        android:process="string">
    . . .
</service>
```

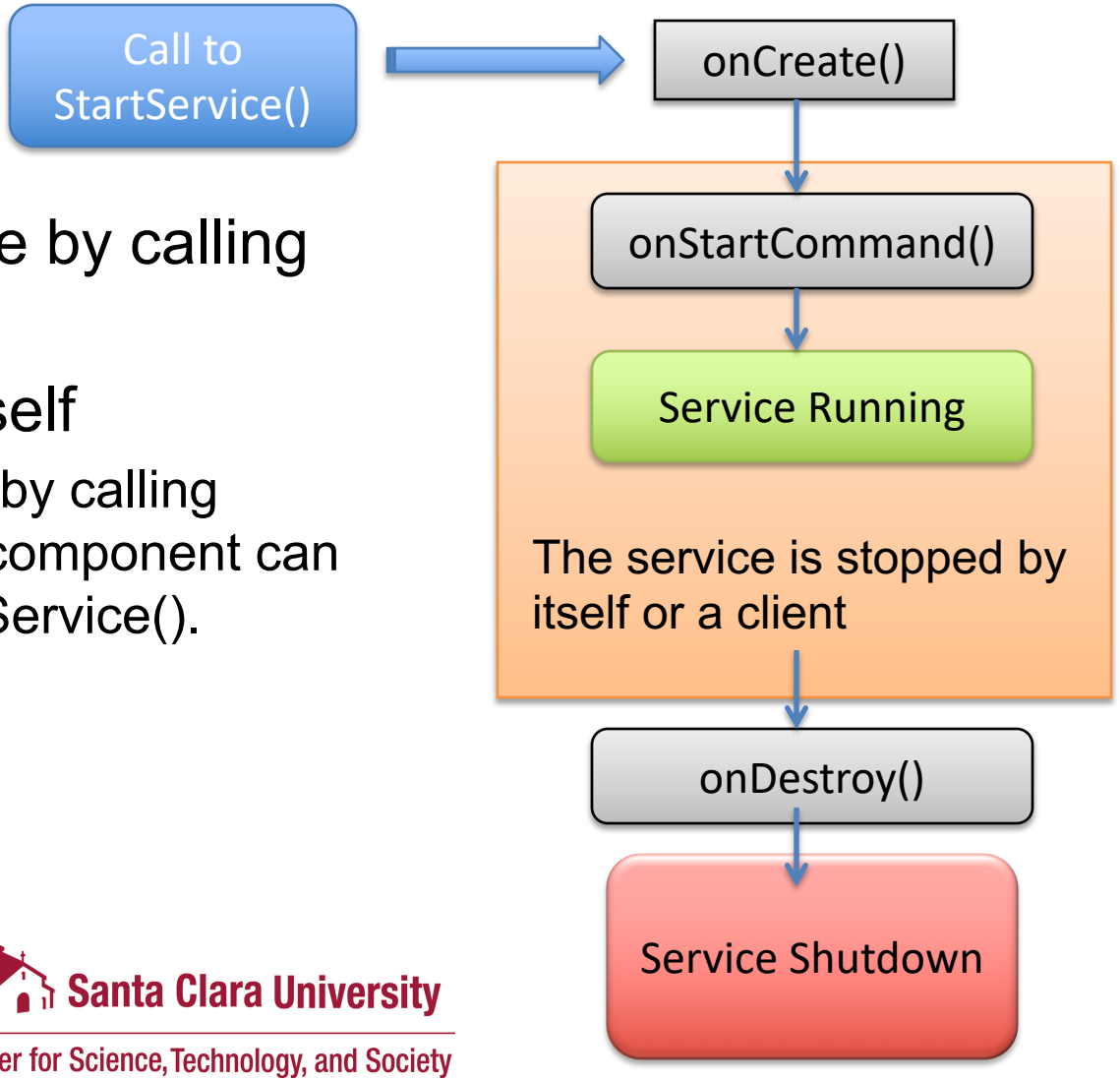
Service Types

There are three types of Services:

