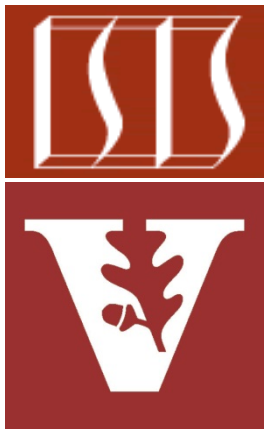
Android Services & Local IPC: Service and Activity Communication via Android Messengers

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



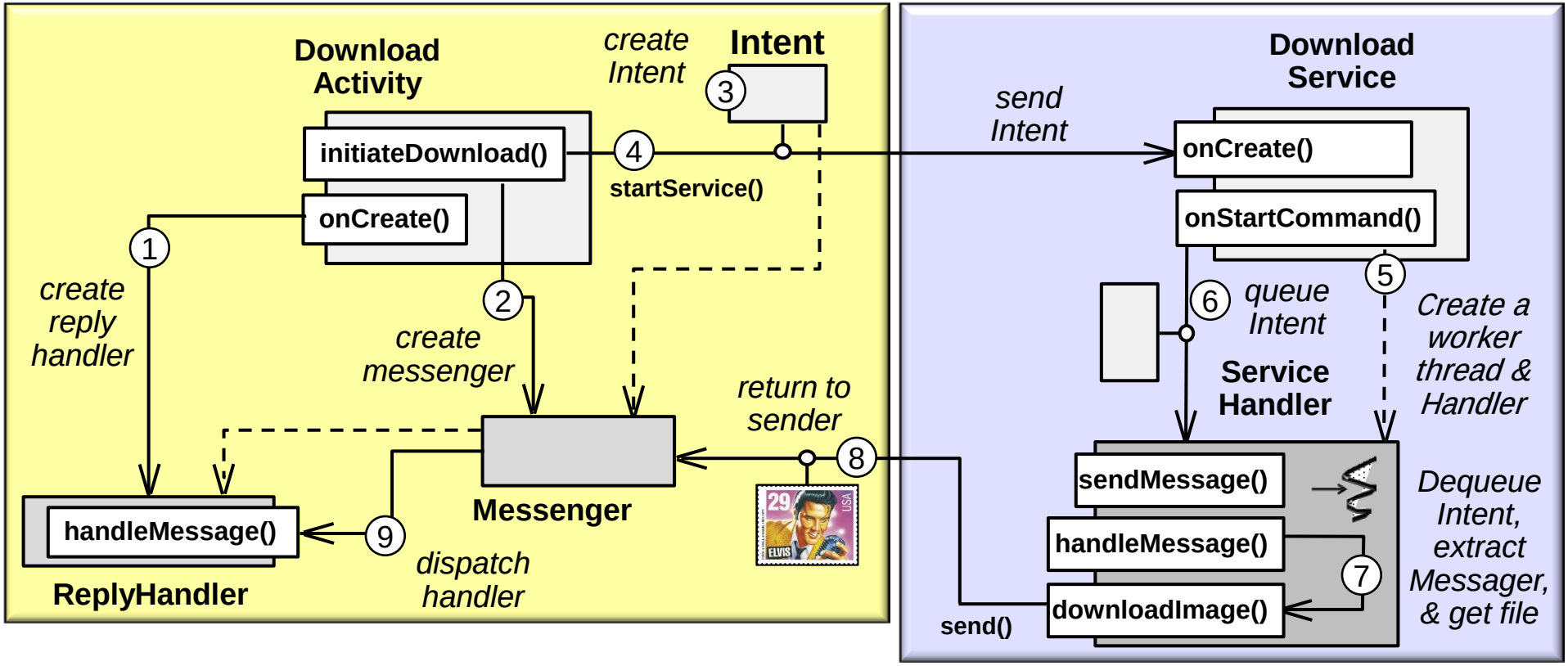
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

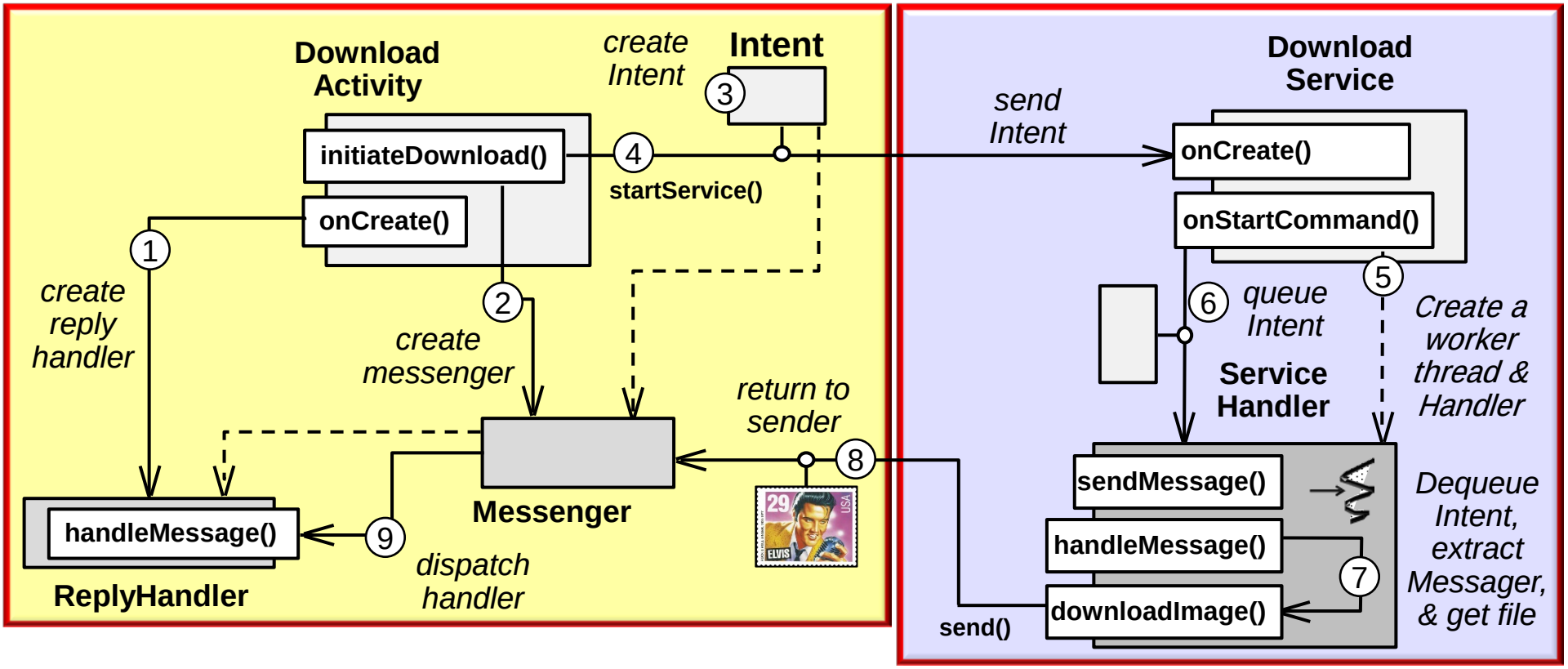
Learning Objectives in this Part of the Module

- Understand how the Android Messenger generalizes the HaMeR concurrency framework to enable Message-based communication with Handlers across components & processes



Learning Objectives in this Part of the Module

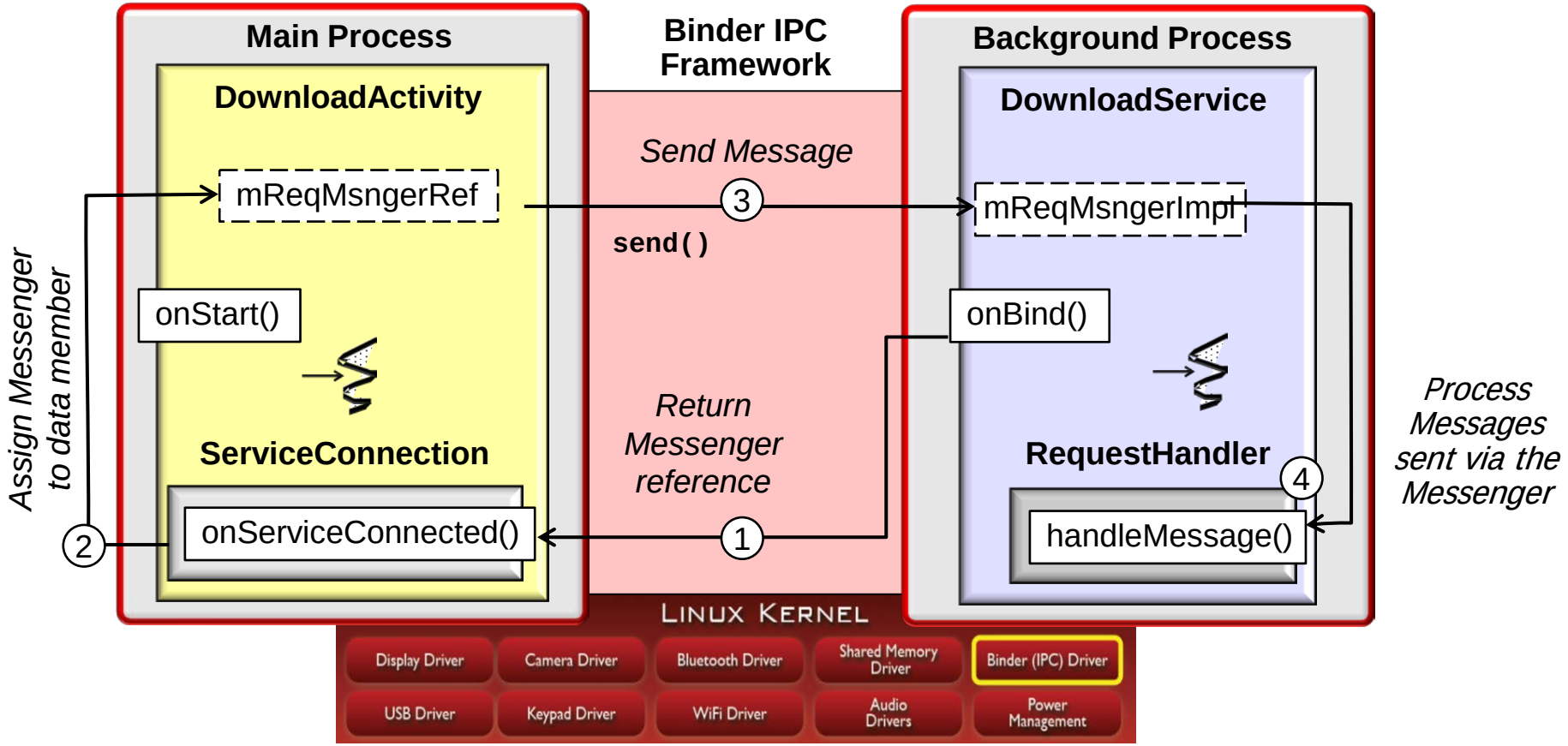
- Understand how the Android Messenger generalizes the HaMeR concurrency framework to enable Message-based communication with Handlers across components & processes



A Started Service can use a Messenger to send a Message to an Activity

Learning Objectives in this Part of the Module

- Understand how the Android Messenger generalizes the HaMeR concurrency framework to enable Message-based communication with Handlers across components & processes



A Bound Service can also use Messengers to exchange Messages with an Activity

Overview of Messengers

Overview of Messengers

- Handlers can be used to send & process Messages in a single component

Handler

extends Object

java.lang.Object
↳ android.os.Handler

▸ Known Direct Subclasses
AsyncQueryHandler, AsyncQueryHandler.WorkerHandler, HttpAuthHandler, SslErrorHandler

Class Overview

A Handler allows you to send and process `Message` and Runnable objects associated with a thread's `MessageQueue`. Each Handler instance is associated with a single thread and that thread's message queue. When you create a new Handler, it is bound to the thread / message queue of the thread that is creating it -- from that point on, it will deliver messages and runnables to that message queue and execute them as they come out of the message queue.

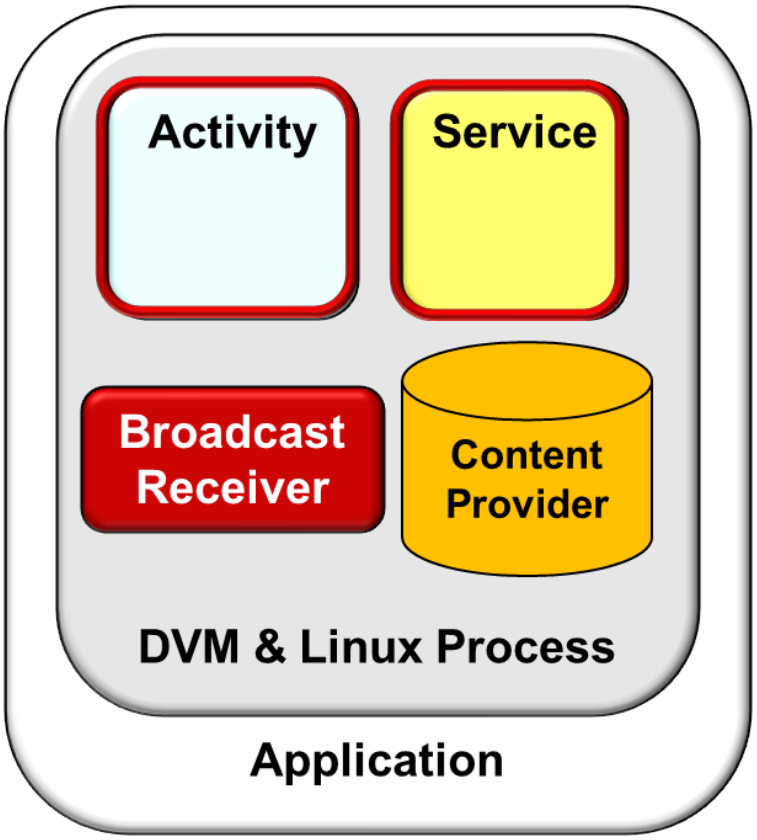
There are two main uses for a Handler: (1) to schedule messages and runnables to be executed as some point in the future; and (2) to enqueue an action to be performed on a different thread than your own.

Scheduling messages is accomplished with the `post(Runnable)`, `postAtTime(Runnable, long)`, `postDelayed(Runnable, long)`, `sendEmptyMessage(int)`, `sendMessage(Message)`, `sendMessageAtTime(Message, long)`, and `sendMessageDelayed(Message, long)` methods. The *post* versions allow you to enqueue Runnable objects to be called by the message queue when they are received; the *sendMessage* versions allow you to enqueue a `Message` object containing a bundle of data that will be processed by the Handler's `handleMessage(Message)` method (requiring that you implement a subclass of Handler).

See developer.android.com/reference/android/os/Handler.html

Overview of Messengers

- Handlers can be used to send & process Messages in a single component



Handler

extends Object

java.lang.Object
↳ android.os.Handler

▸ Known Direct Subclasses
AsyncQueryHandler, AsyncQueryHandler.WorkerHandler, HttpAuthHandler, SslErrorHandler

Class Overview

A Handler allows you to send and process `Message` and Runnable objects associated with a thread's `MessageQueue`. Each Handler instance is associated with a single thread and that thread's message queue. When you create a new Handler, it is bound to the thread / message queue of the thread that is creating it -- from that point on, it will deliver messages and runnables to that message queue and execute them as they come out of the message queue.

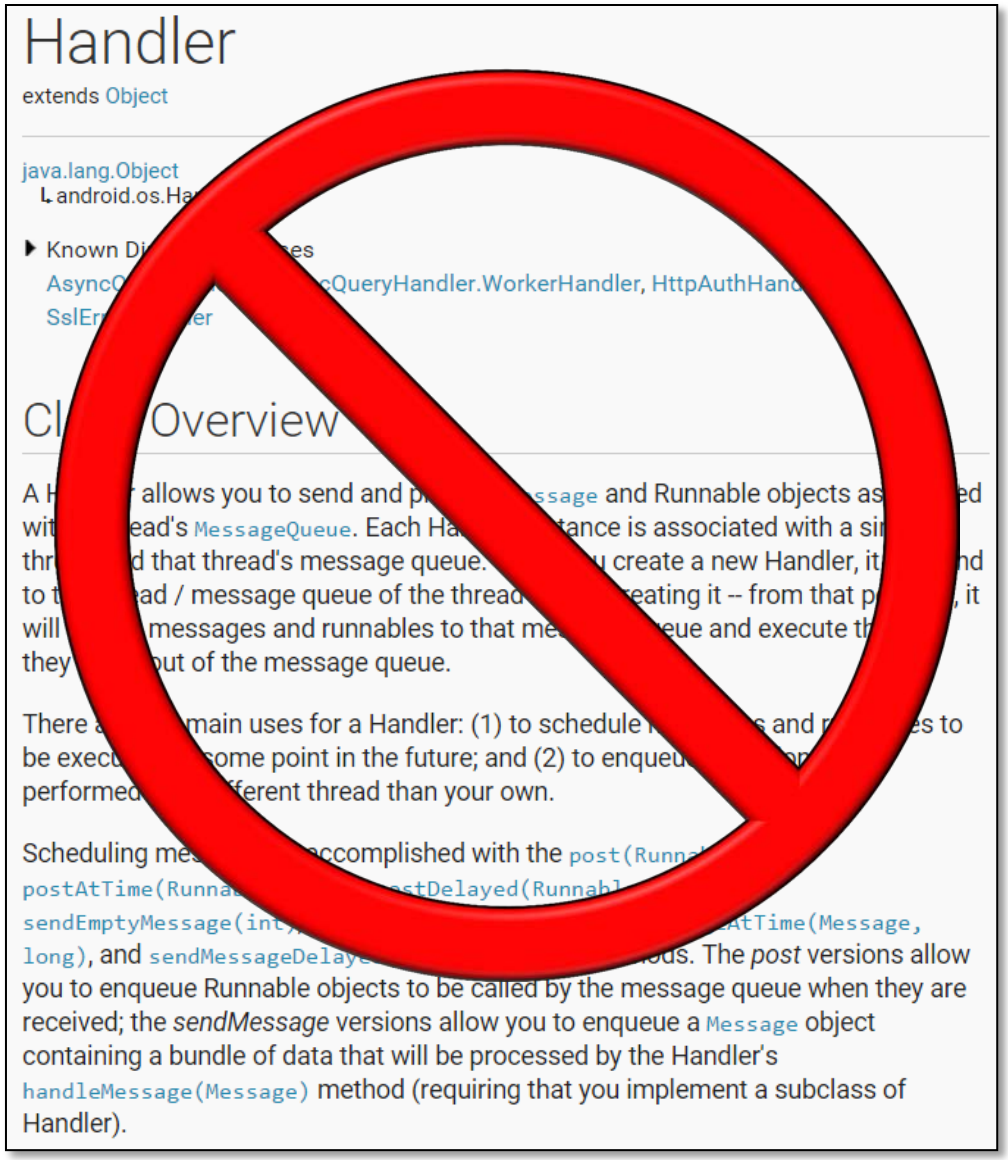
There are two main uses for a Handler: (1) to schedule messages and runnables to be executed as some point in the future; and (2) to enqueue an action to be performed on a different thread than your own.

Scheduling messages is accomplished with the `post(Runnable)`, `postAtTime(Runnable, long)`, `postDelayed(Runnable, long)`, `sendEmptyMessage(int)`, `sendMessage(Message)`, `sendMessageAtTime(Message, long)`, and `sendMessageDelayed(Message, long)` methods. The *post* versions allow you to enqueue Runnable objects to be called by the message queue when they are received; the *sendMessage* versions allow you to enqueue a `Message` object containing a bundle of data that will be processed by the Handler's `handleMessage(Message)` method (requiring that you implement a subclass of Handler).

