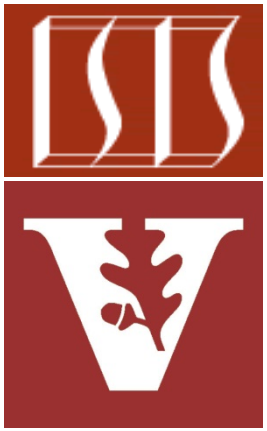
Android Services & Local IPC: Overview of the Android Interface Definition Language & Binder Framework

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



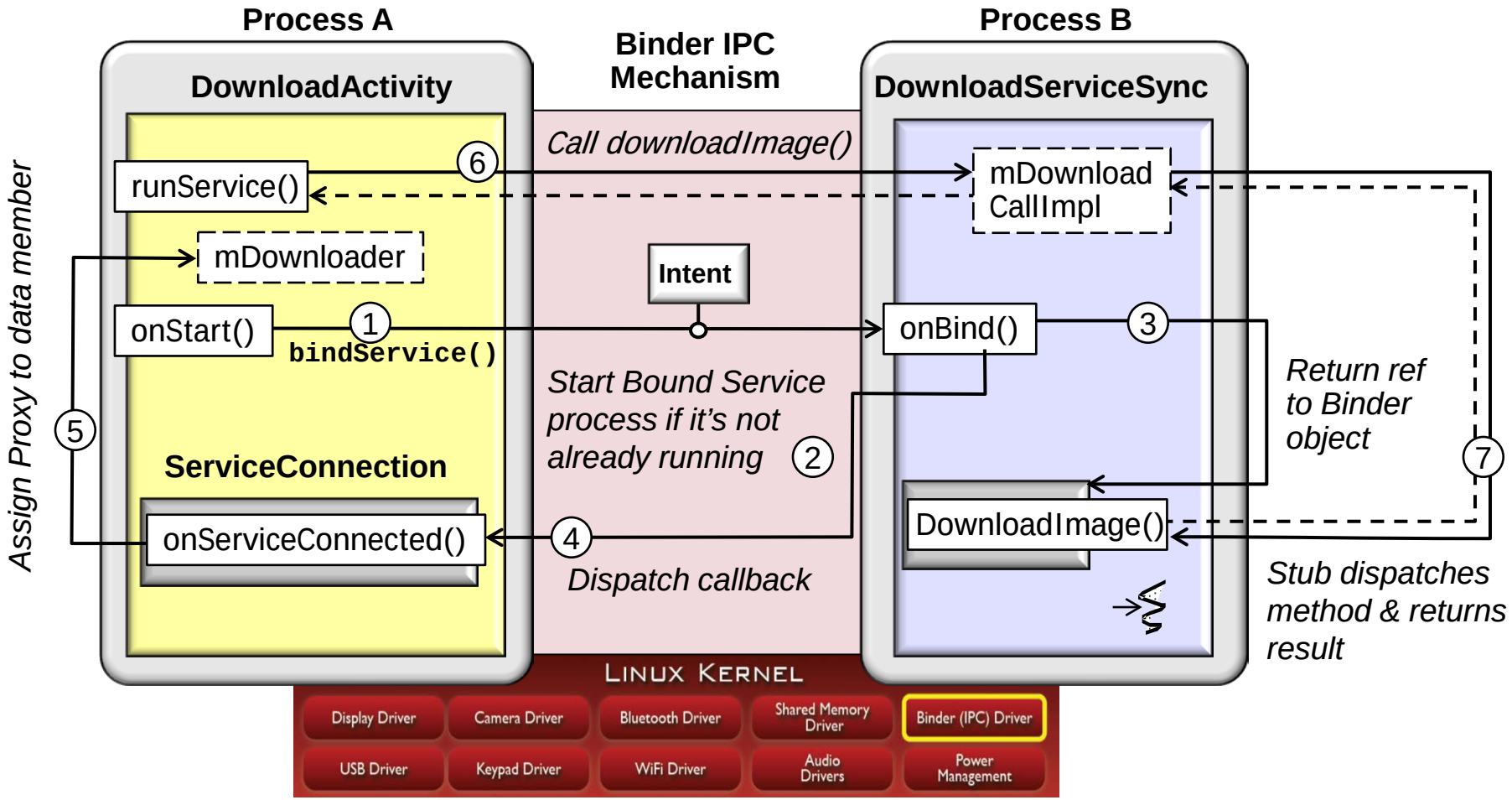
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