1. Started Service: Often perform a single operation and might not return a result to caller directly.
   - Example: SMS Service, Alert Service
2. Scheduled Service
3. Bound Service: Provide a client/server interface that allows for a conversation with the Service
   - Example: MediaPlaybackService, Email exchange service

Started Service

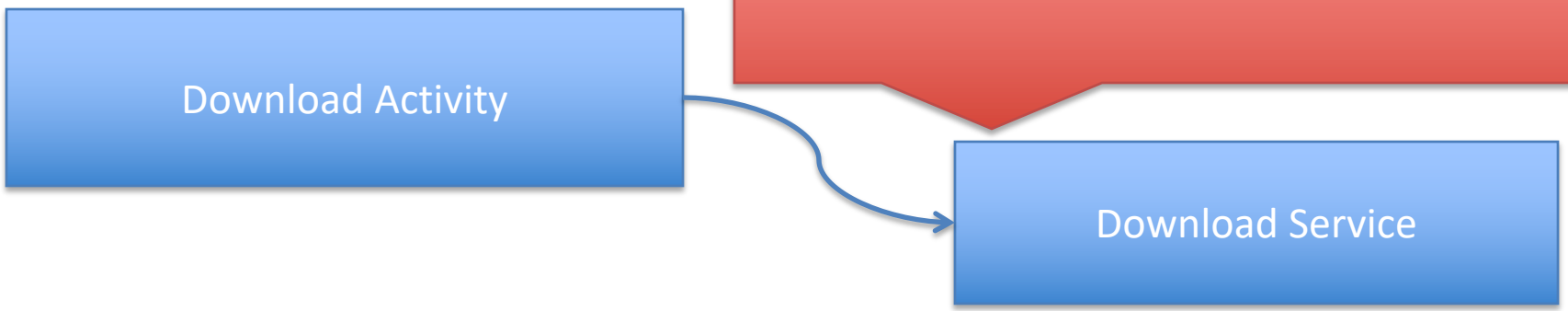
- We can start a service by calling startService().
- A Service can stop itself
  - when the job is done, by calling stopSelf() or another component can stop it by calling stopService().



Started Service(2)

```
Intent intent = new Intent(this, DownloadService.class);
intent.putExtra("URL", imageUrl);
startService(intent);
```

```
public class DownloadService extends Service{
    public int onCreate(…) {}
    public int onStartCommand(..) {}
    public void onDestroy() {}
}
```



Programming Started Service

import android.app.service

```
public class DownloadService extends Service {
    public int onCreate() {..}
    public int onStartCommand(Intent intent, int flags, int startId) {
      return super.onStartCommand(intent, flags, startId);
    }
    public void onDestroy() {
      super.onDestroy();
    }
    public Ibinder onBind(Intent intent) {
        return null;
    }
}
```

Programming Started Service (2)

- A Service is activated via Context.startService()
- startService() does not block
- Service doesn't return results to callers, but do return value to Android system via onStartCommand()

```
public int onStartCommand(Intent intent, int flags, int startId) {
    Log.i("LocalService", "Received start id " + startId + ": " + intent);
    return START_NOT_STICKY;
}
```

When to Implement a Service?

- As a good rule of thumb, if the task
  - . . . requires the use of a worker thread
  - . . . might affect application responsiveness and performance
  - . . . and is not time sensitive to the application

  Consider implementing a service to handle the task outside the main application and any individual `Activity` lifecycles.

IntentService

- IntentService is a subclass for Services that handle asynchronous requests (expressed as Intents) on demand.
  - Clients send requests through startService(Intent) calls;
  - the service is started as needed, handles each Intent in turn using a worker thread, and stops itself when it runs out of work.