See earlier parts on the Android HaMeR framework

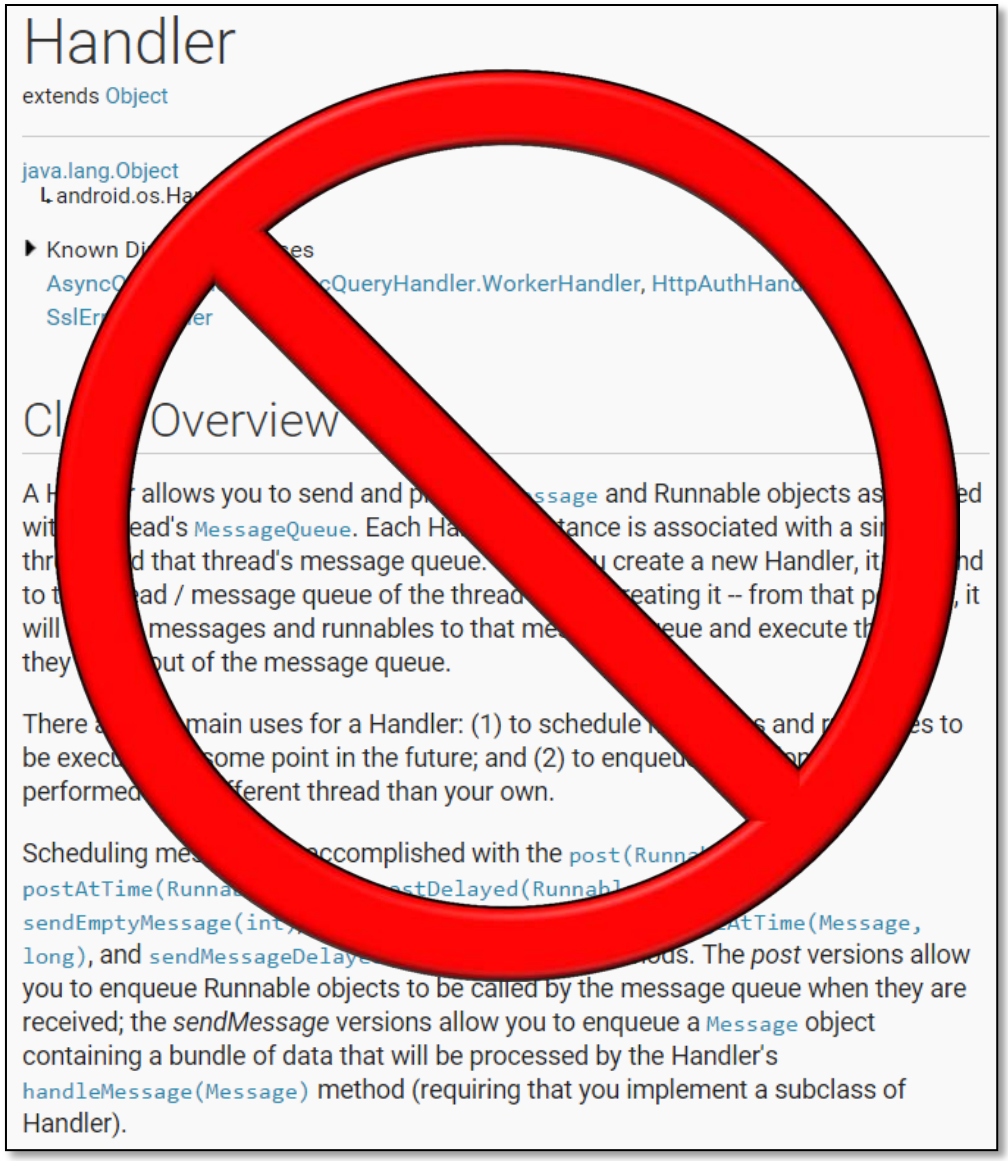
Overview of Messengers

- Handlers can be used to send & process Messages in a single component
  - However, Handler's can't used for Message-based IPC *between* components



Overview of Messengers

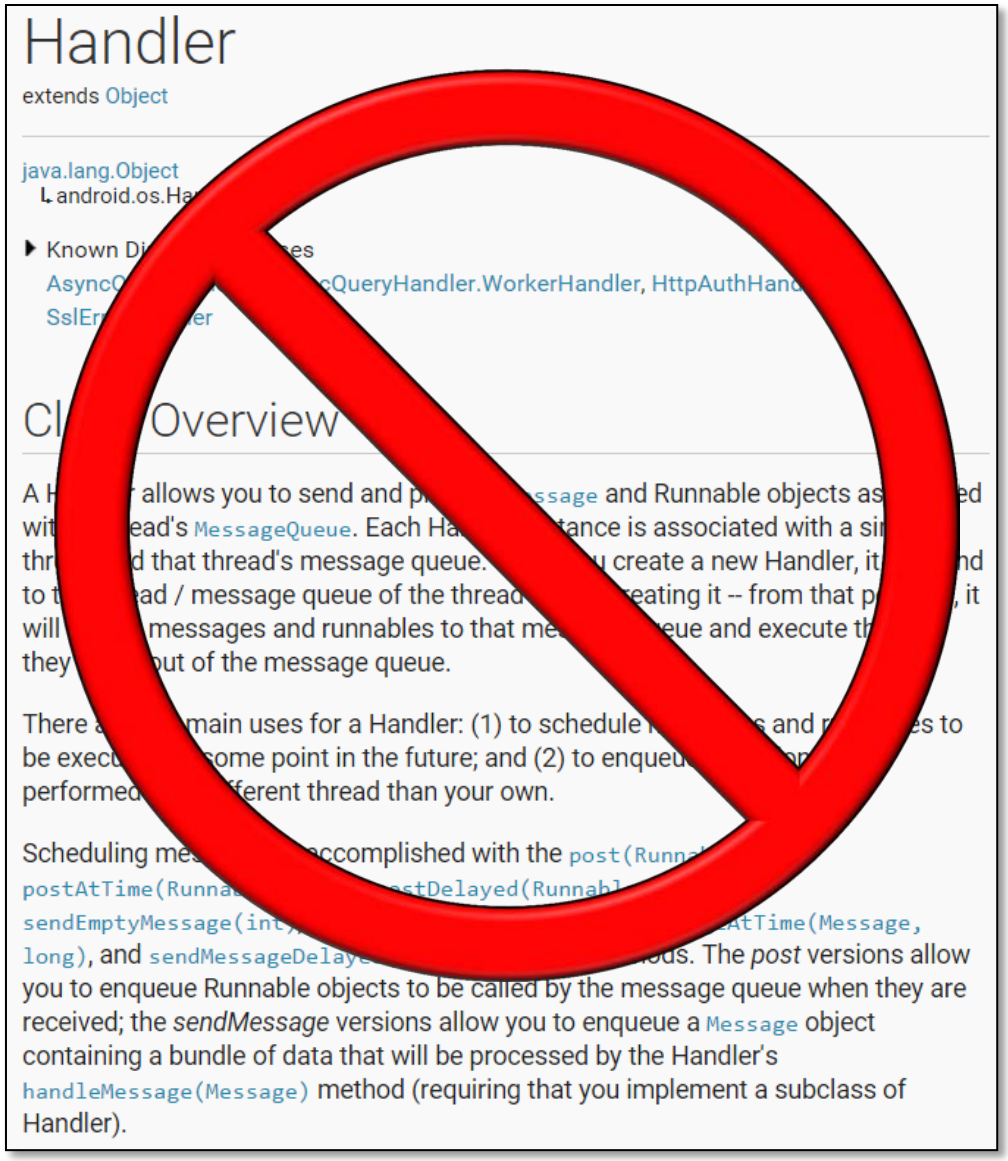
- Handlers can be used to send & process Messages in a single component
  - However, Handler's can't used for Message-based IPC *between* components
  - Handlers don't implement the Parcelable interface



See developer.android.com/reference/android/os/Parcelable.html

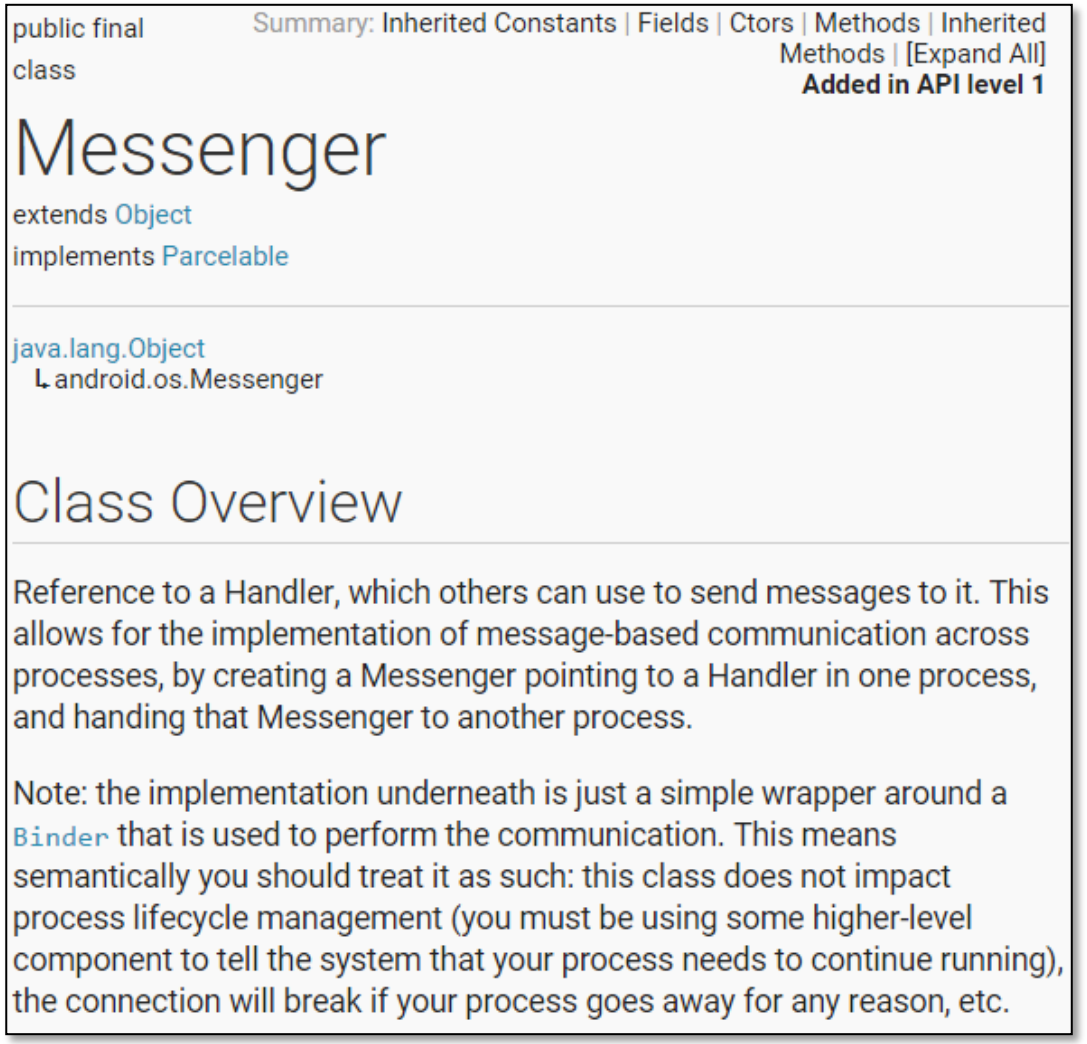
Overview of Messengers

- Handlers can be used to send & process Messages in a single component
  - However, Handler's can't used for Message-based IPC *between* components
  - Handlers don't implement the Parcelable interface
    - They can't be passed as data in a Message or as an "extra" in an Intent



Overview of Messengers

- Handlers can be used to send & process Messages in a single component
- A Messenger encapsulates access to a Handler in one component



public final class

Summary: Inherited Constants | Fields | Ctors | Methods | Inherited Methods | [Expand All]

Added in API level 1

Messenger

extends Object

implements Parcelable

java.lang.Object
↳ android.os.Messenger

Class Overview

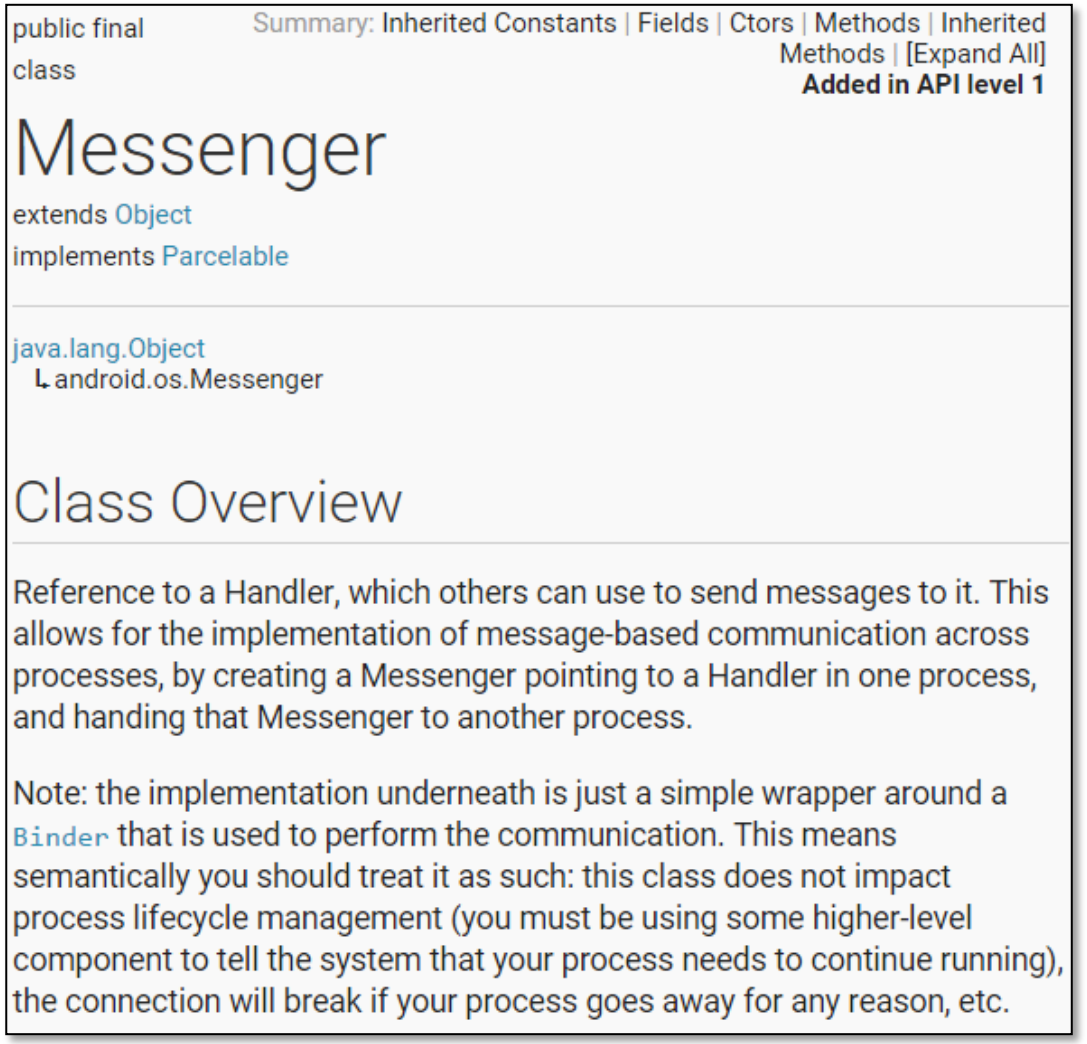
Reference to a Handler, which others can use to send messages to it. This allows for the implementation of message-based communication across processes, by creating a Messenger pointing to a Handler in one process, and handing that Messenger to another process.

Note: the implementation underneath is just a simple wrapper around a `Binder` that is used to perform the communication. This means semantically you should treat it as such: this class does not impact process lifecycle management (you must be using some higher-level component to tell the system that your process needs to continue running), the connection will break if your process goes away for any reason, etc.

See developer.android.com/reference/android/os/Messenger.html

Overview of Messengers

- Handlers can be used to send & process Messages in a single component
- A Messenger encapsulates access to a Handler in one component
  - A Messenger reference can be passed to a component in another process



public final class

Summary: Inherited Constants | Fields | Ctors | Methods | Inherited Methods | [Expand All]

Added in API level 1

Messenger

extends Object

implements Parcelable

java.lang.Object
  ↳ android.os.Messenger

Class Overview

Reference to a Handler, which others can use to send messages to it. This allows for the implementation of message-based communication across processes, by creating a Messenger pointing to a Handler in one process, and handing that Messenger to another process.

Note: the implementation underneath is just a simple wrapper around a `Binder` that is used to perform the communication. This means semantically you should treat it as such: this class does not impact process lifecycle management (you must be using some higher-level component to tell the system that your process needs to continue running), the connection will break if your process goes away for any reason, etc.

Overview of Messengers

- Handlers can be used to send & process Messages in a single component
- A Messenger encapsulates access to a Handler in one component
  - A Messenger reference can be passed to a component in another process
  - The component can use the reference to send Messages to the Handler

public final class

Summary: Inherited Constants | Fields | Ctors | Methods | Inherited Methods | [Expand All]

Added in API level 1

Messenger

extends Object

implements Parcelable

java.lang.Object

↳ android.os.Messenger

Class Overview

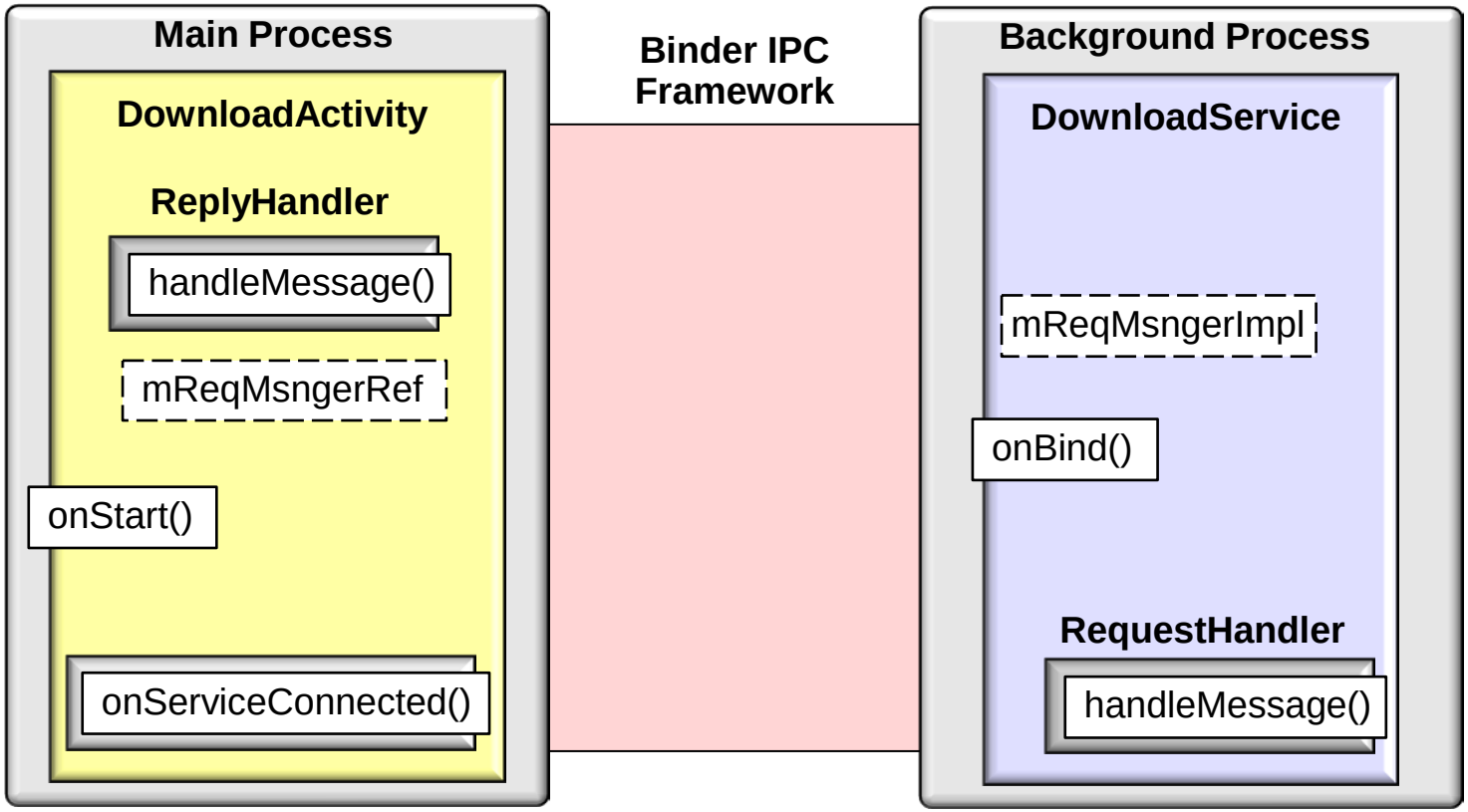
Reference to a Handler, which others can use to send messages to it. This allows for the implementation of message-based communication across processes, by creating a Messenger pointing to a Handler in one process, and handing that Messenger to another process.

Note: the implementation underneath is just a simple wrapper around a `Binder` that is used to perform the communication. This means semantically you should treat it as such: this class does not impact process lifecycle management (you must be using some higher-level component to tell the system that your process needs to continue running), the connection will break if your process goes away for any reason, etc.