- This "work queue processor" pattern is commonly used to offload tasks from an application's main thread.
- All requests are handled on a single worker thread
  - they may take as long as necessary (and will not block the application's main loop), but only one request will be processed at a time.

Service vs. IntentService

- Service runs actually in the same thread of your app; when you extends Service, you must manually spawn new threads to run CPU blocking operations.
- IntentService
  - Is a subclass of Service which spawns a thread to do background work. Handles asynchronous requests.
  - Creates a worker queue that passes one Intent at a time
  - Automatically stops itself

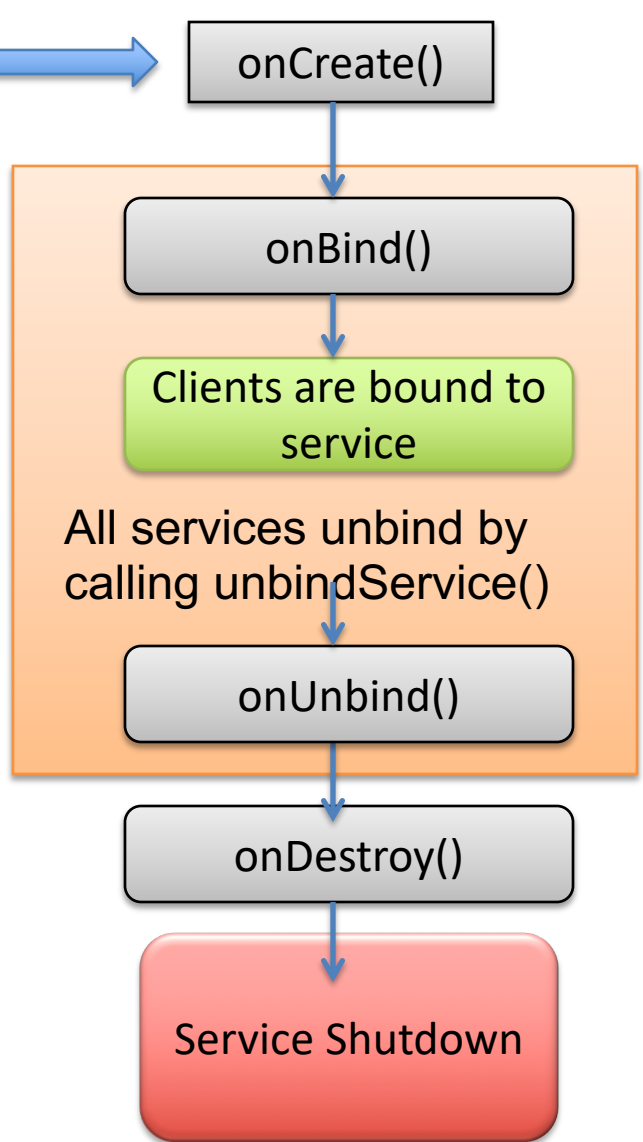
Bound Service

Santa Clara University

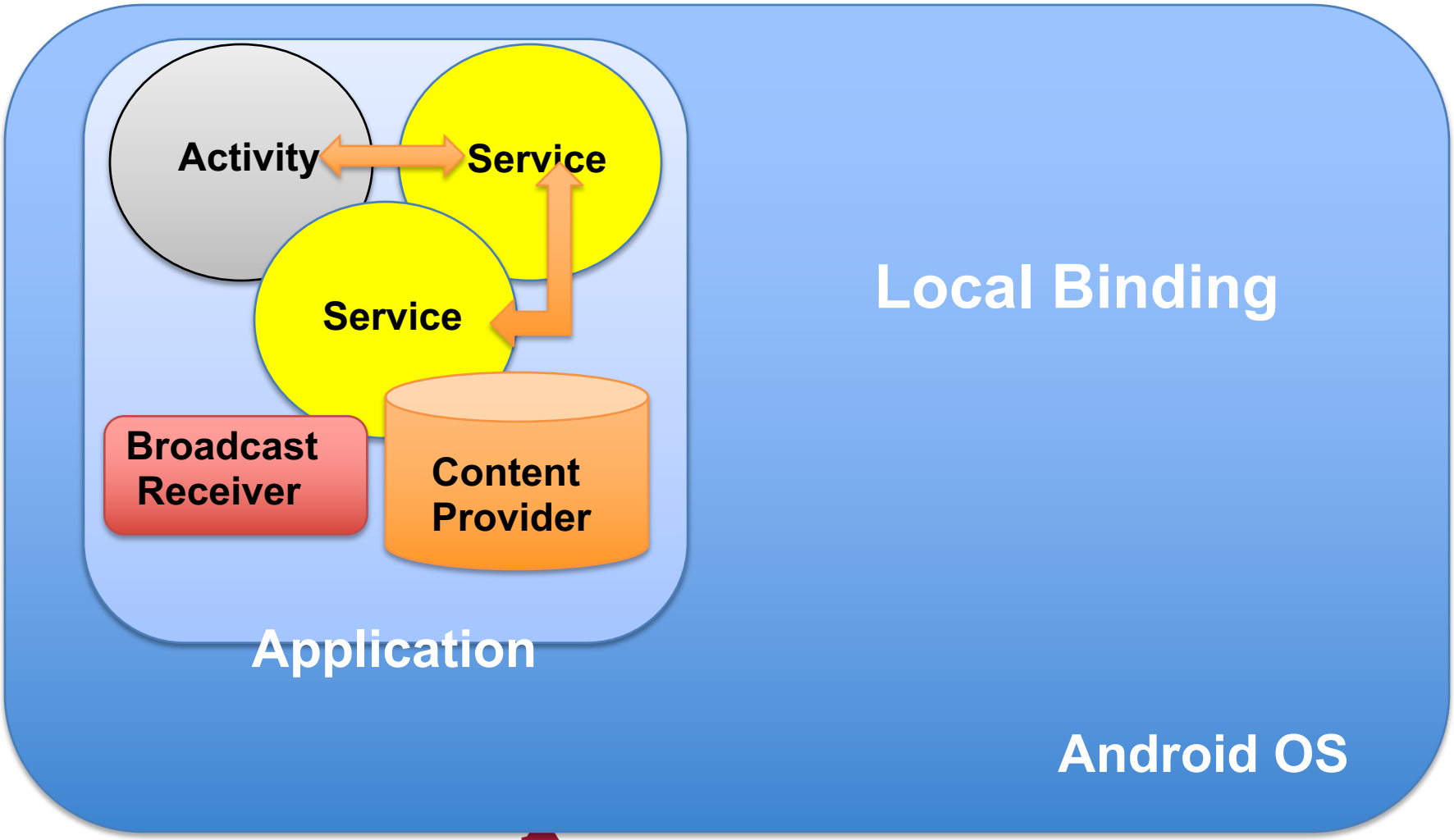
Center for Science, Technology, and Society

Bound Service Behavior

- A bound service allows app component to bind to it by calling bindService() to create a long-standing connection
- A bound service offers a client-server interface that allows components to interact with the service. Send requests, get results via IPC
- **A bound service runs only as long as another application component is bound to it**



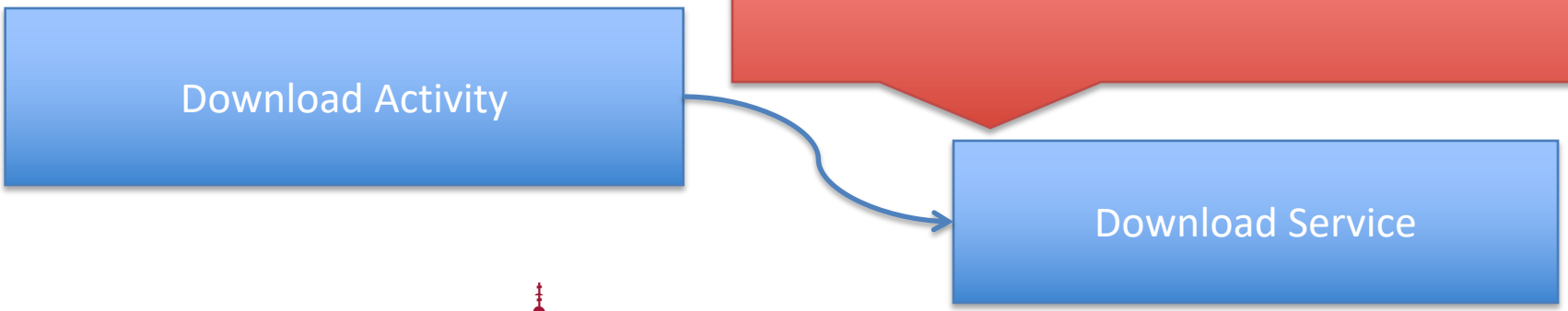
Bound Service



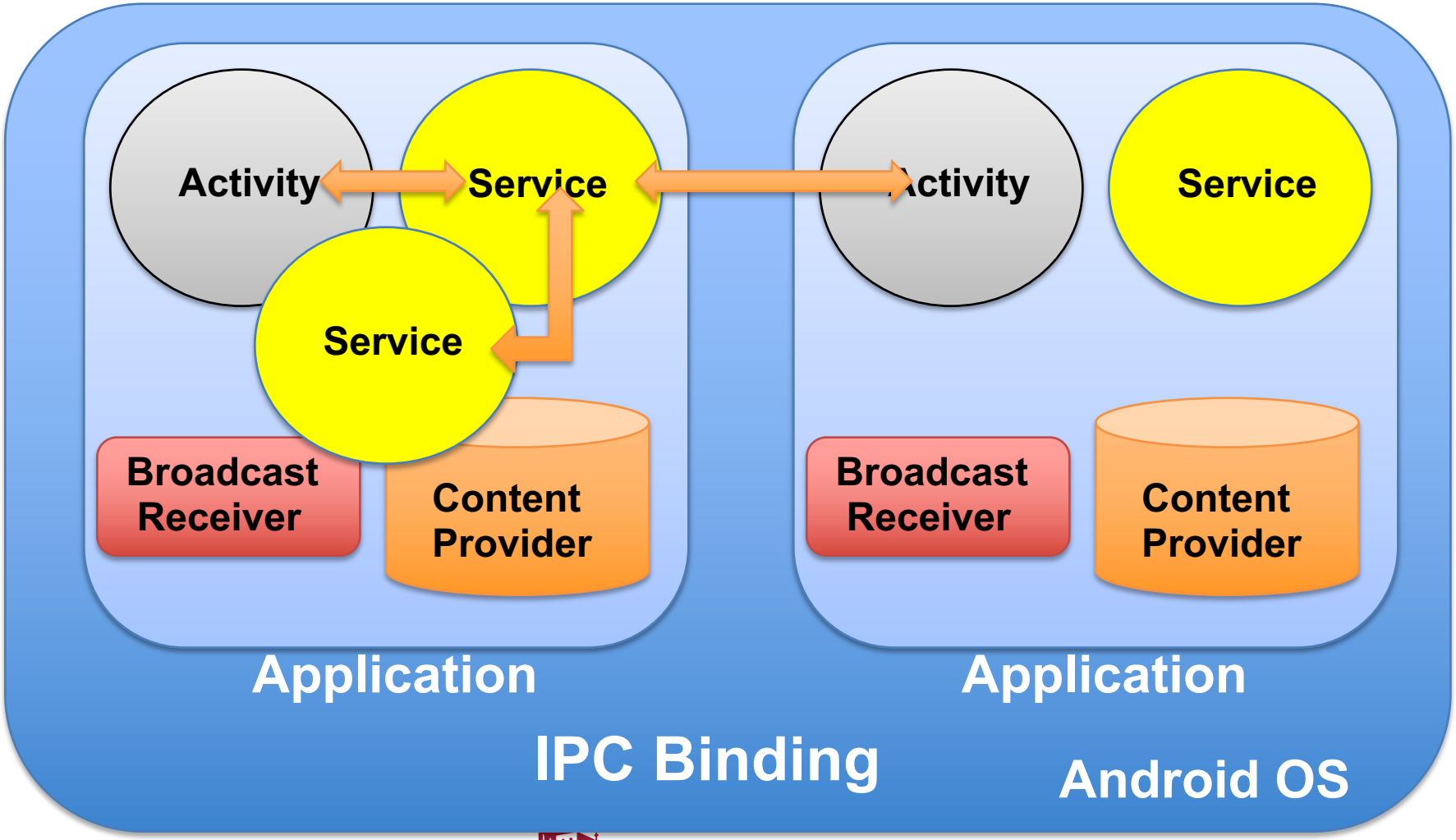
Bound Service(cont'd)

```
Intent intent = new Intent(IDownloadSync.class.getName());
bindService(intent, this.syncConnection, Context.BIND_AUTO_CREATE);
```