Activity to Service Communication Via Messengers

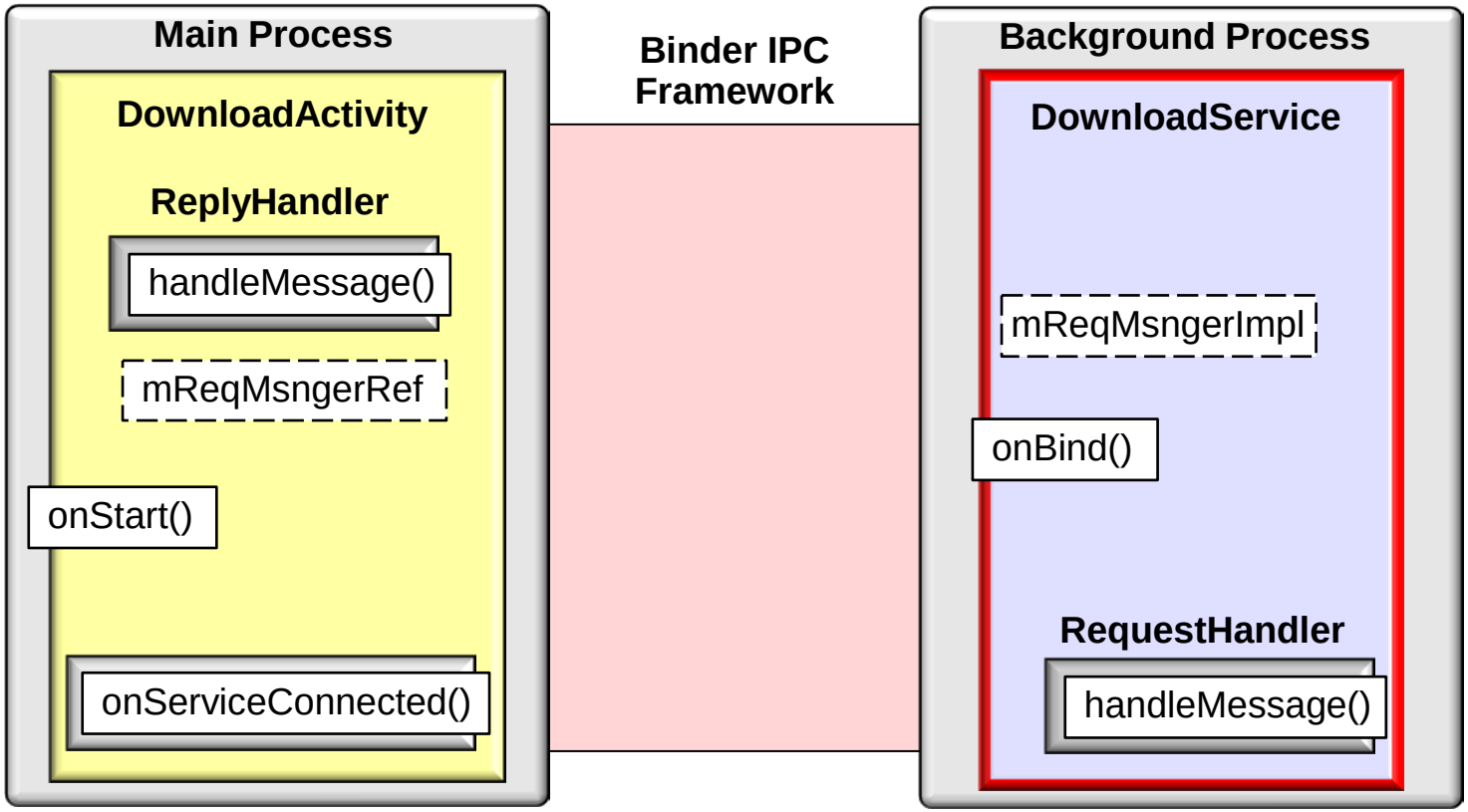
Activity to Service Communication w/Messengers

- Messengers can communicate with both Started & bound Services



Activity to Service Communication w/Messengers

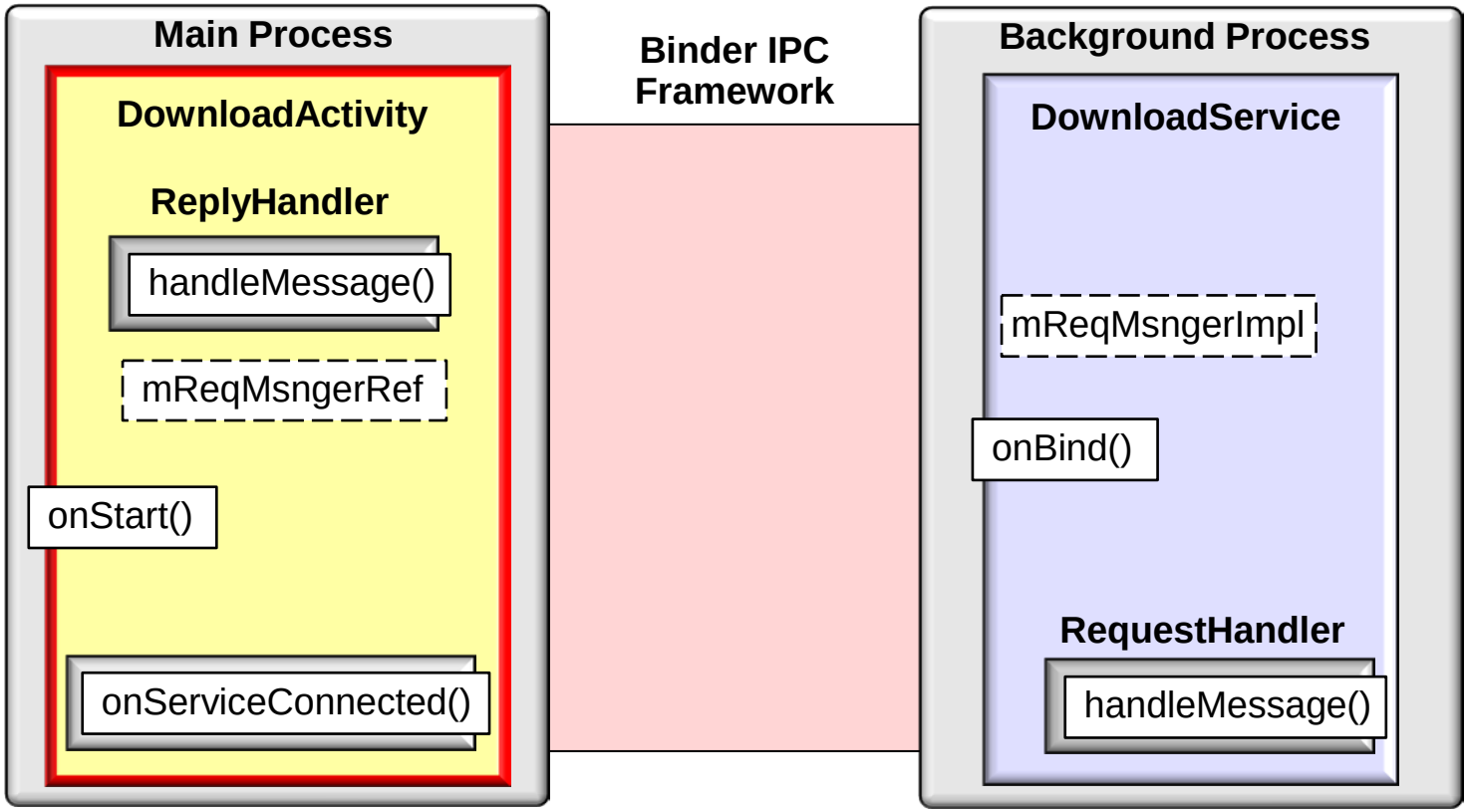
- Messengers can communicate with both Started & bound Services



A Bounded Service can contain a Request Messenger that encapsulates a Request Handler

Activity to Service Communication w/Messengers

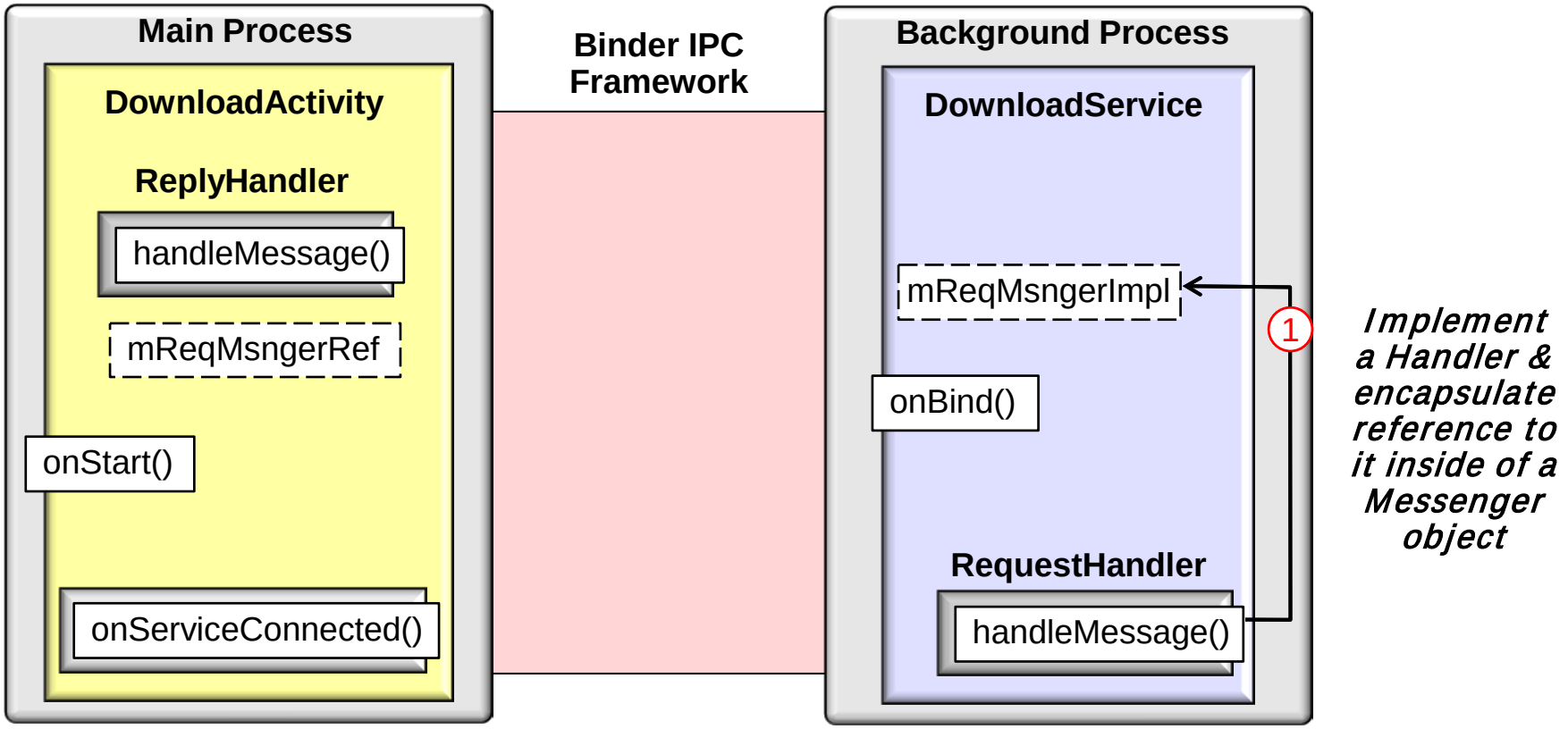
- Messengers can communicate with both Started & bound Services



An Activity can contain a Reply Messenger that encapsulates a Reply Handler

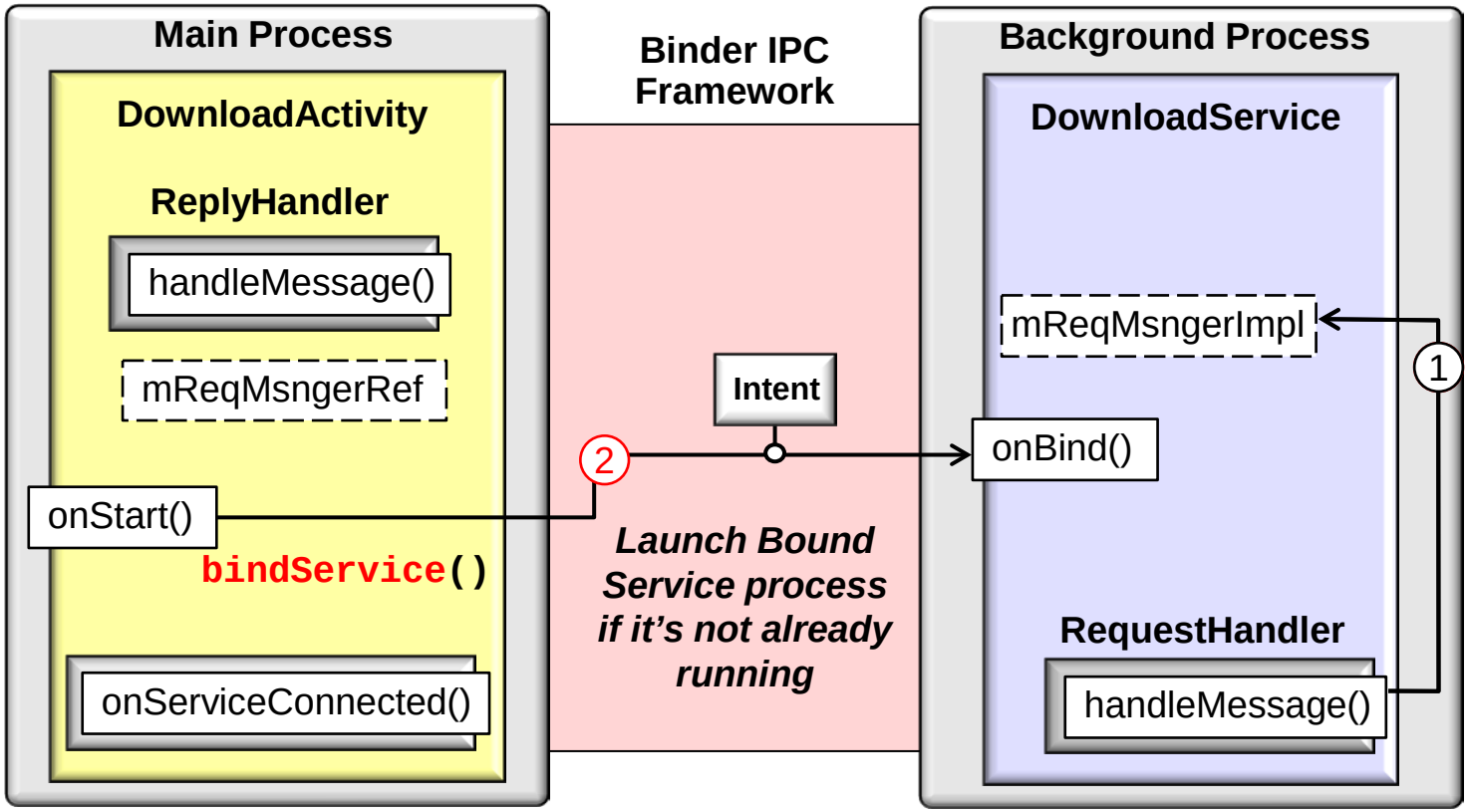
Activity to Service Communication w/Messengers

- Messengers can communicate with both Started & bound Services
  - e.g., consider a bound Service implementation of the DownloadService



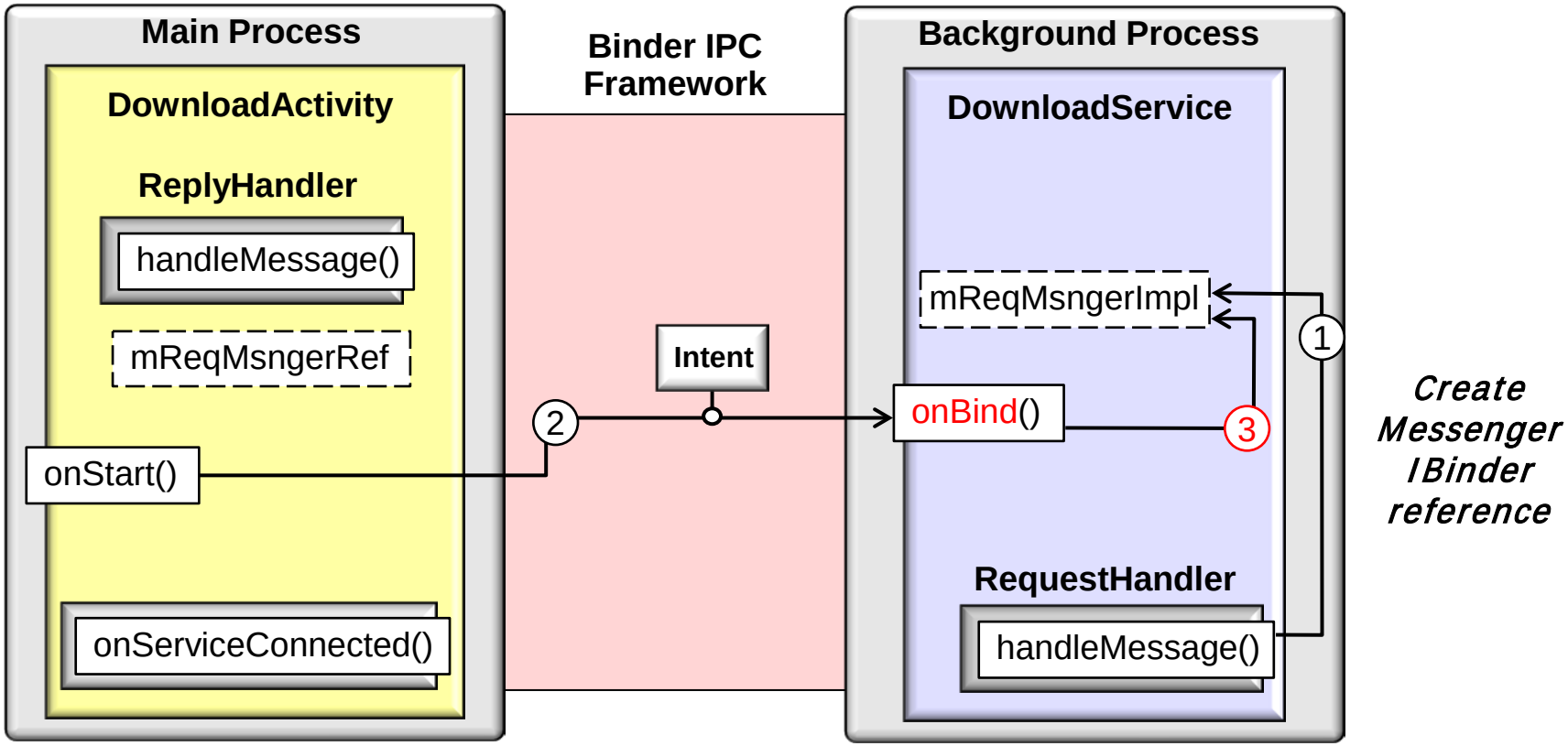
Activity to Service Communication w/Messengers

- Messengers can communicate with both Started & bound Services
  - e.g., consider a bound Service implementation of the DownloadService



Activity to Service Communication w/Messengers

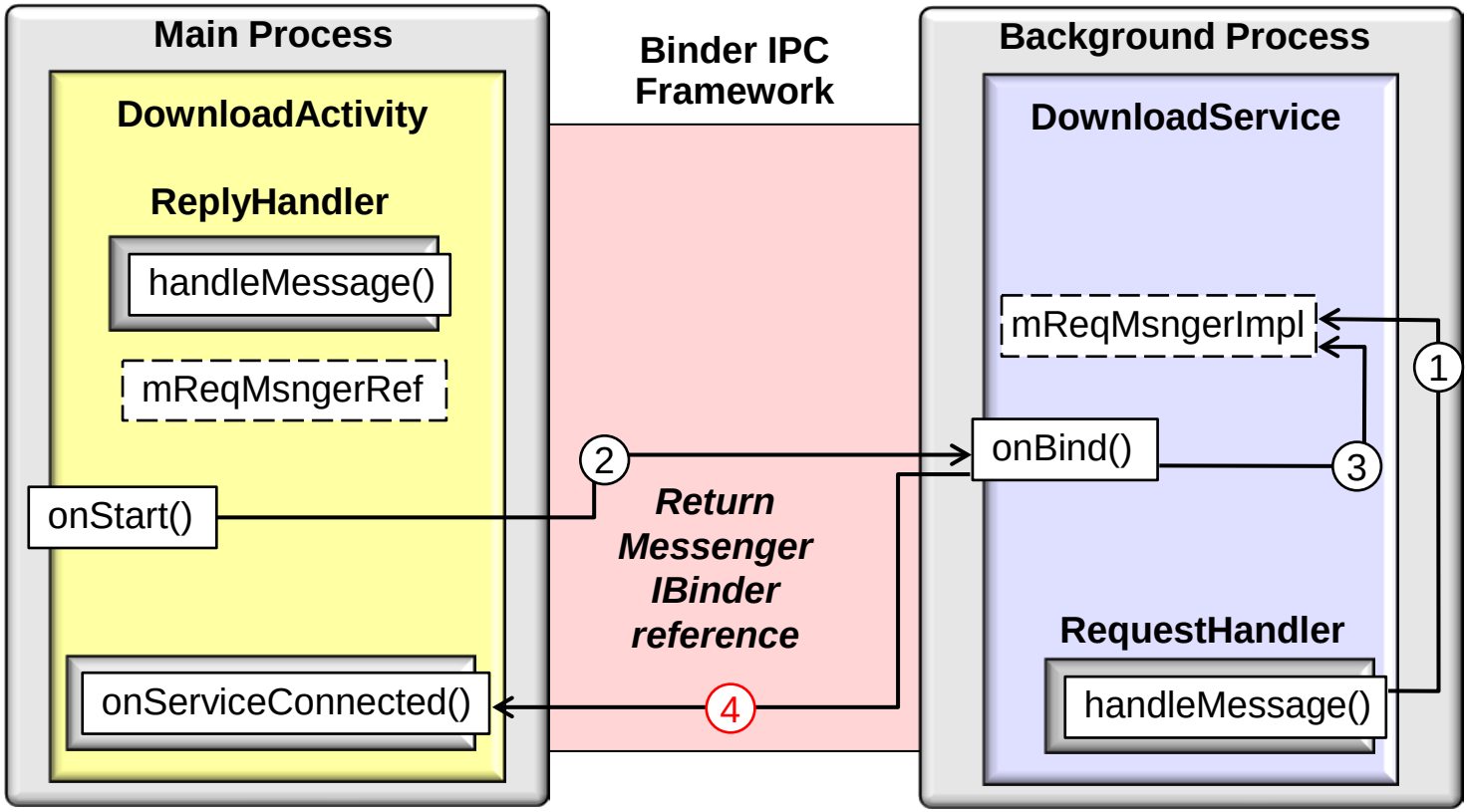
- Messengers can communicate with both Started & bound Services
  - e.g., consider a bound Service implementation of the DownloadService



See developer.android.com/reference/android/os/IBinder.html

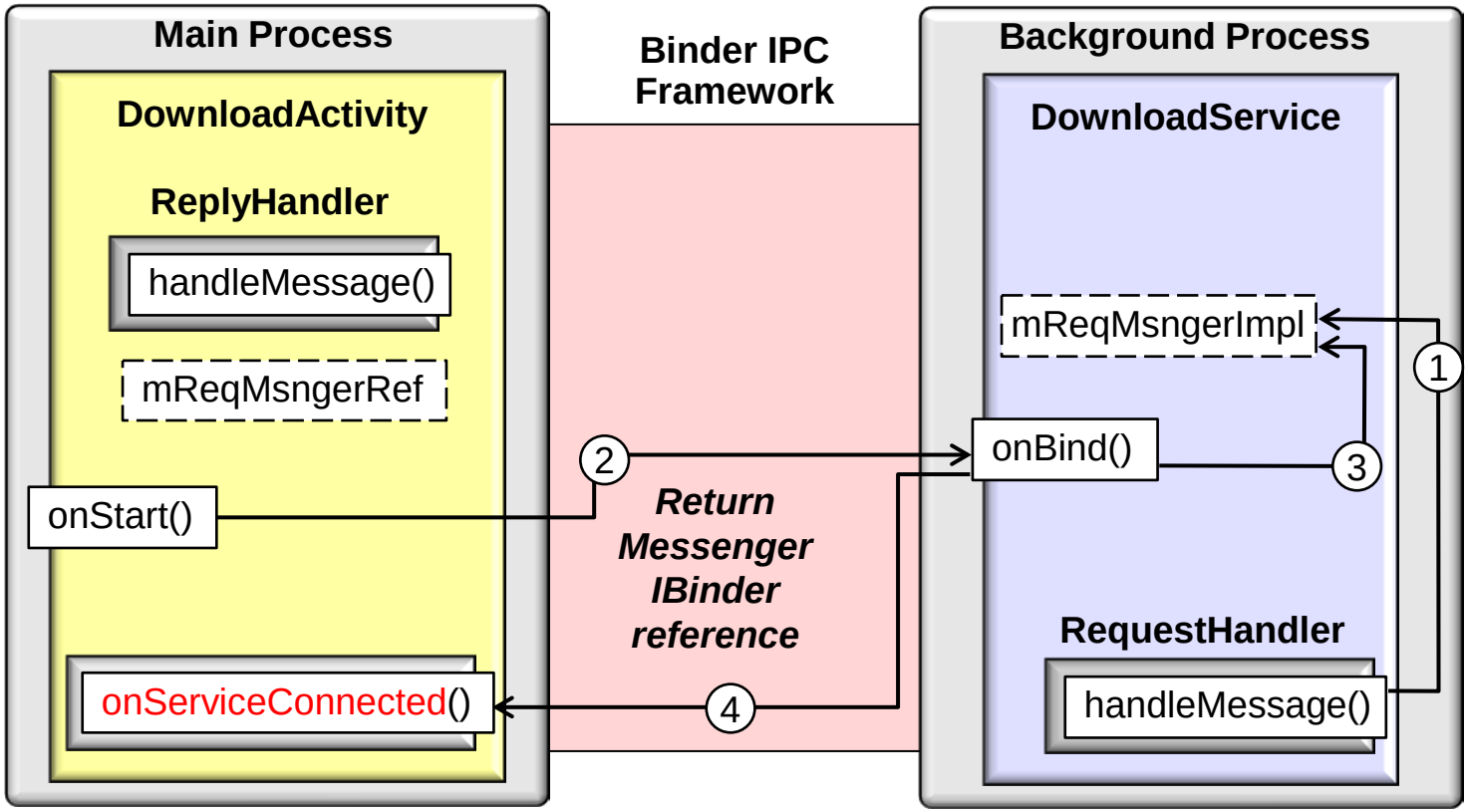
Activity to Service Communication w/Messengers

- Messengers can communicate with both Started & bound Services
  - e.g., consider a bound Service implementation of the DownloadService



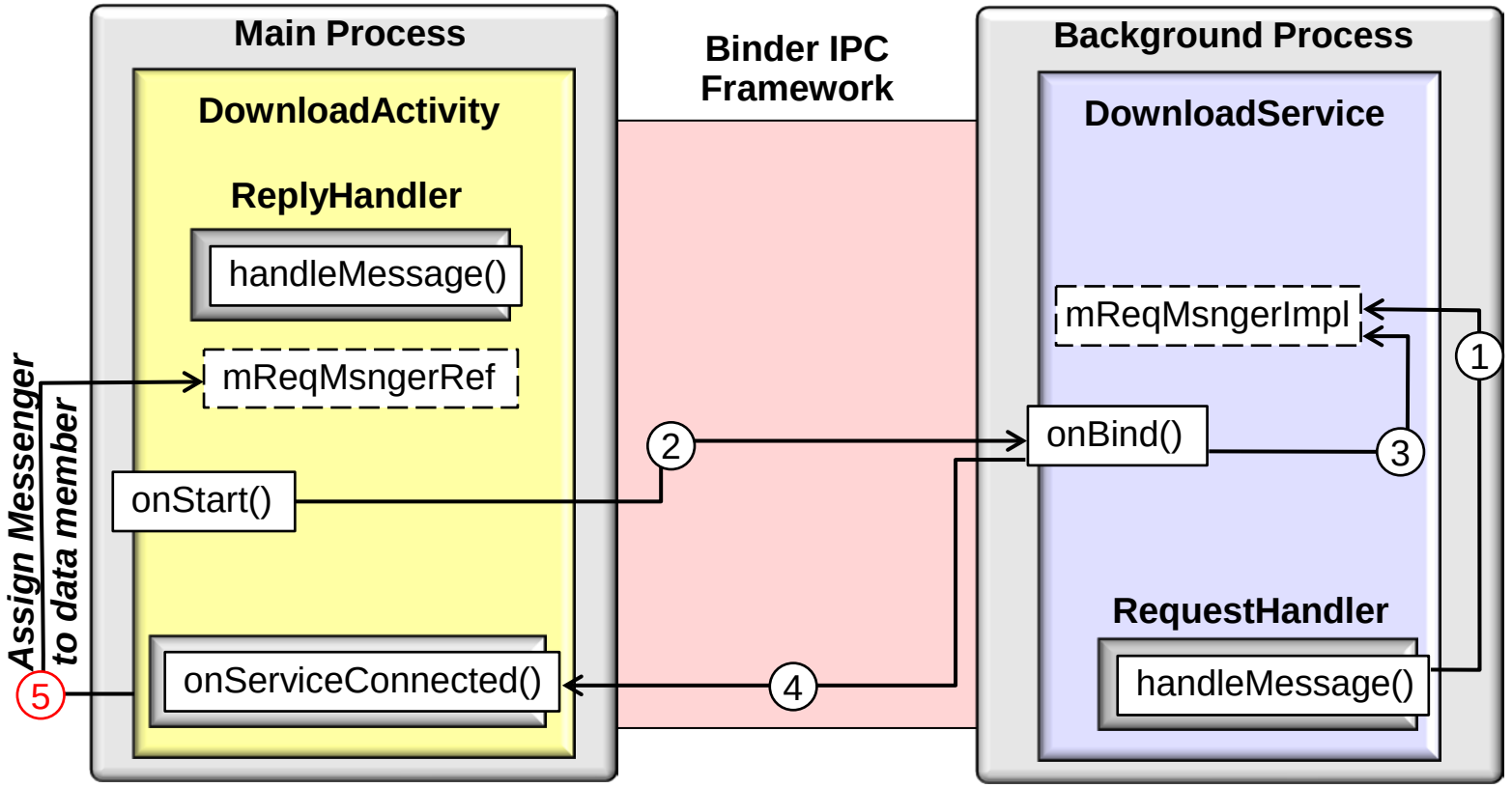
Activity to Service Communication w/Messengers

- Messengers can communicate with both Started & bound Services
  - e.g., consider a bound Service implementation of the DownloadService



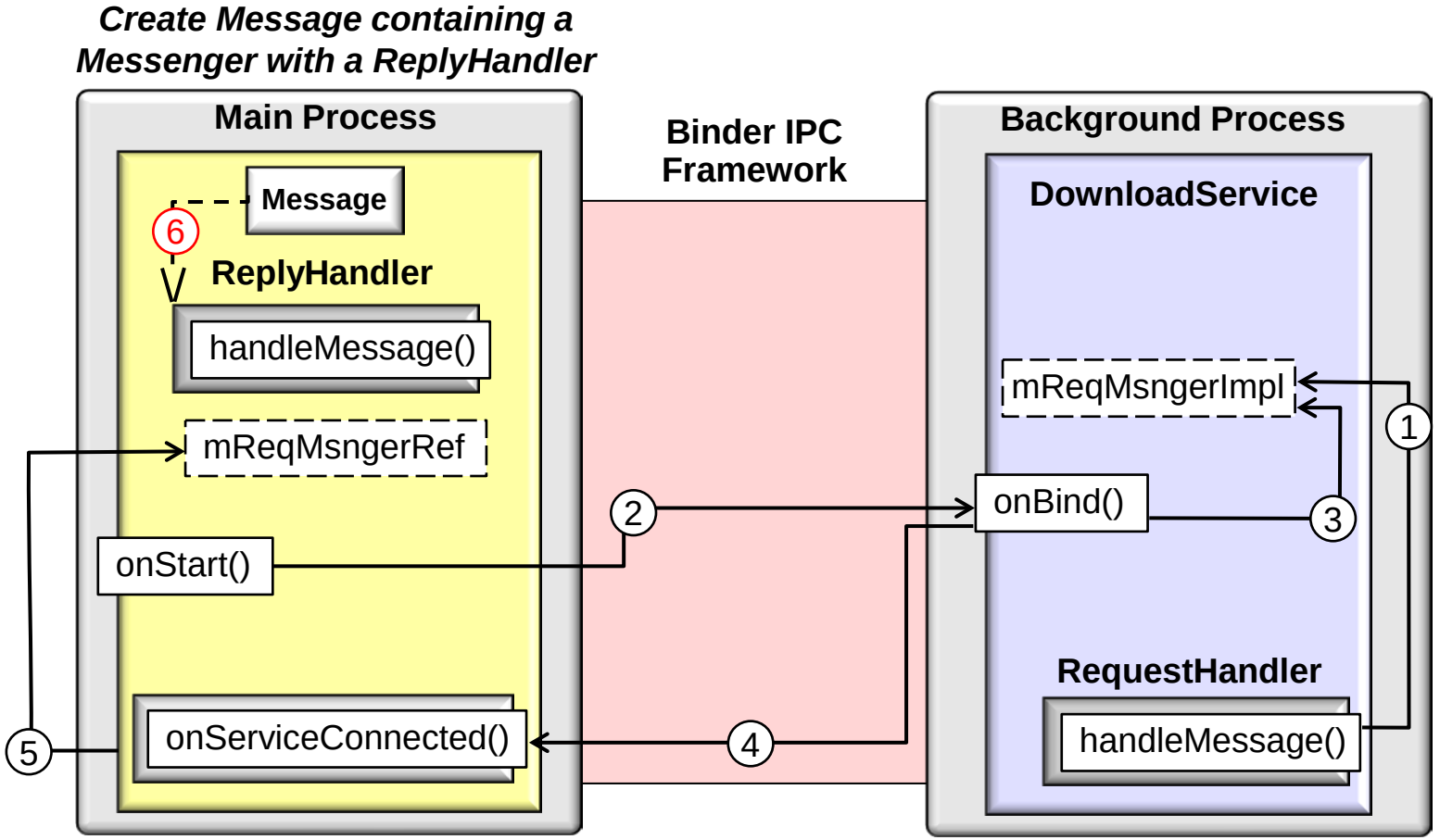
Activity to Service Communication w/Messengers

- Messengers can communicate with both Started & bound Services
  - e.g., consider a bound Service implementation of the DownloadService



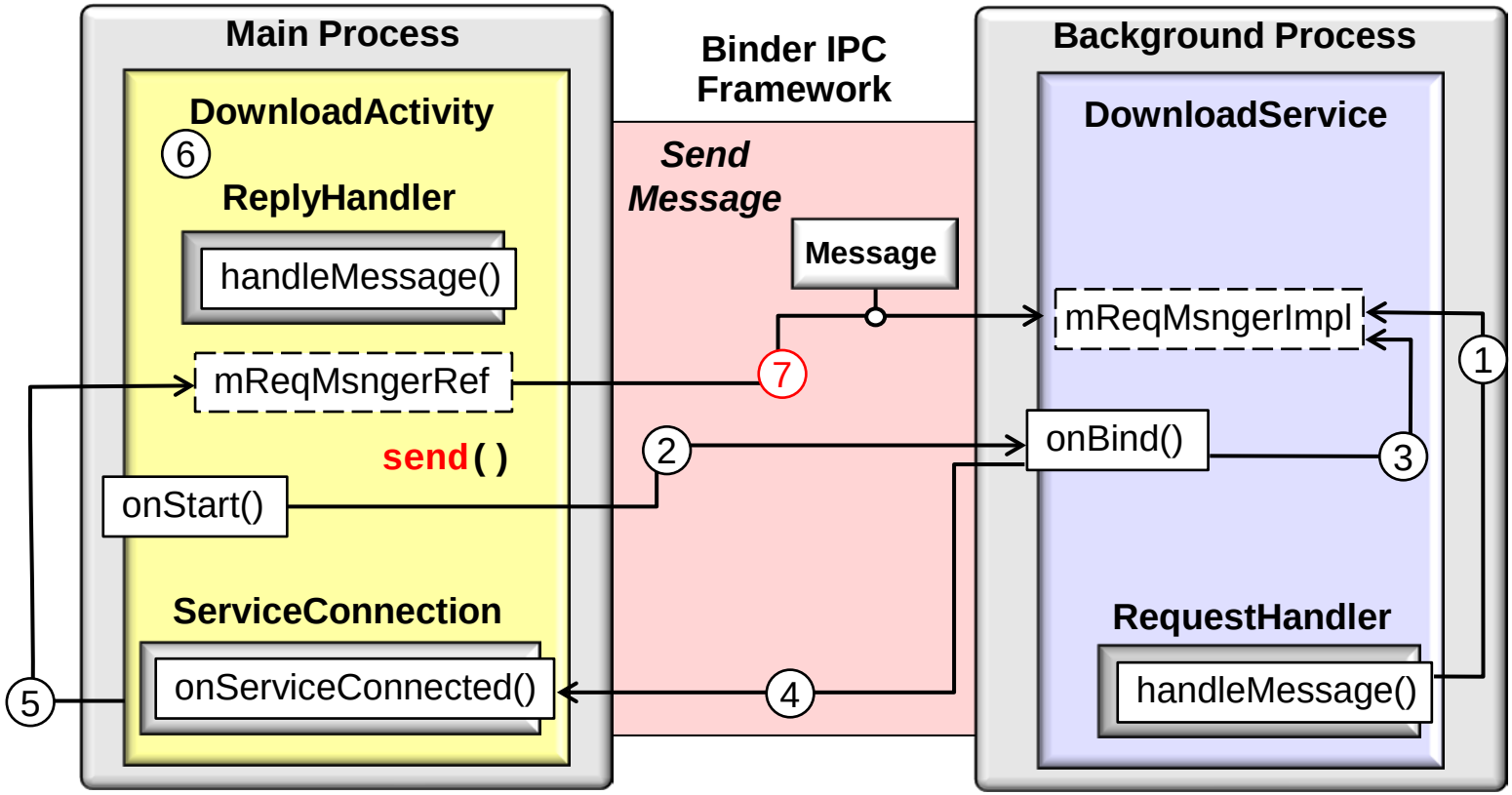
Activity to Service Communication w/Messengers

- Messengers can communicate with both Started & bound Services
  - e.g., consider a bound Service implementation of the DownloadService



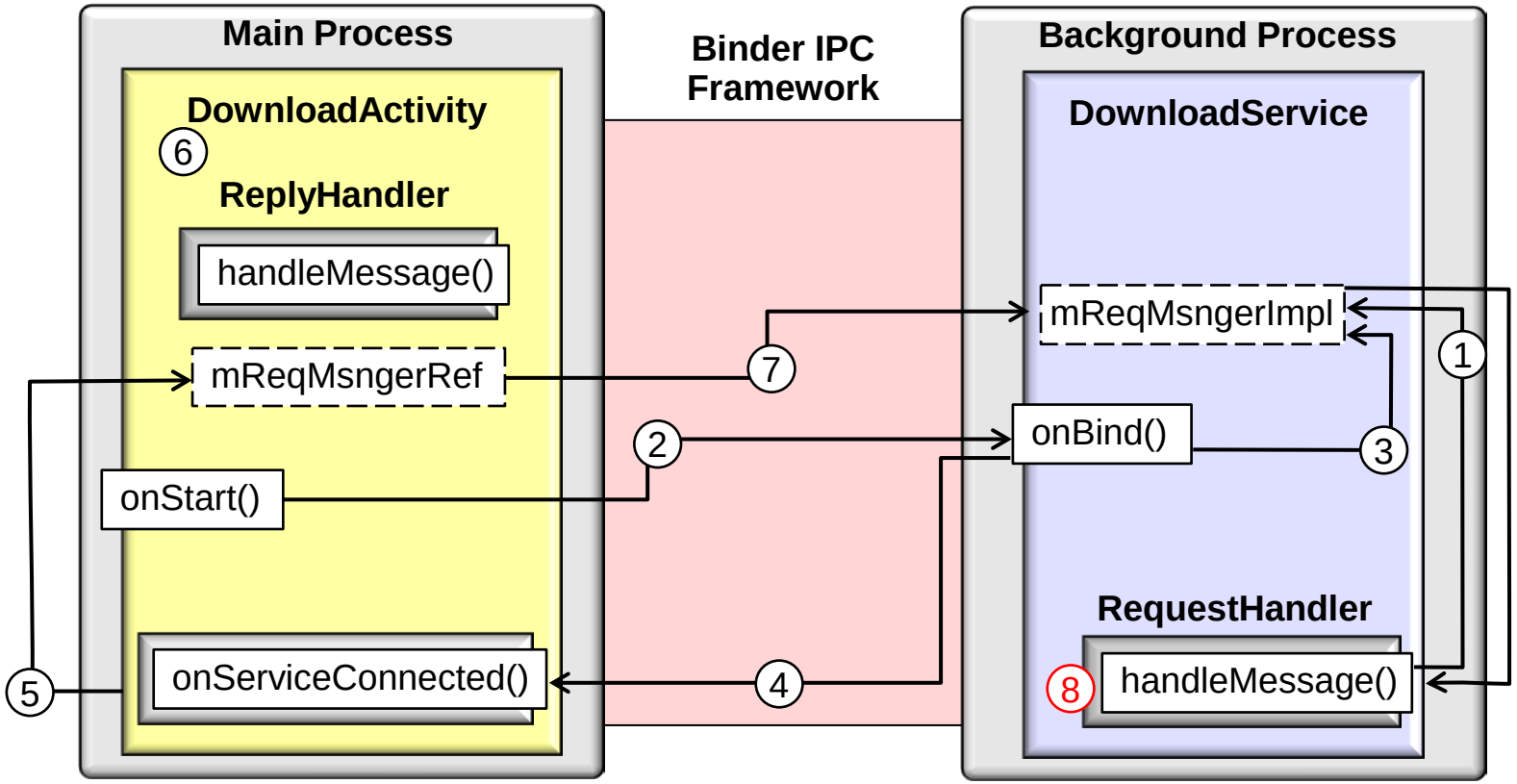
Activity to Service Communication w/Messengers

- Messengers can communicate with both Started & bound Services
  - e.g., consider a bound Service implementation of the DownloadService



Activity to Service Communication w/Messengers

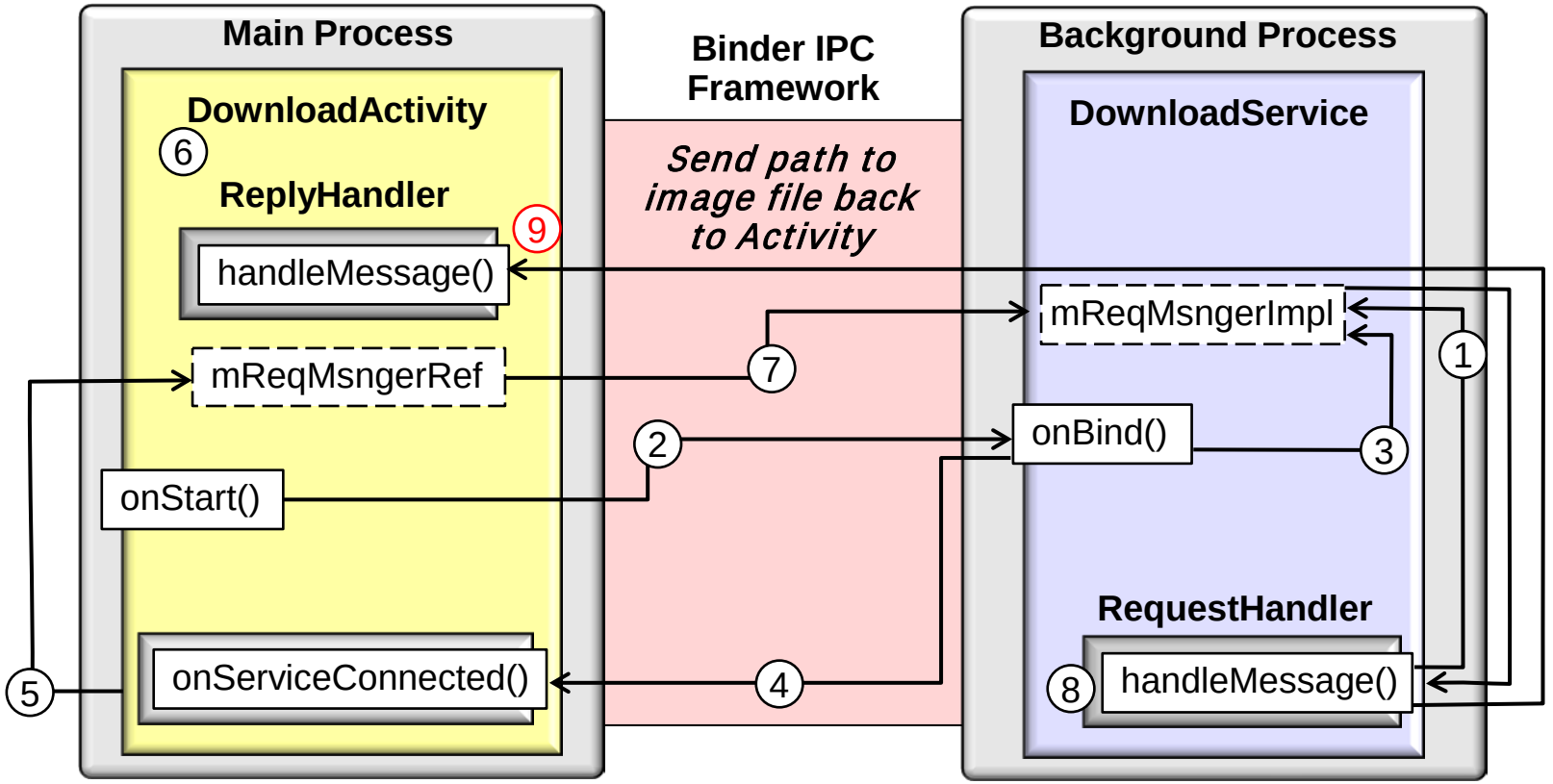
- Messengers can communicate with both Started & bound Services
  - e.g., consider a bound Service implementation of the DownloadService