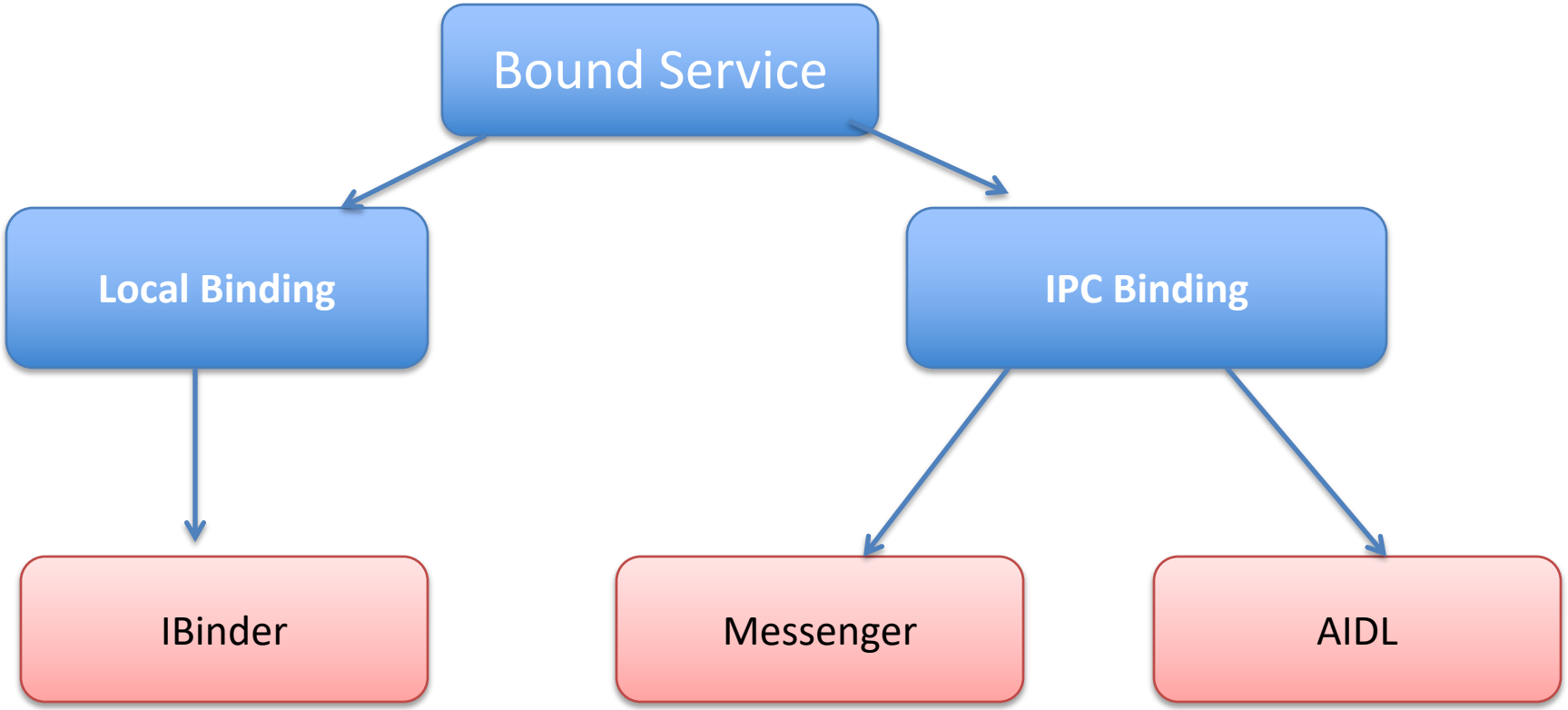
```
public class DownloadService extends Service{
    public int onCreate(…) {}
    public Ibinder onBind(Intent intent) {}
    public void onDestroy() {}
}
```



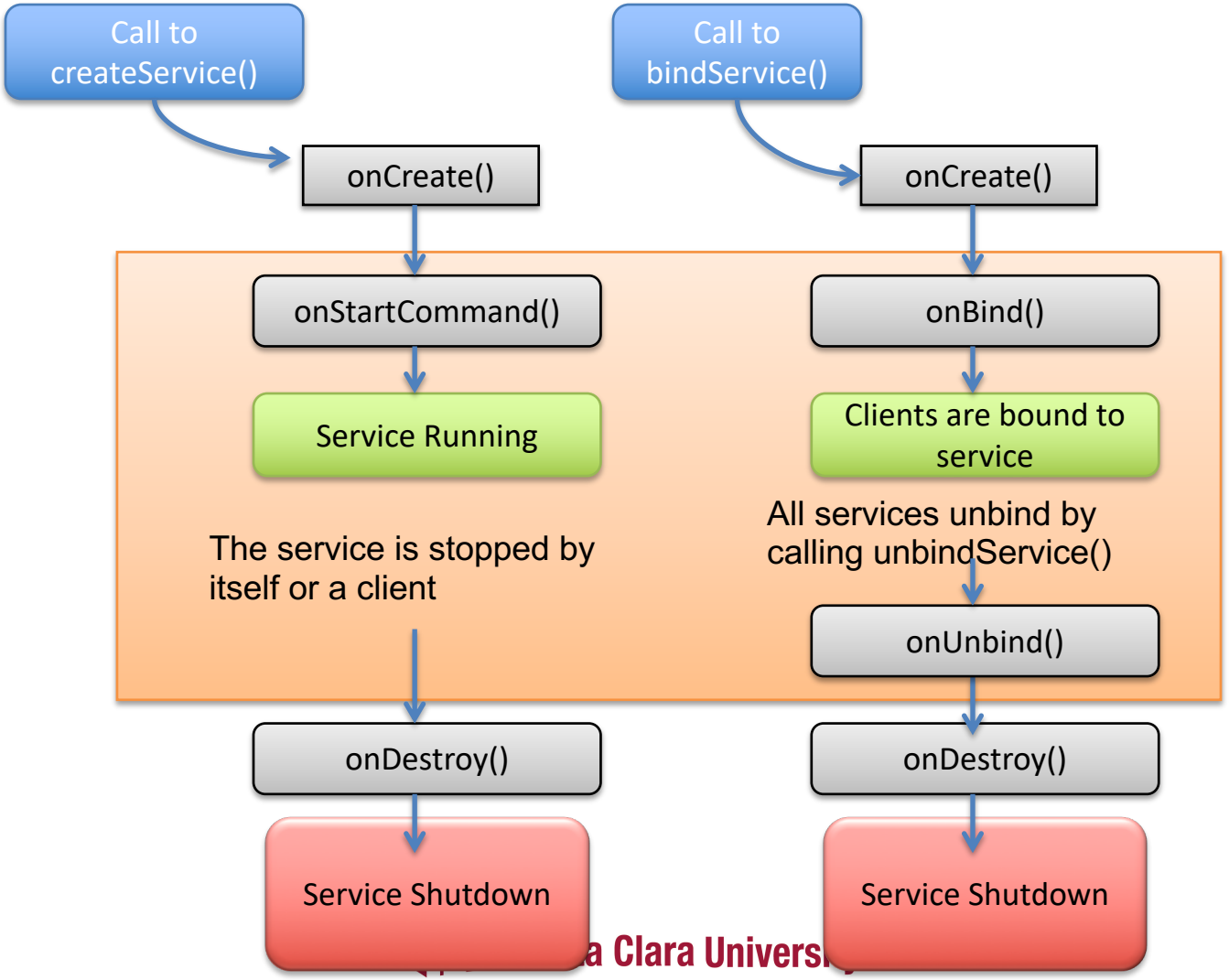
Bound Service Behavior



Bound Service Behavior (2)



Summary


Call to createService()
Call to bindService()
onCreate()
onCreate()
onStartCommand()
onBind()
Service Running
Clients are bound to service
The service is stopped by itself or a client
All services unbind by calling unbindService()
onUnbind()
onDestroy()
onDestroy()
Service Shutdown
Service Shutdown

Example

- There are many examples in the Internet
- Check
  - https://www.techotopia.com/index.php/Android_Local_Bound_Services_%E2%80%93_A_Worked_Example

System Service

- System Services are services provided by the system, not necessarily tied to user Activity at all
- "Manager" – type classes that you can get by calling getSystemService() on *Context , e.g, Activity, Service*
- Examples:
  - AlarmManager, AudioManager, DownloadManager, NotificationManager; LocationManager, SensorManager; WiFiManager.

Service Example 1

- Our service, when called, creates a thread
- The thread is waiting for a message from the service module
- Based on the message it can do something
  - Sleep
  - Stop the service

Service 1

- onCreate
  - Create the thread
  - Create service handler which manages the message queue for the thread
- onStart
  - Send a message to the thread queue to process
  - Toast.makeText
- onDestroy
  - Toast.makeText

Thread

- Start a thread, since the service is running in the application main process

```
HandlerThread thread = new
HandlerThread("ServiceStartArguments",
        Process.THREAD_PRIORITY_BACKGROUND);
thread.start();
```

- ***THREAD_PRIORITY_BACKGROUND*** is priority 10 which is lower than normal priority: won't impact UI

Message Handler

- Handler which needs a looper
- thread.getLooper which gives you the looper for that particular thread

```
private final class ServiceHandler extends
Handler {
    public ServiceHandler(Looper looper) {
        super(looper);
    }
```

Activity work

- Create the intent for the service
  - onResume

```
Intent intent = new Intent(this, HelloService.class);
startService(intent);
registerReceiver(receiver,new
IntentFilter(DownloadService.NOTIFICATION));
```

Service 2: Intent Service

- Handles asynch requests on demand
- Clients send requests through Context.startService(Intent) calls
- The service is started as needed, handles each Intent in turn using a worker thread, and stops itself when it runs out of work (work queue process pattern)
- When a button is clicked
  - download the content of a website.
  - Store it locally
  - Report back the result success/failure to activity

Download Service

- Extend IntentService
- Implement onHandleIntent
- Send/publish results back to activity
- Activity:
  - Create Intent
  - Send the URL
  - Create a receiver to get the result back