Process Message sent via Messenger to download & store an mage

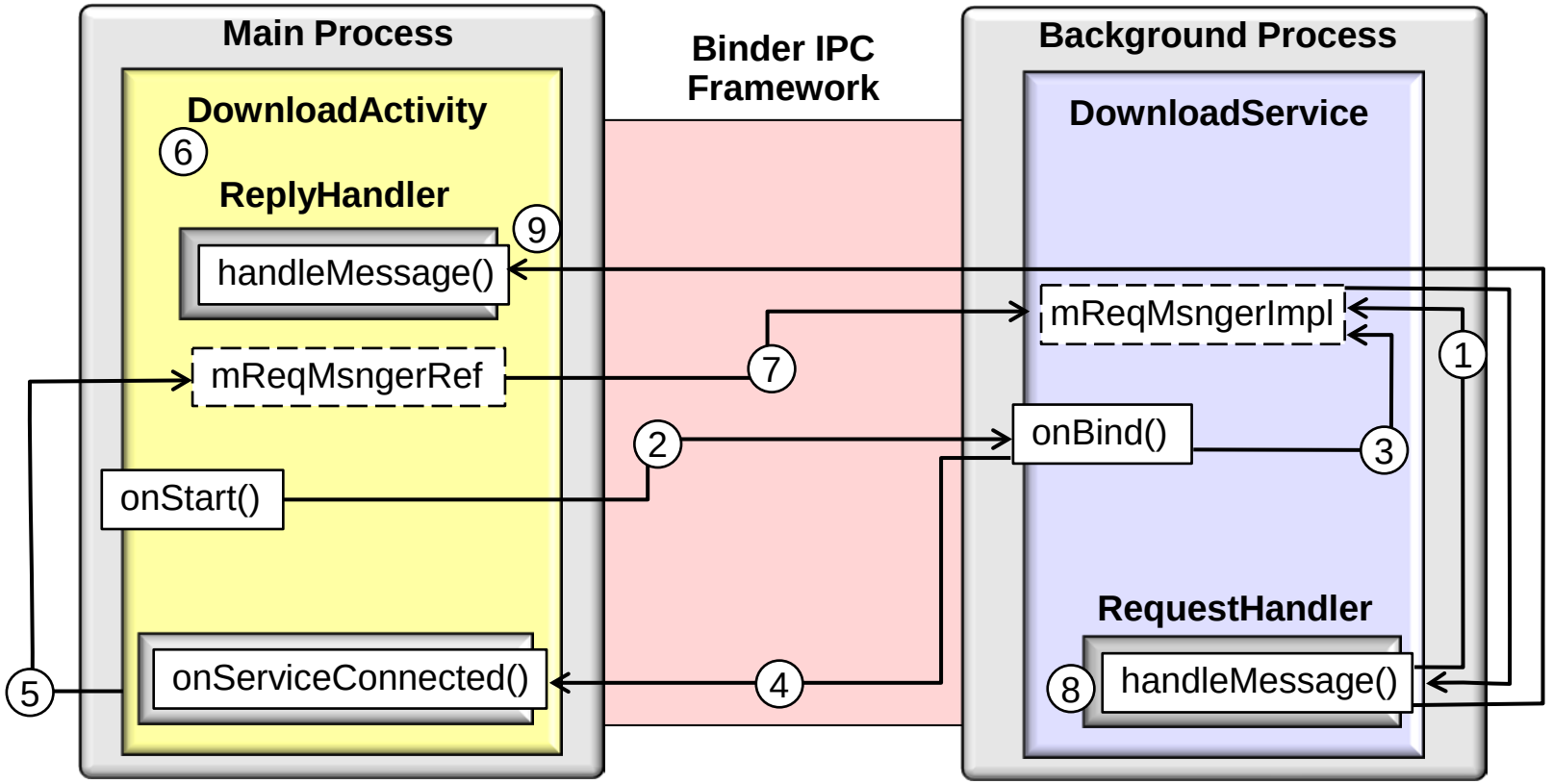
Activity to Service Communication w/Messengers

- Messengers can communicate with both Started & bound Services
  - e.g., consider a bound Service implementation of the DownloadService



Activity to Service Communication w/Messengers

- Messengers can communicate with both Started & bound Services
  - e.g., consider a bound Service implementation of the DownloadService

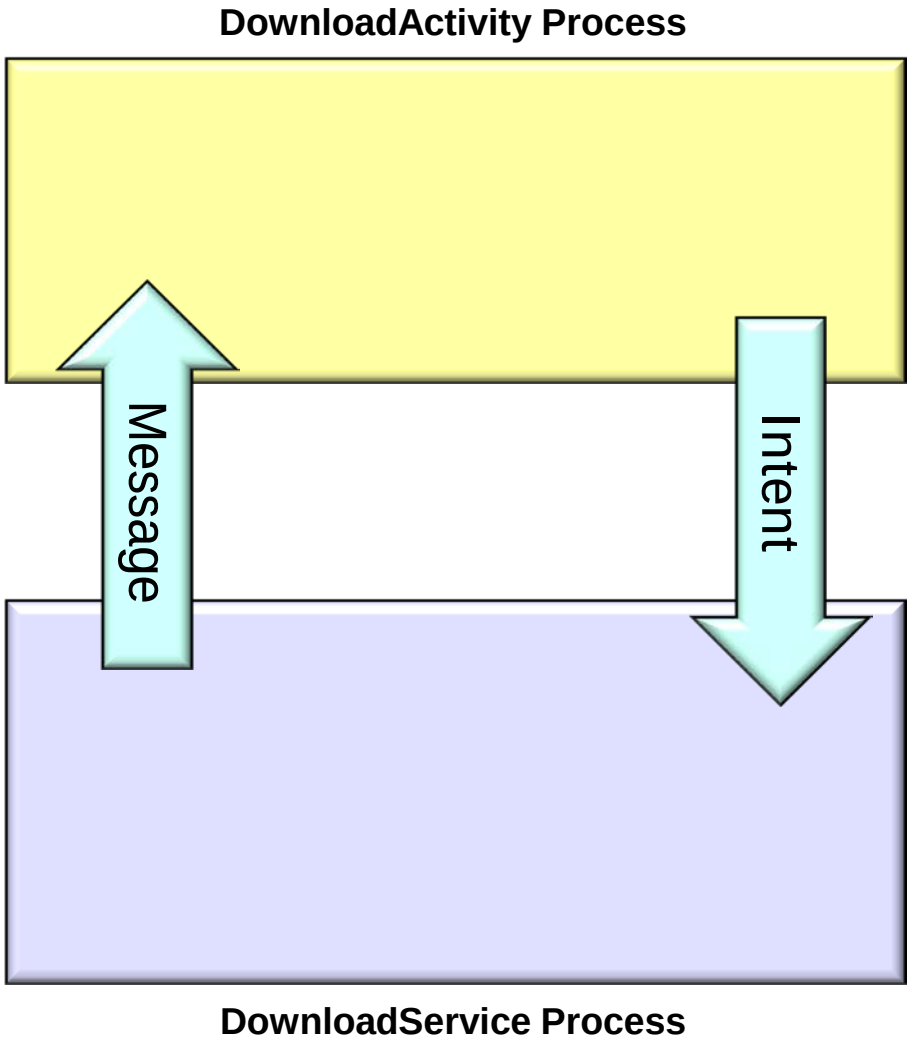


See upcoming parts on “Programming Bound Services with Messengers”

Service to Activity Communication with Messengers

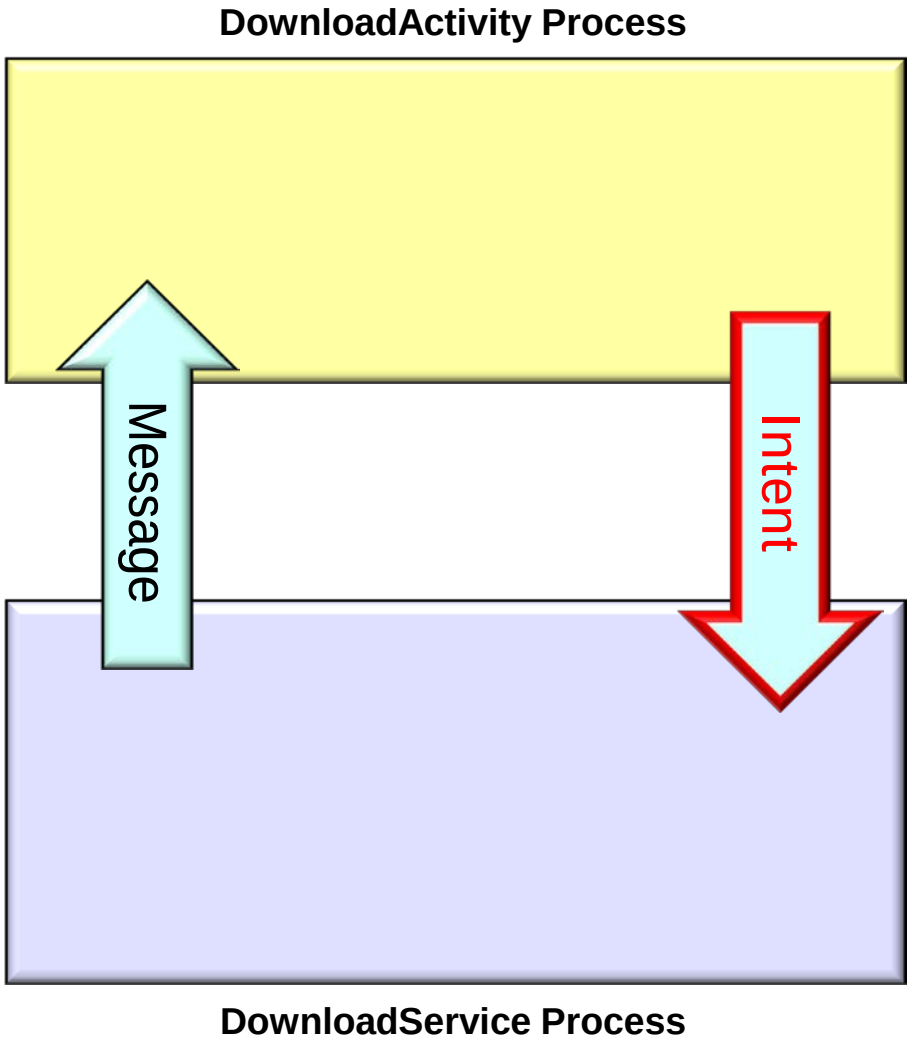
Service to Activity Communication w/Messengers

- A started Service can perform IPC with an Activity via a Messenger



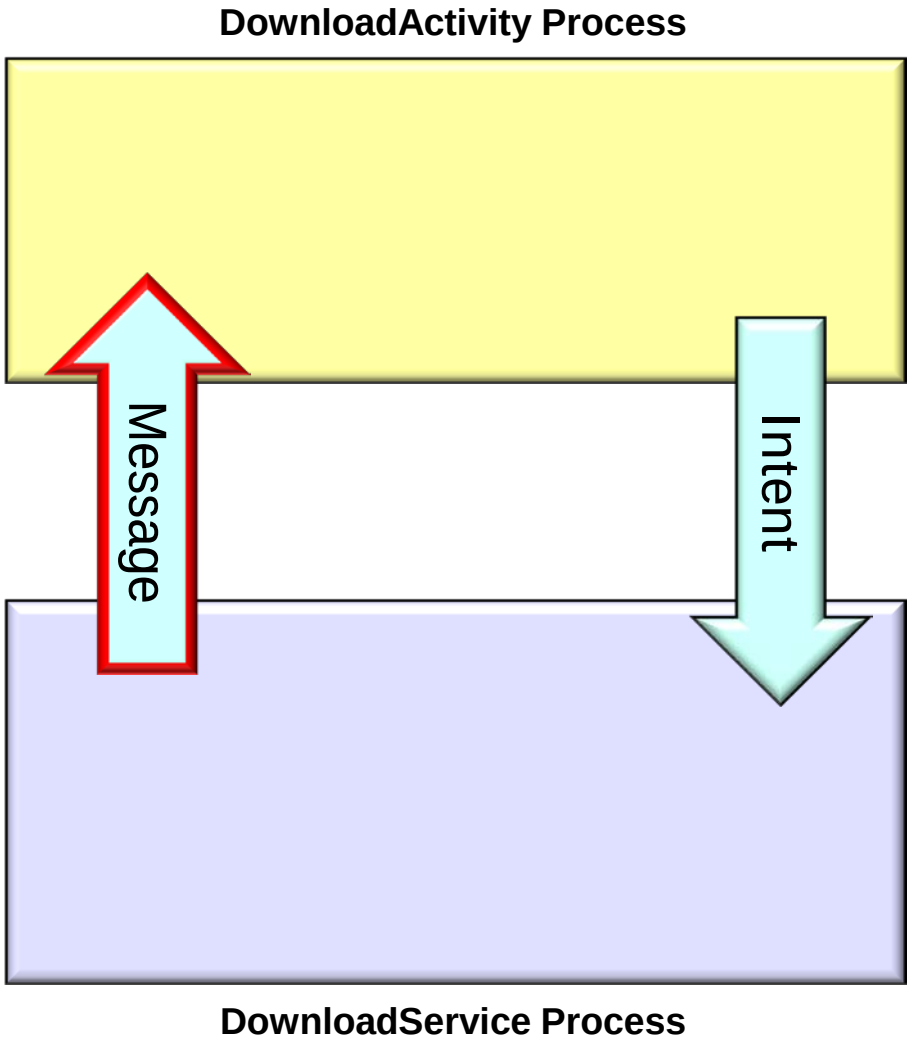
Service to Activity Communication w/Messengers

- A started Service can perform IPC with an Activity via a Messenger



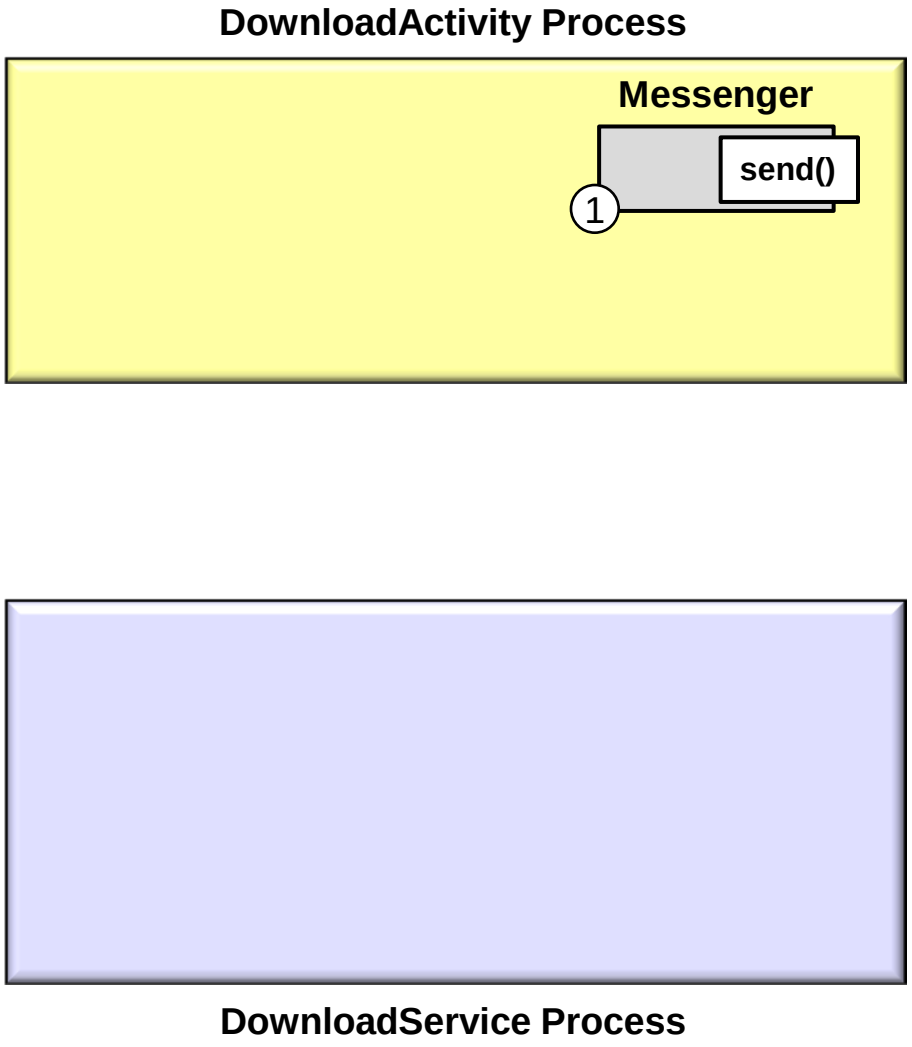
Service to Activity Communication w/Messengers

- A started Service can perform IPC with an Activity via a Messenger



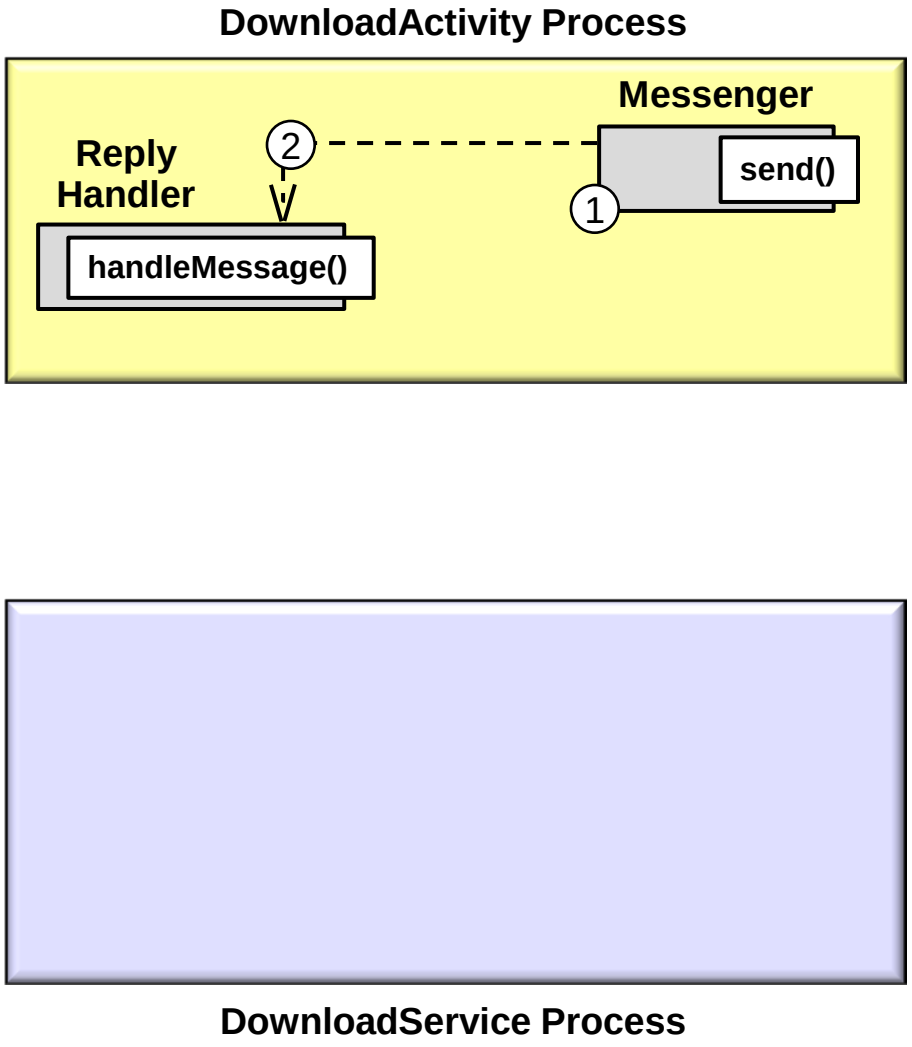
Service to Activity Communication w/Messengers

- A started Service can perform IPC with an Activity via a Messenger
  - The Activity creates a Messenger object that's encapsulates a Reply Handler object



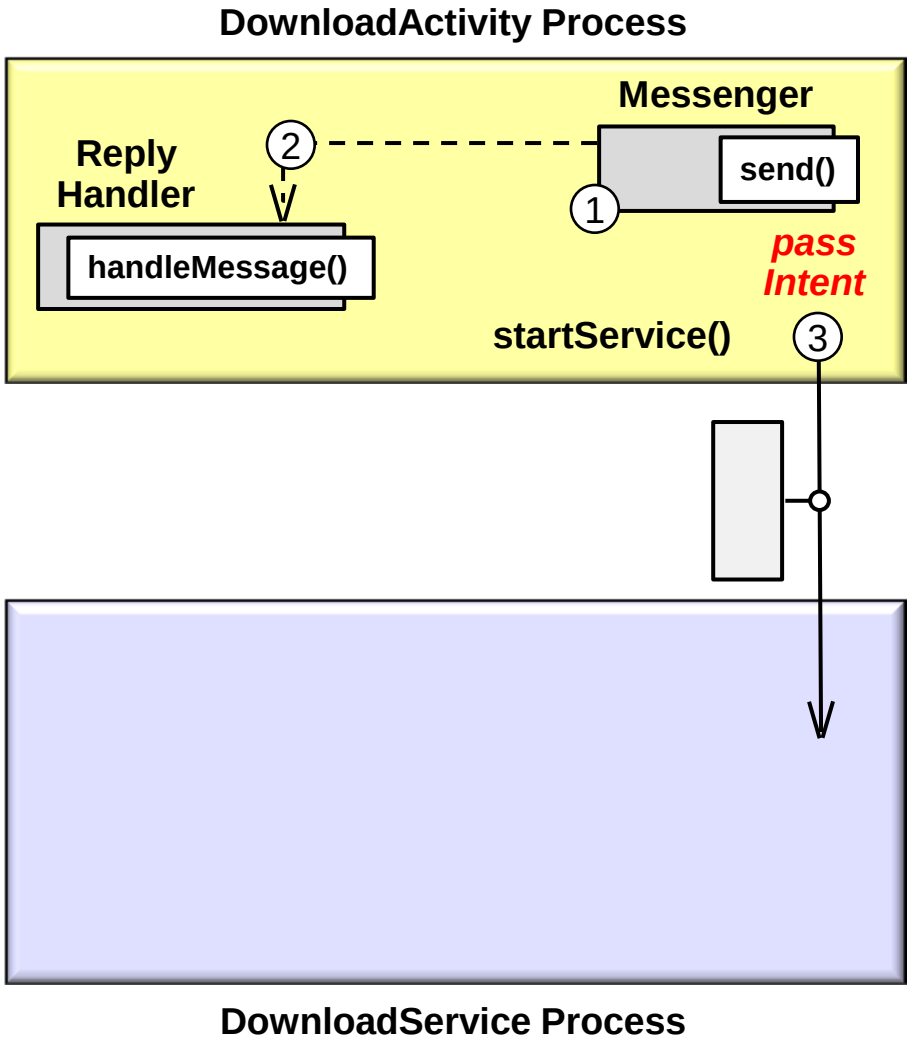
Service to Activity Communication w/Messengers

- A started Service can perform IPC with an Activity via a Messenger
  - The Activity creates a Messenger object that's encapsulates a Reply Handler object



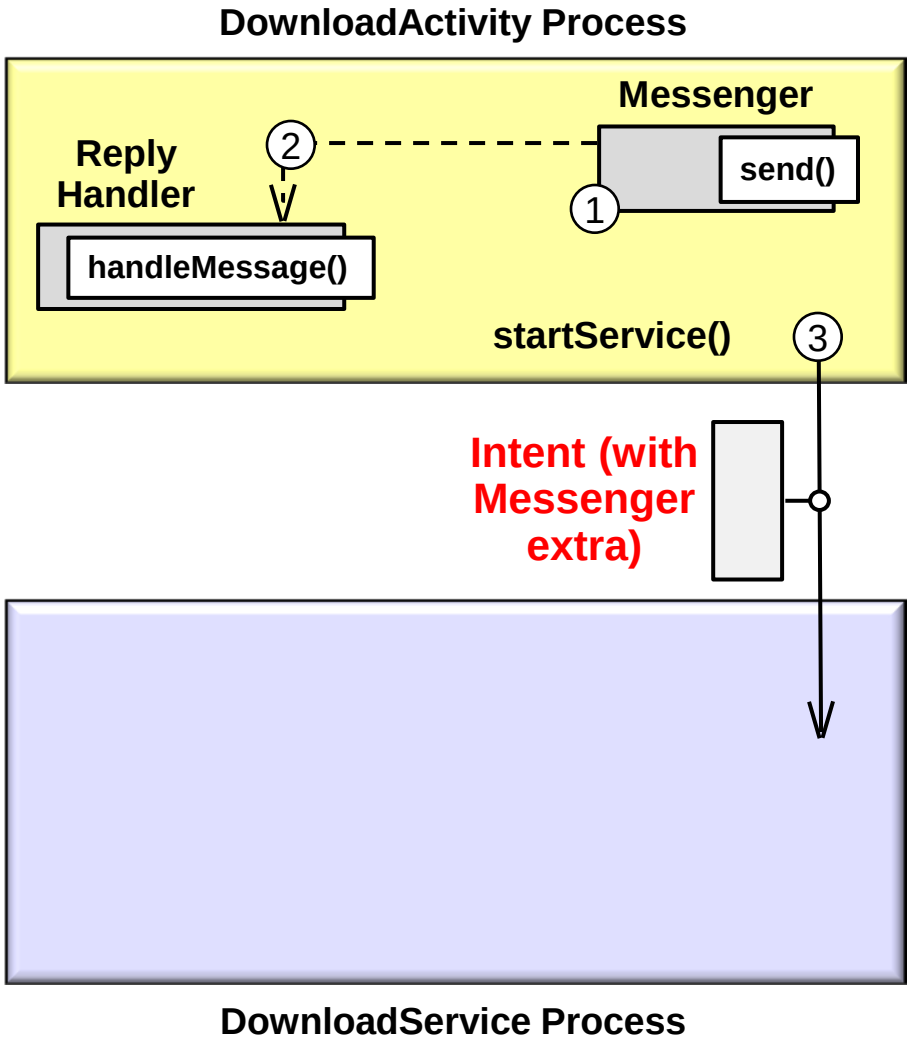
Service to Activity Communication w/Messengers

- A started Service can perform IPC with an Activity via a Messenger
  - The Activity creates a Messenger object that's encapsulates a Reply Handler object
  - A reference to a Messenger is then passed to the started Service as an "extra" to an Intent



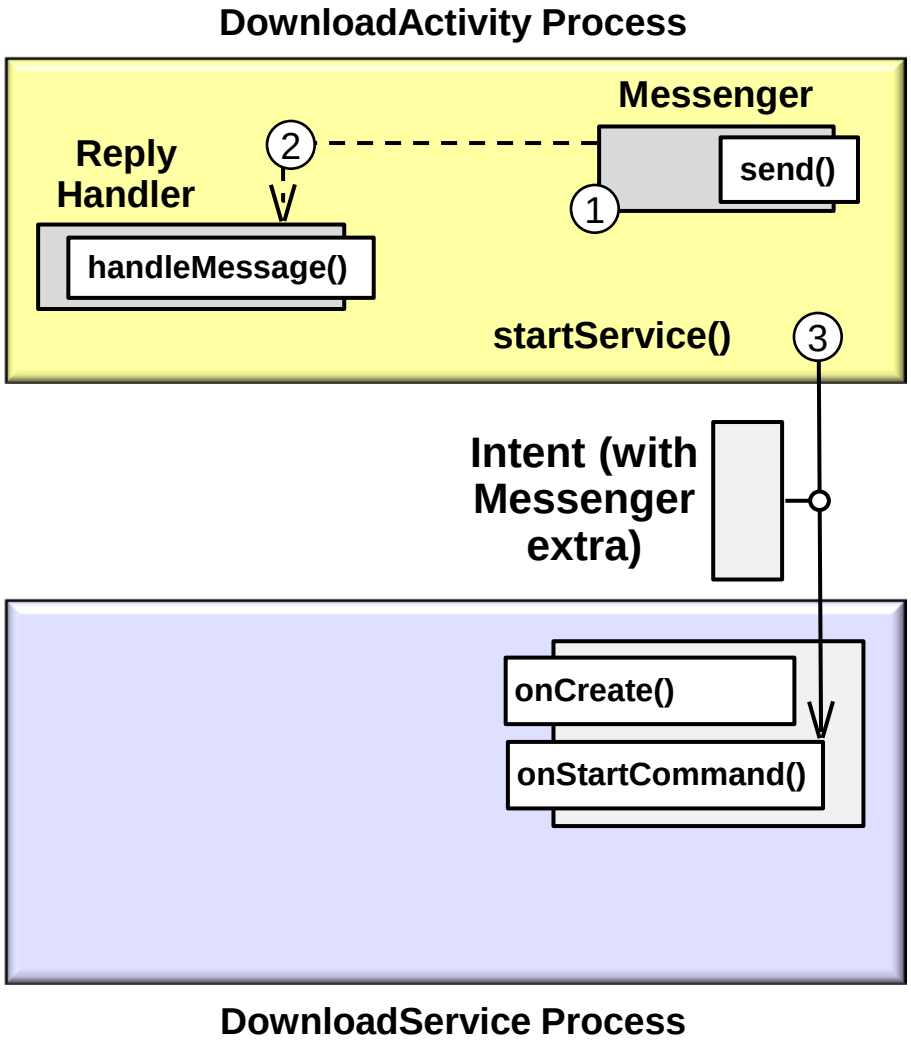
Service to Activity Communication w/Messengers

- A started Service can perform IPC with an Activity via a Messenger
  - The Activity creates a Messenger object that's encapsulates a Reply Handler object
  - A reference to a Messenger is then passed to the started Service as an "extra" to an Intent



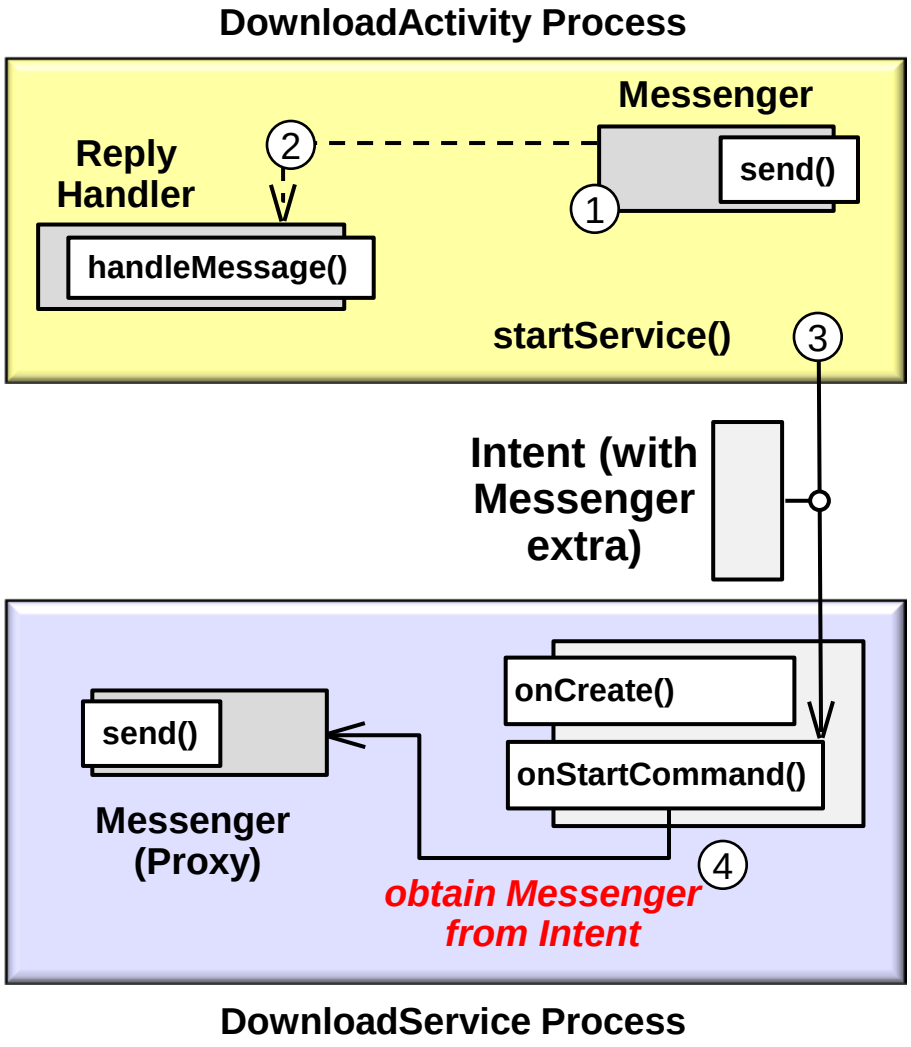
Service to Activity Communication w/Messengers

- A started Service can perform IPC with an Activity via a Messenger
- A started Service typically does three things when it receives an Intent



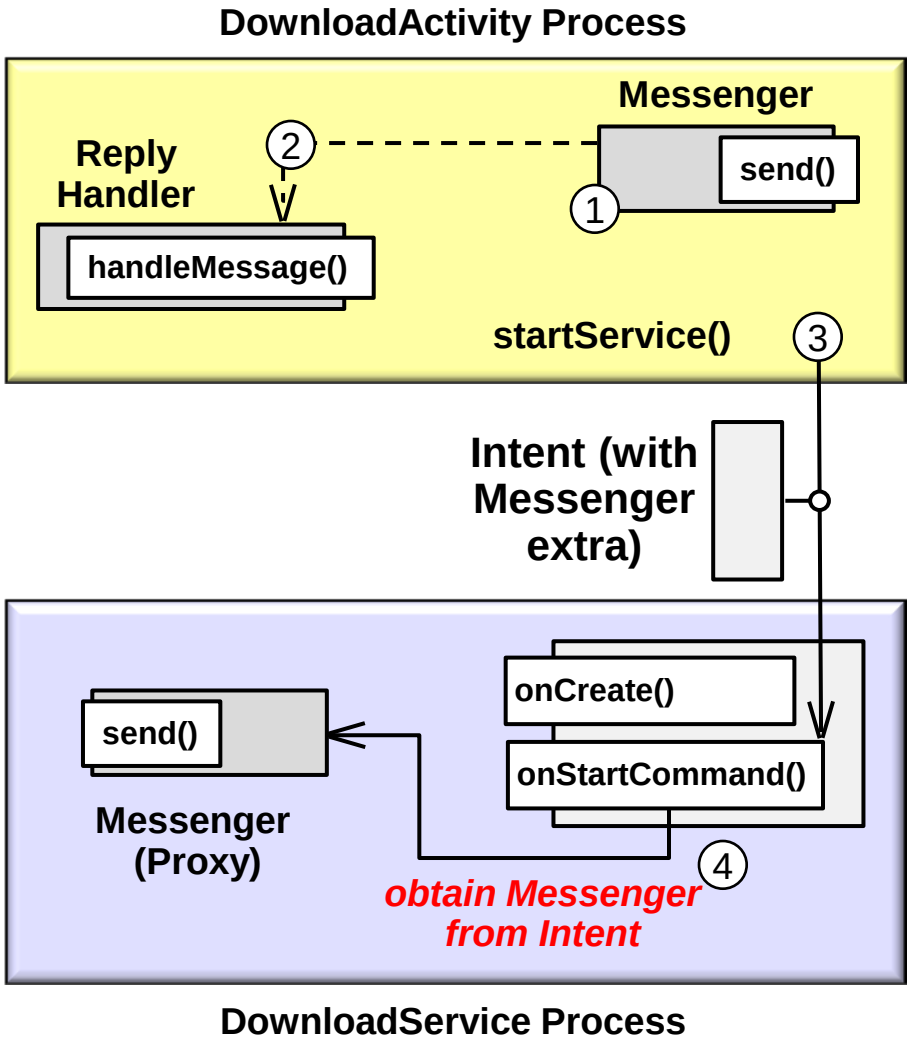
Service to Activity Communication w/Messengers

- A started Service can perform IPC with an Activity via a Messenger
- A started Service typically does three things when it receives an Intent
  - Obtains Messenger from Intent



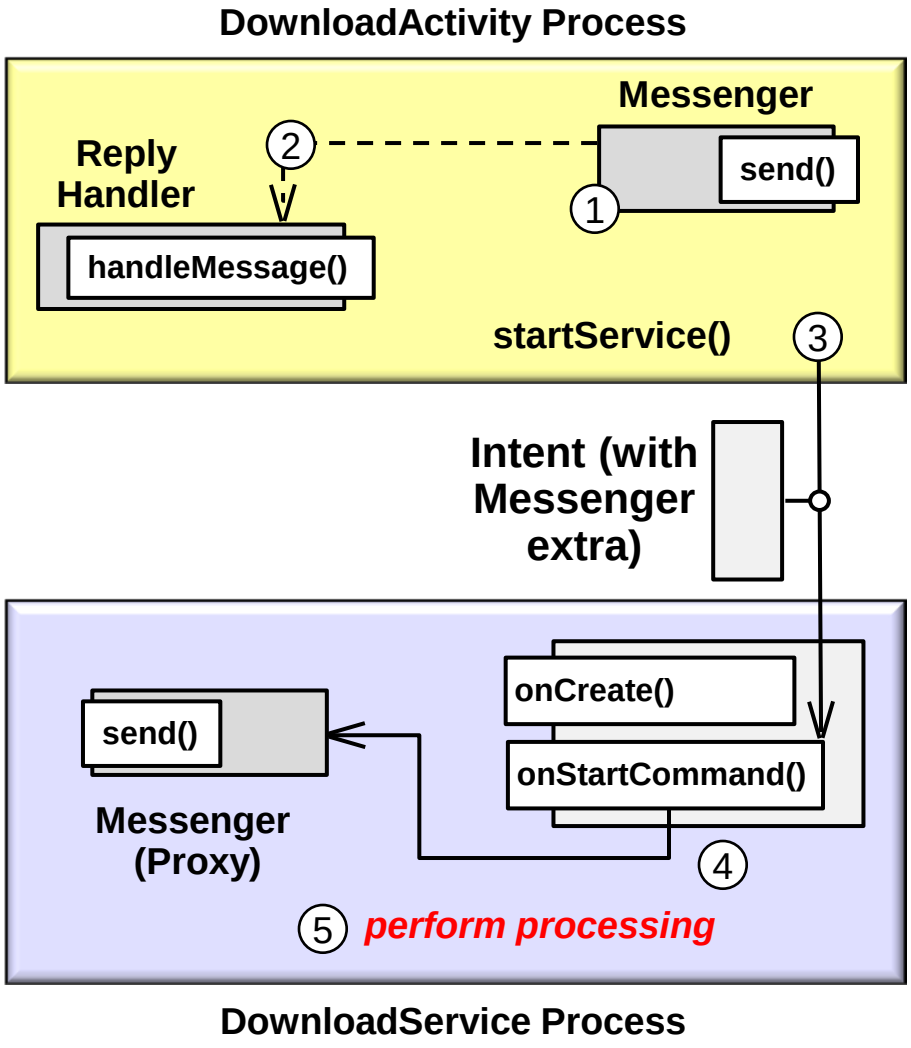
Service to Activity Communication w/Messengers

- A started Service can perform IPC with an Activity via a Messenger
- A started Service typically does three things when it receives an Intent
  - Obtains Messenger from Intent
    - e.g., extract it from the "extra" placed in the Intent by the DownloadActivity



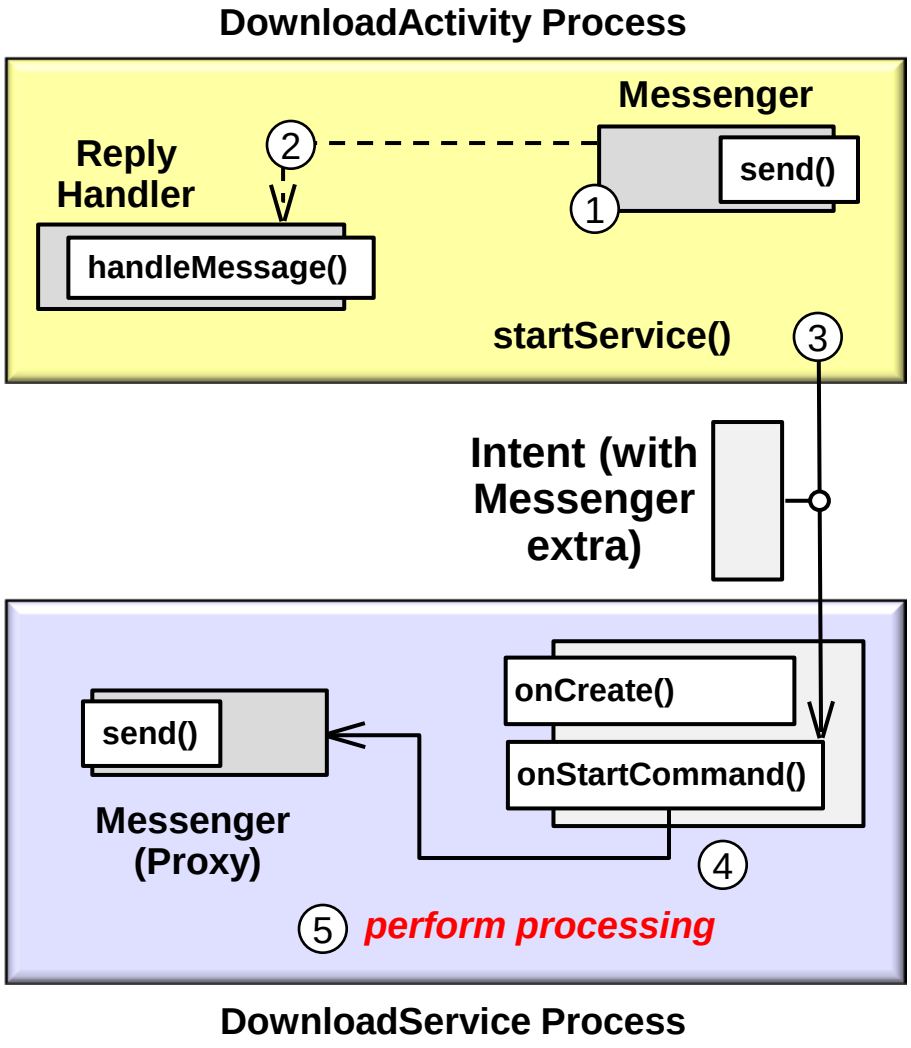
Service to Activity Communication w/Messengers

- A started Service can perform IPC with an Activity via a Messenger
- A started Service typically does three things when it receives an Intent
  - Obtains Messenger from Intent
  - Performs some processing



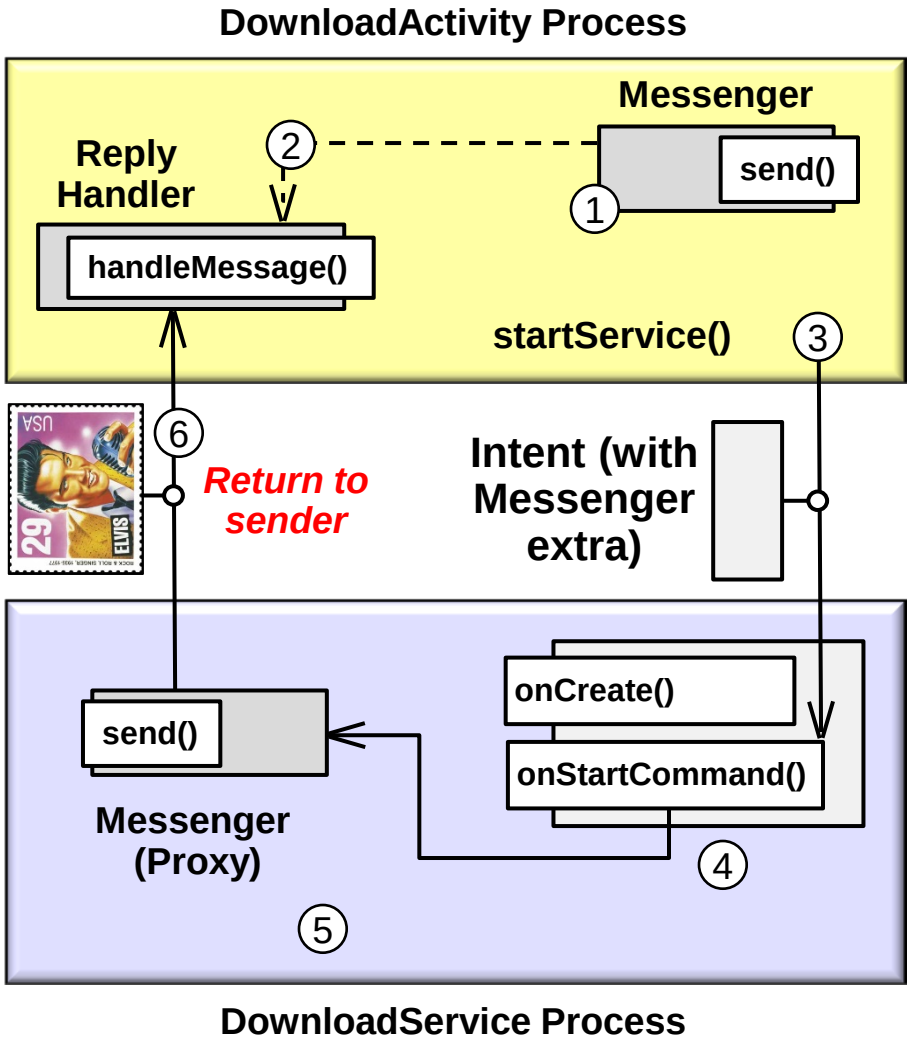
Service to Activity Communication w/Messengers

- A started Service can perform IPC with an Activity via a Messenger
- A started Service typically does three things when it receives an Intent
  - Obtains Messenger from Intent
  - Performs some processing
    - e.g., retrieves an image from a remote server



Service to Activity Communication w/Messengers

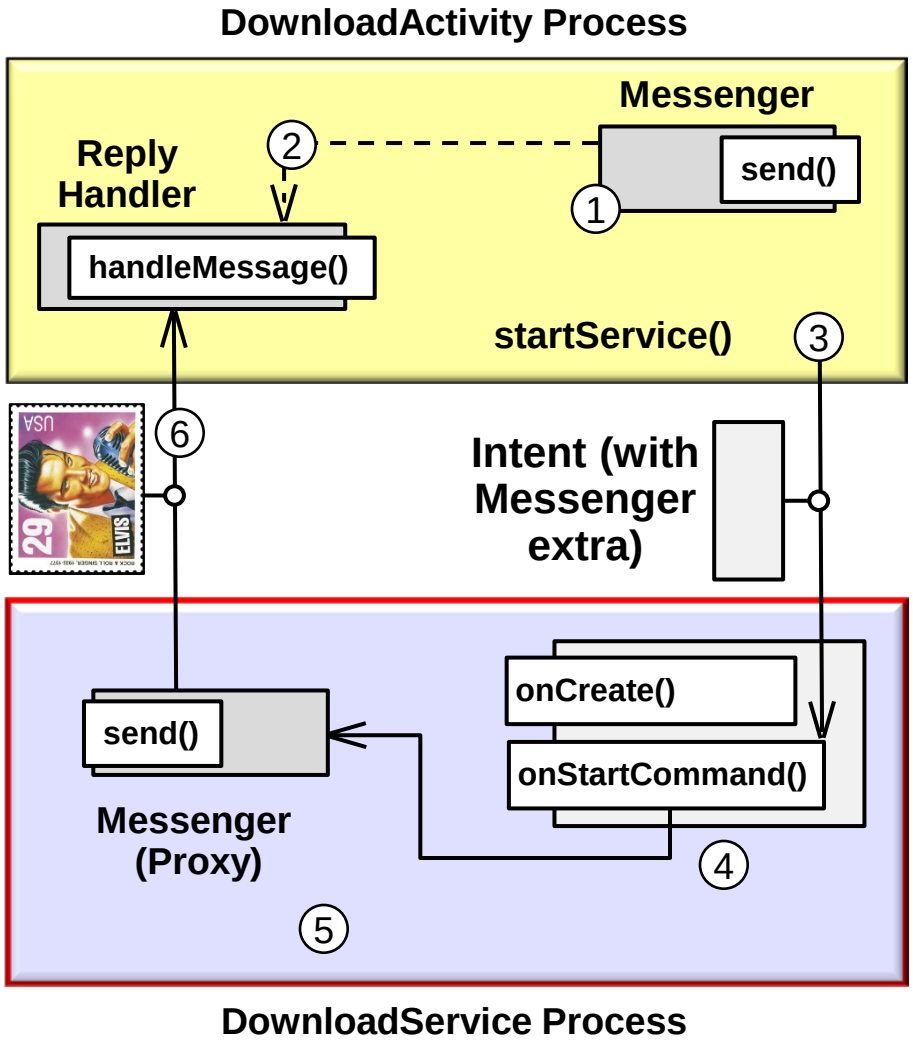
- A started Service can perform IPC with an Activity via a Messenger
- A started Service typically does three things when it receives an Intent
  - Obtains Messenger from Intent
  - Performs some processing
  - Returns the results back to the sender process



The ReplyHandler can reside in a different process than the Service!

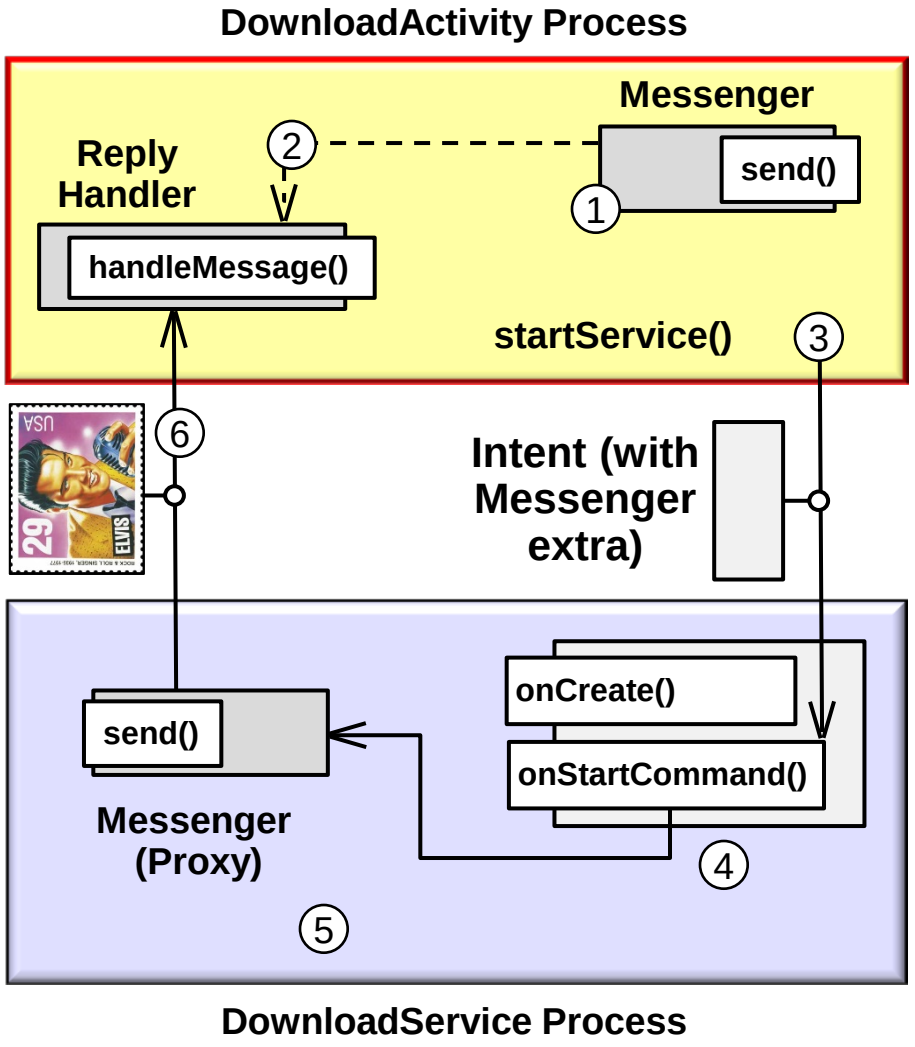
Service to Activity Communication w/Messengers

- A started Service can perform IPC with an Activity via a Messenger
- A started Service typically does three things when it receives an Intent
  - Obtains Messenger from Intent
  - Performs some processing
  - Returns the results back to the sender process
    - e.g., DownloadService sends the image path back to the DownloadActivity



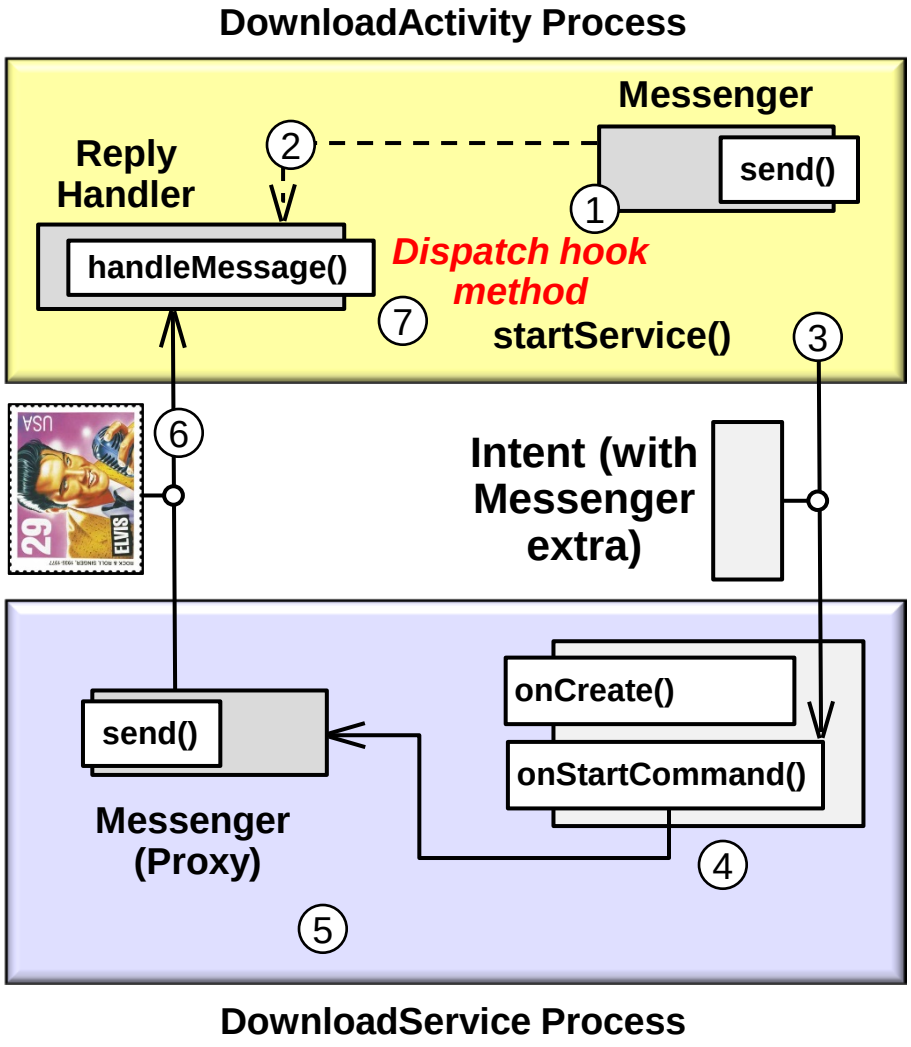
Service to Activity Communication w/Messengers

- A started Service can perform IPC with an Activity via a Messenger
- A started Service typically does three things when it receives an Intent
- The Message returned from the Started Service is dispatched via ReplyHandler.handleMessage()



Service to Activity Communication w/Messengers

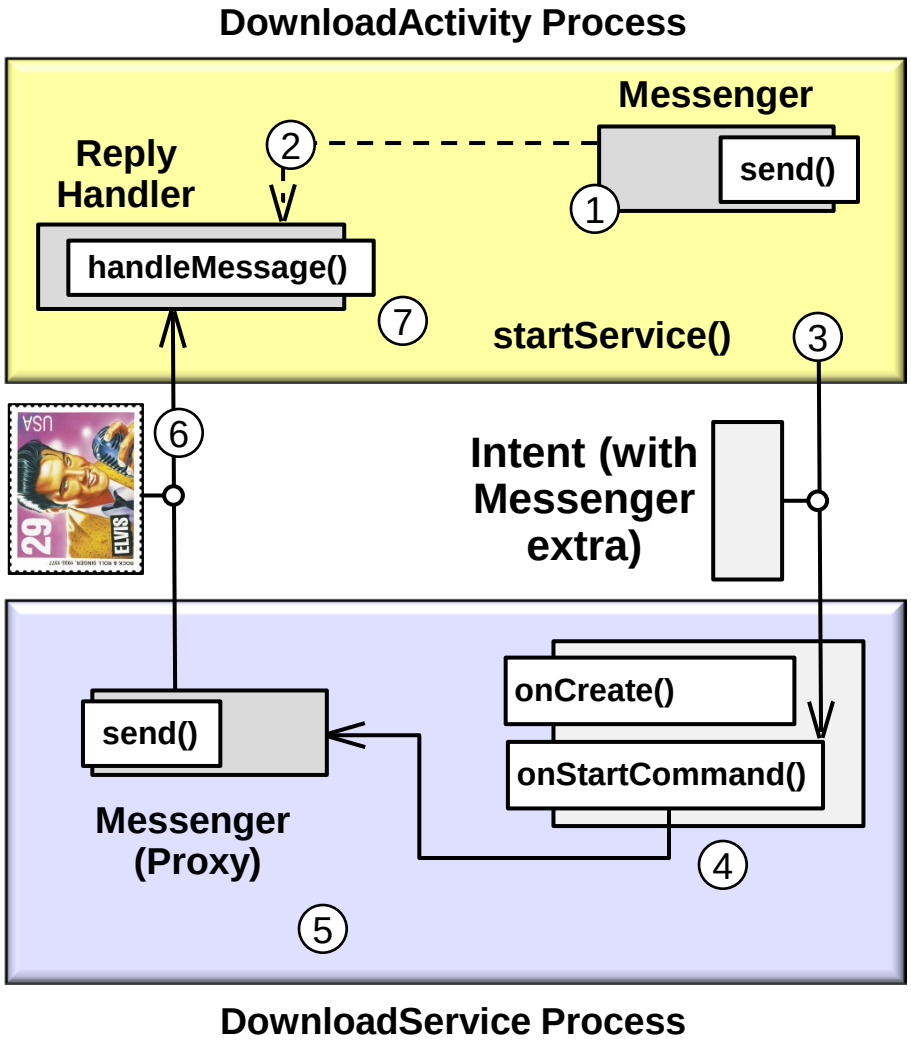
- A started Service can perform IPC with an Activity via a Messenger
- A started Service typically does three things when it receives an Intent
- The Message returned from the Started Service is dispatched via ReplyHandler.handleMessage()
  - e.g., DownloadActivity displays the image whose pathname is returned from DownloadService



handleMessage() runs in the Thread of the ReplyHandler

Service to Activity Communication w/Messengers

- A started Service can perform IPC with an Activity via a Messenger
- A started Service typically does three things when it receives an Intent
- The Message returned from the Started Service is dispatched via ReplyHandler.handleMessage()

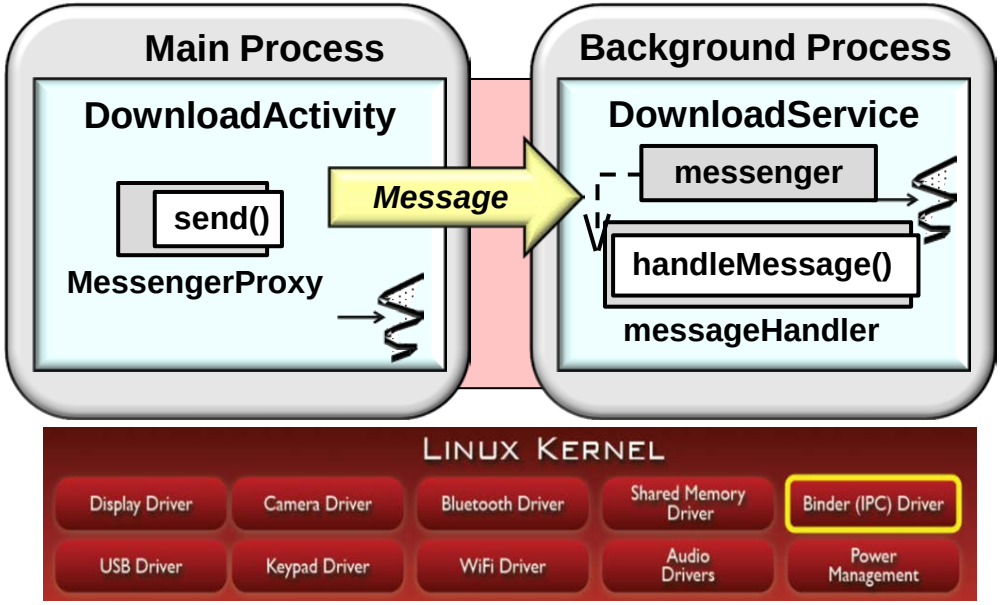


See upcoming parts on “Programming Bound Services with Messengers”

Patterns Applied with Messengers

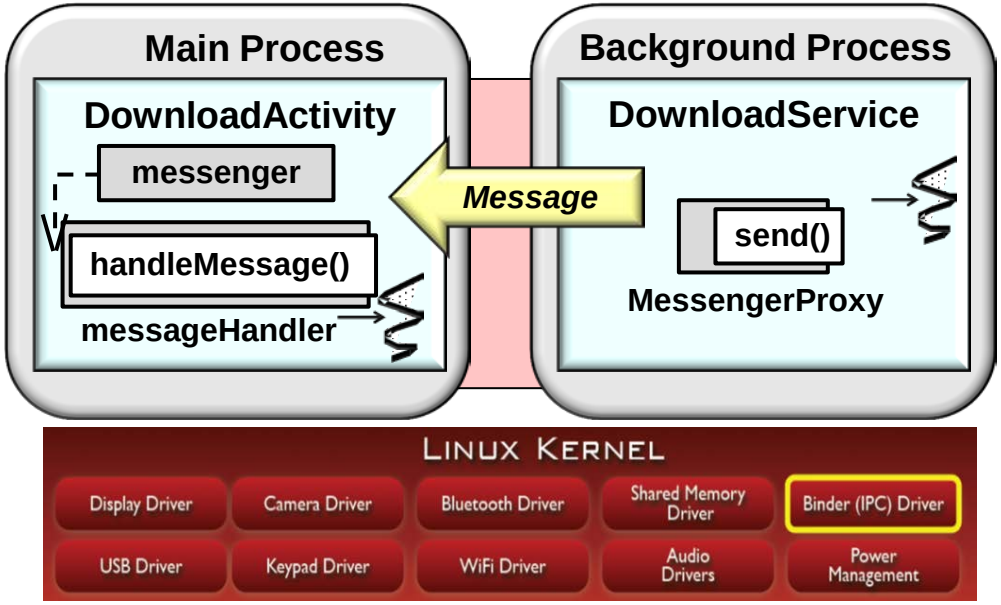
Patterns Applied with Messengers

- Messenger-based programs apply the *Active Object* pattern



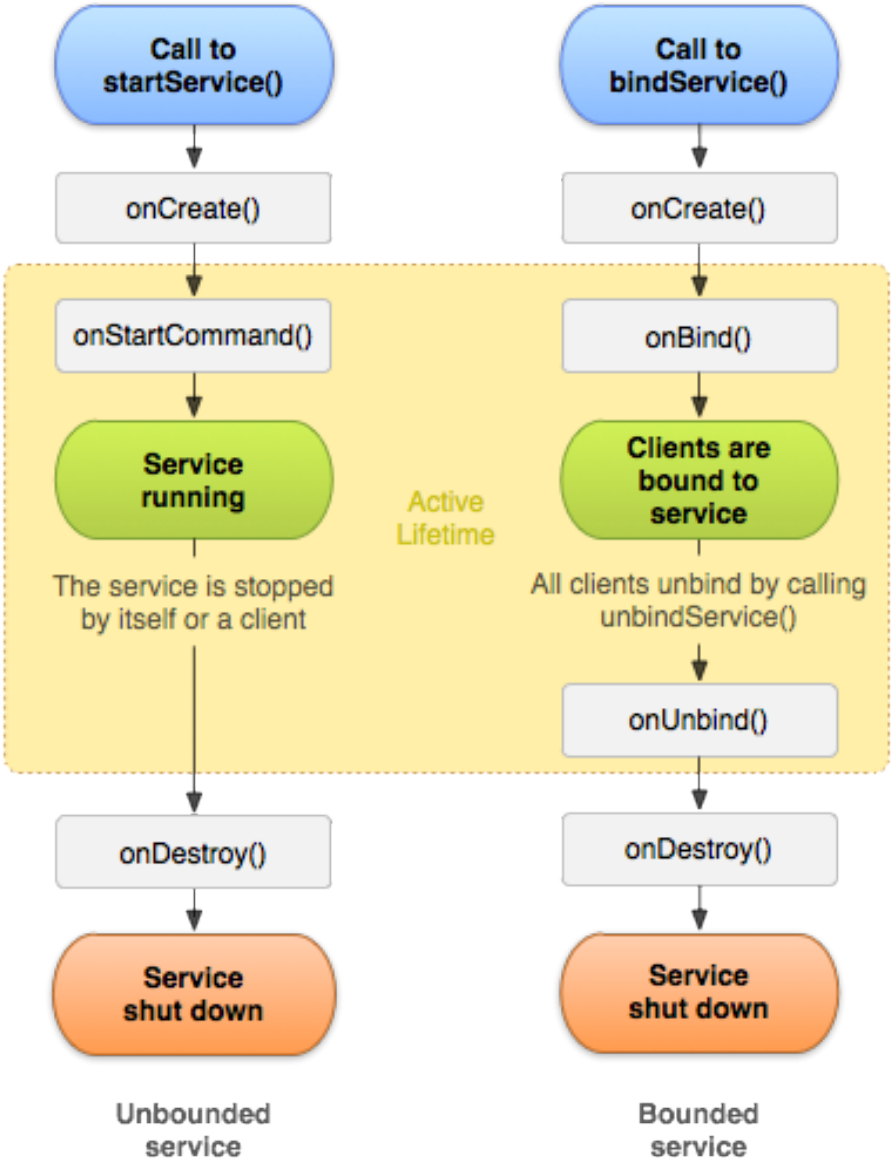
Patterns Applied with Messengers

- Messenger-based programs apply the *Active Object* pattern



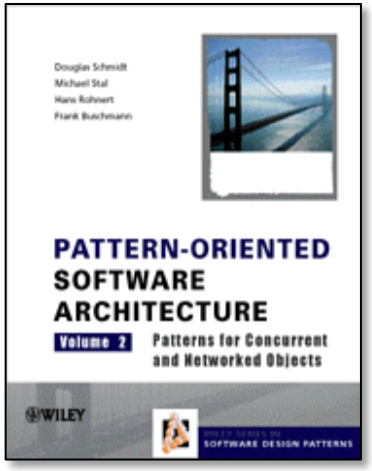
Patterns Applied with Messengers

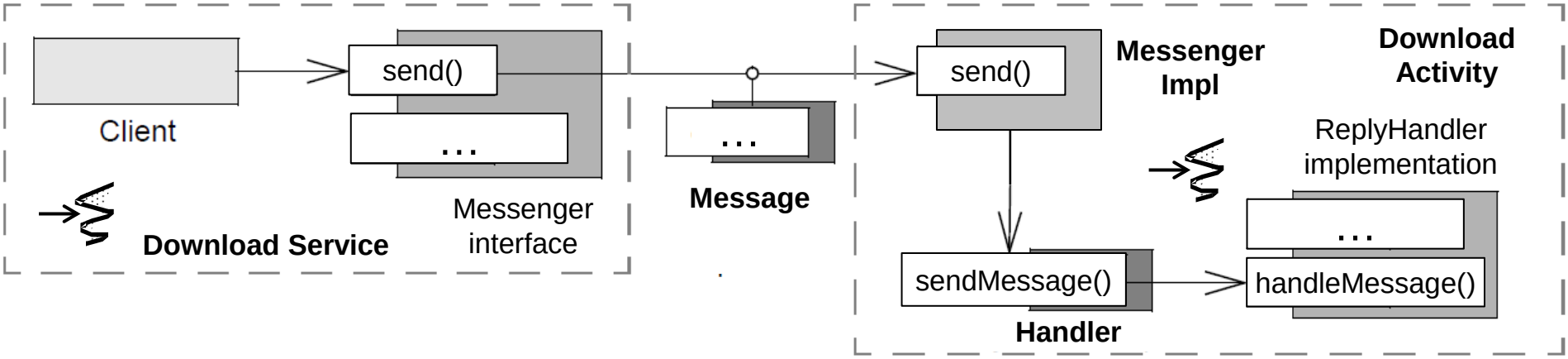
- Messenger-based programs apply the *Active Object* pattern



Patterns Applied with Messengers

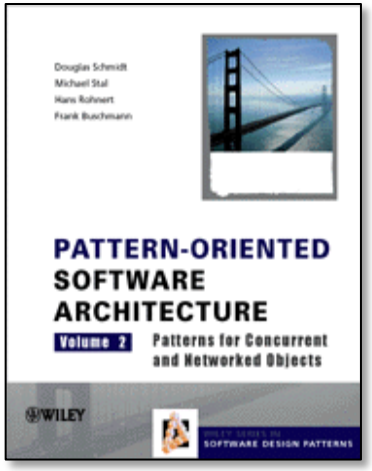
- Messenger-based programs apply the *Active Object* pattern

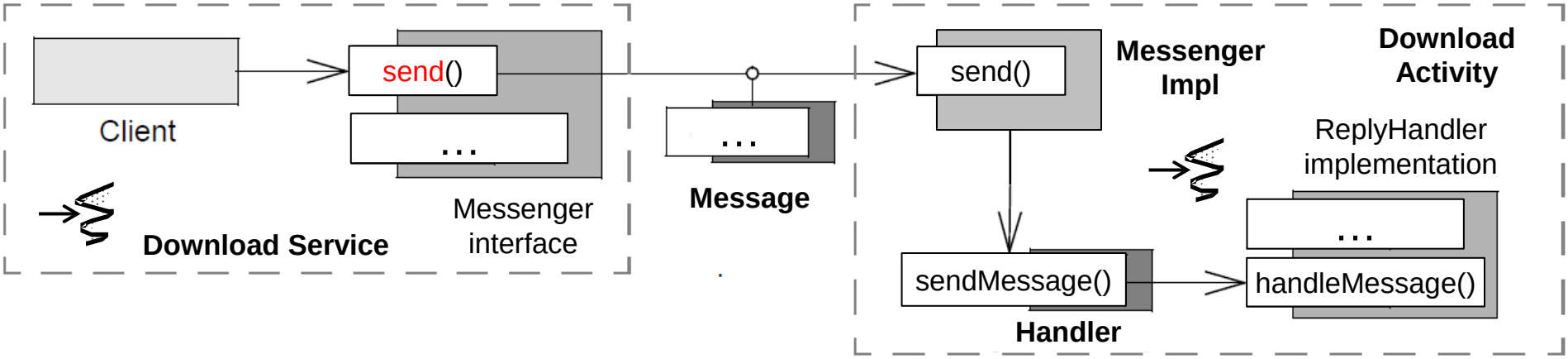




Patterns Applied with Messengers

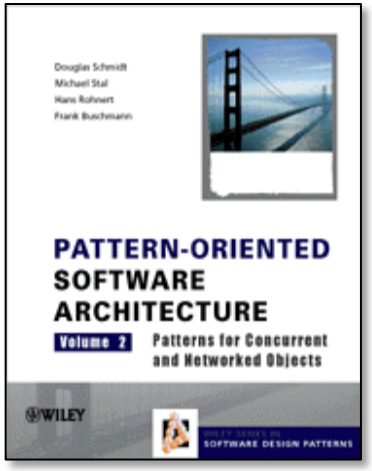
- Messenger-based programs apply the *Active Object* pattern

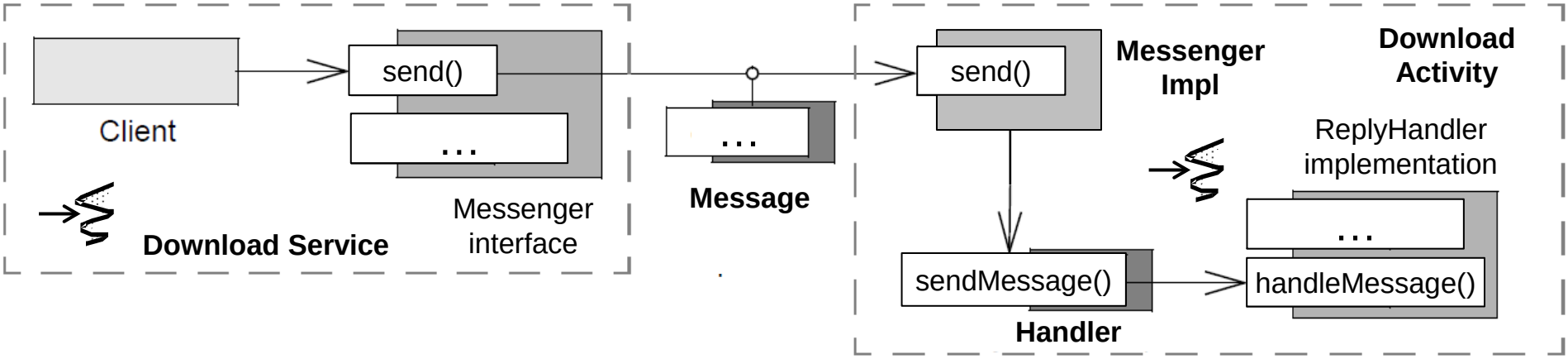




Patterns Applied with Messengers

- Messenger-based programs apply the *Active Object* pattern



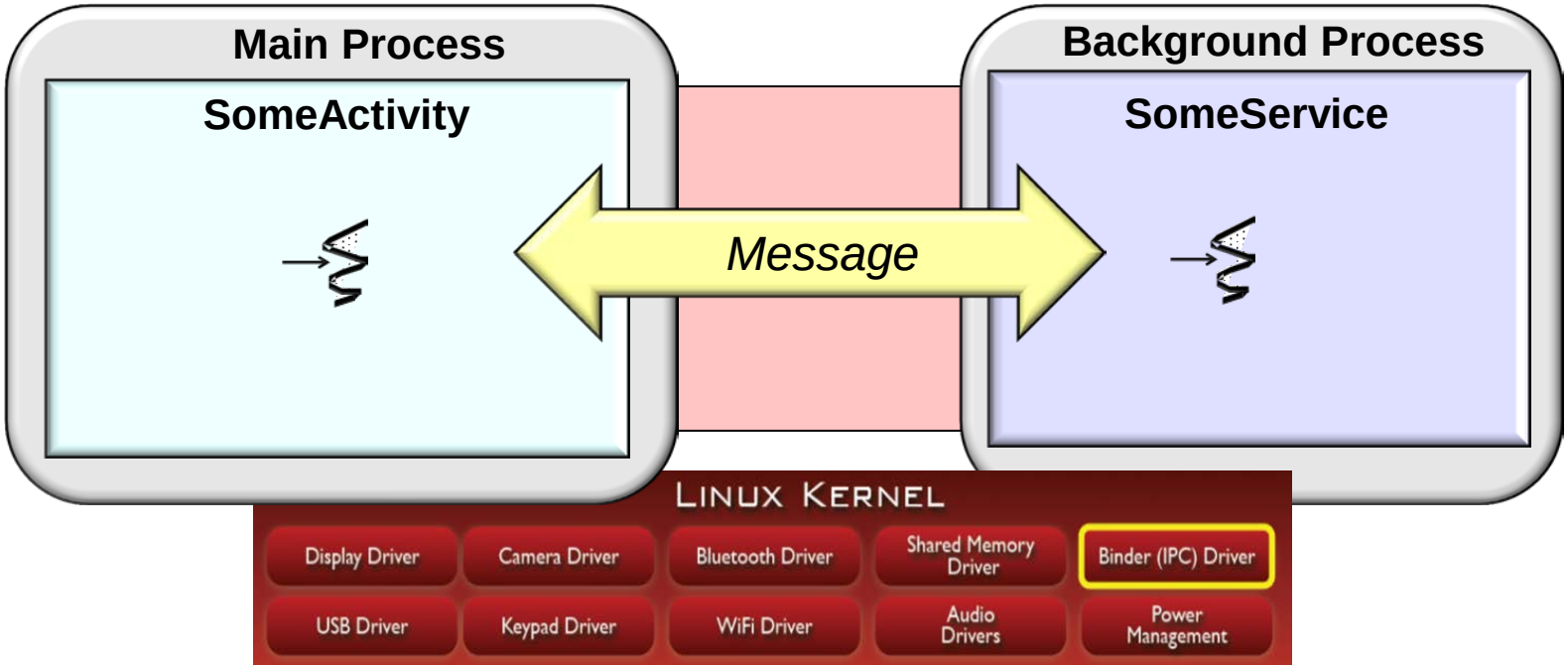


See upcoming parts on “The Active Object Pattern”

Usage Considerations for Messengers

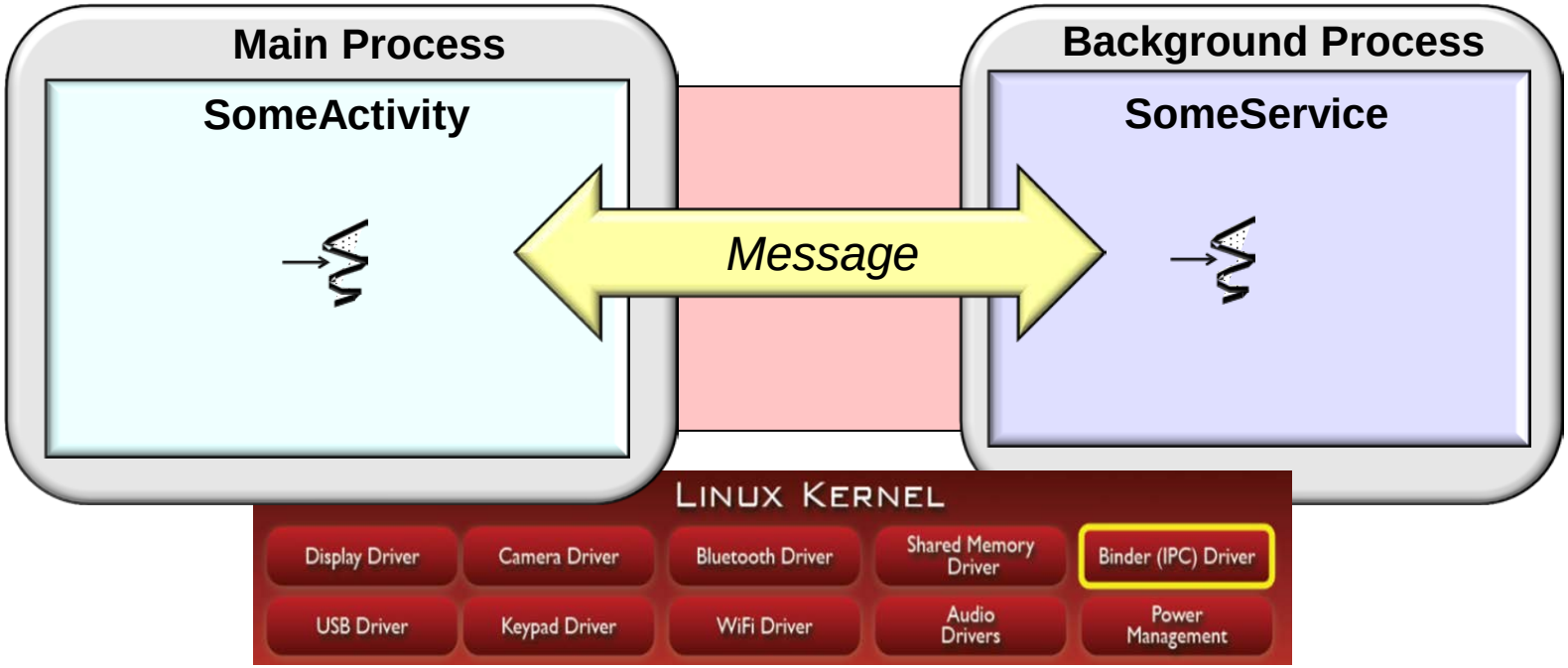
Usage Considerations for Messengers

- Messengers enable Message-based Activity & Service communication



Usage Considerations for Messengers

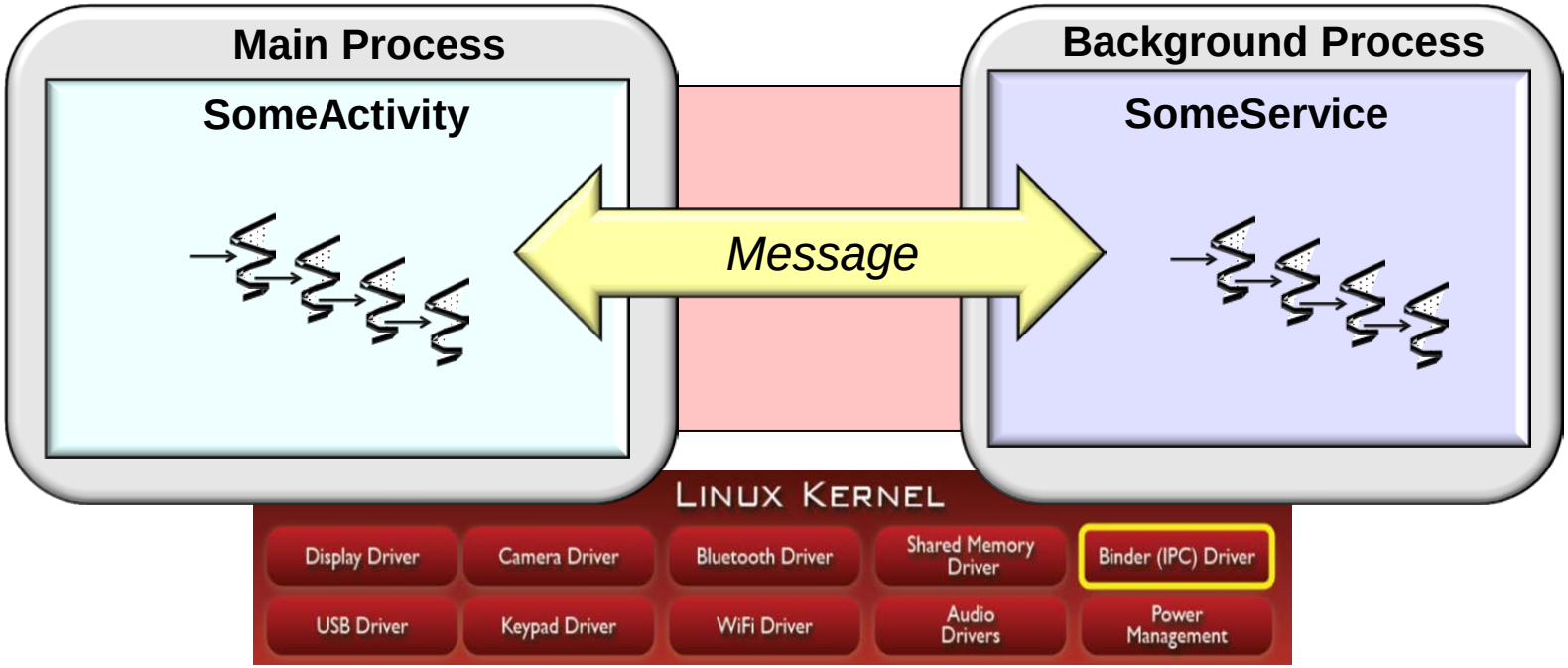
- Messengers enable Message-based Activity & Service communication
  - They receive Messages sequentially



See developer.android.com/guide/components/bound-services.html#Messenger

Usage Considerations for Messengers

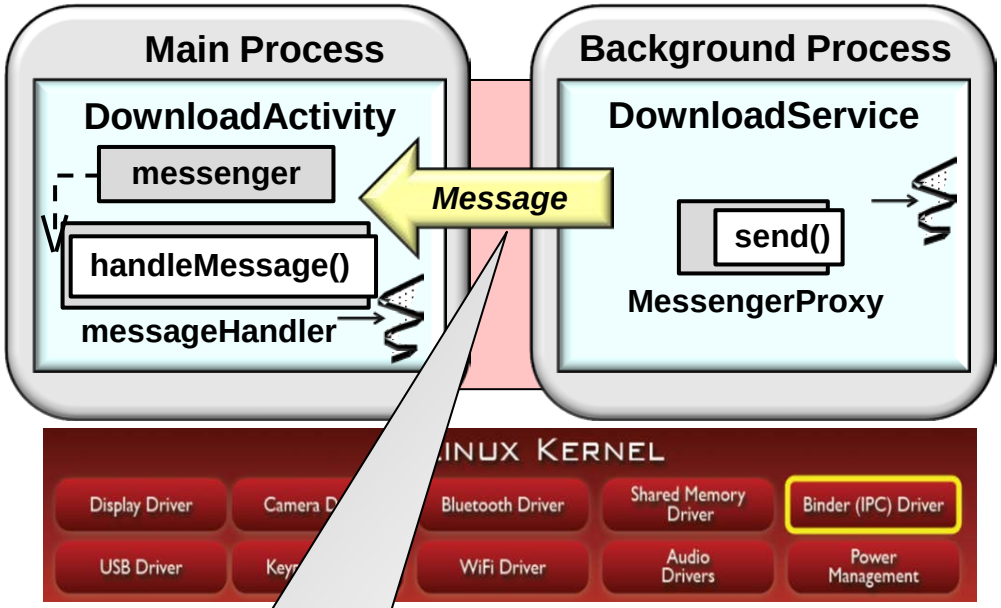
- Messengers enable Message-based Activity & Service communication
  - They receive Messages sequentially
  - It's straightforward to combine them with concurrency mechanisms



See upcoming parts on "Programming Bound Services with Messengers"

Usage Considerations for Messengers

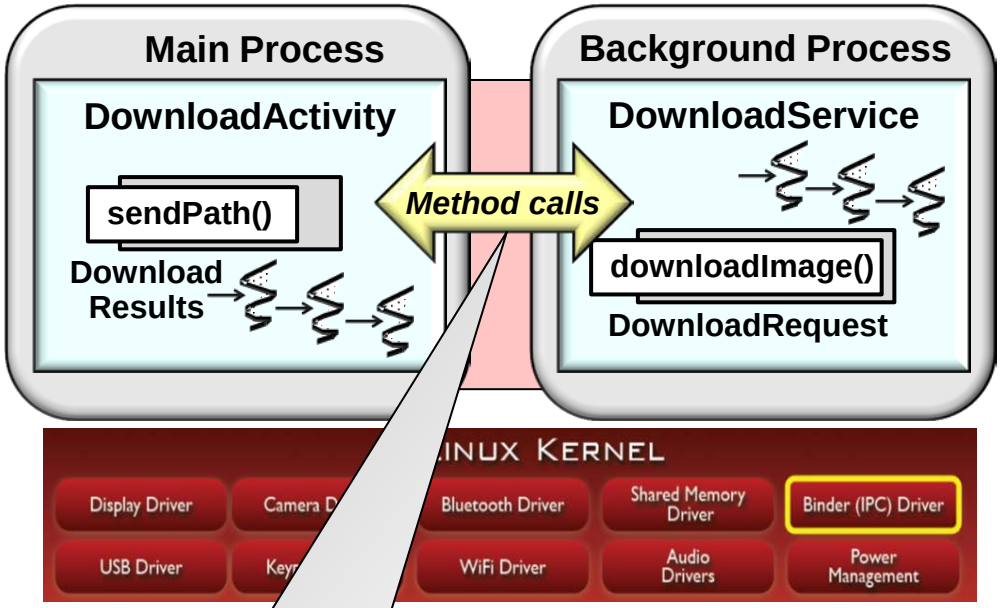
- Messengers enable Message-based Activity & Service communication
- Messengers are best suited for simple interactions & data types




```
private Message makeReplyMessage(String pathname){
  Message message = Message.obtain();
  message.arg1 = pathname == null
    ? Activity.RESULT_CANCELED : Activity.RESULT_OK;
  Bundle bundle = new Bundle();
  bundle.putString("PATHNAME", pathname);
  message.setData(bundle);
  return message;
}
```

Usage Considerations for Messengers

- Messengers enable Message-based Activity & Service communication
- Messengers are best suited for simple interactions & data types
  - AIDL may be better suited for more sophisticated interactions, complex data types, & concurrency models




```
interface DownloadResults {
  oneway void sendPath (in String filePath);
}

interface DownloadRequest {
  oneway void downloadImage (in Uri uri,
                             in DownloadResults Results);
}
```

See upcoming parts on “Overview of AIDL & the Binder Framework”