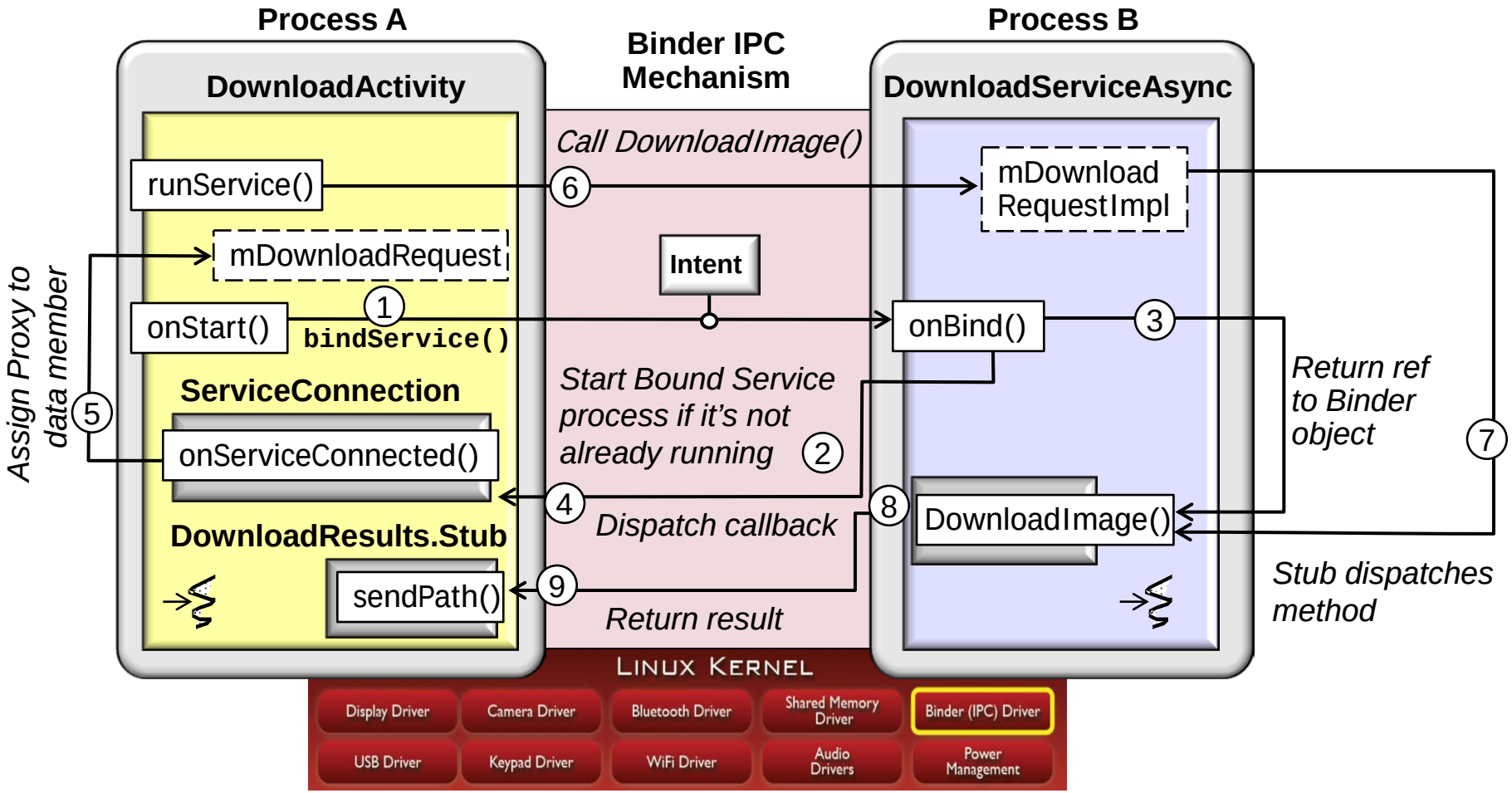
Learning Objectives in this Part of the Module

- Understand AIDL & Binder mechanisms Activities use to communicate with Bound Services synchronously



Learning Objectives in this Part of the Module

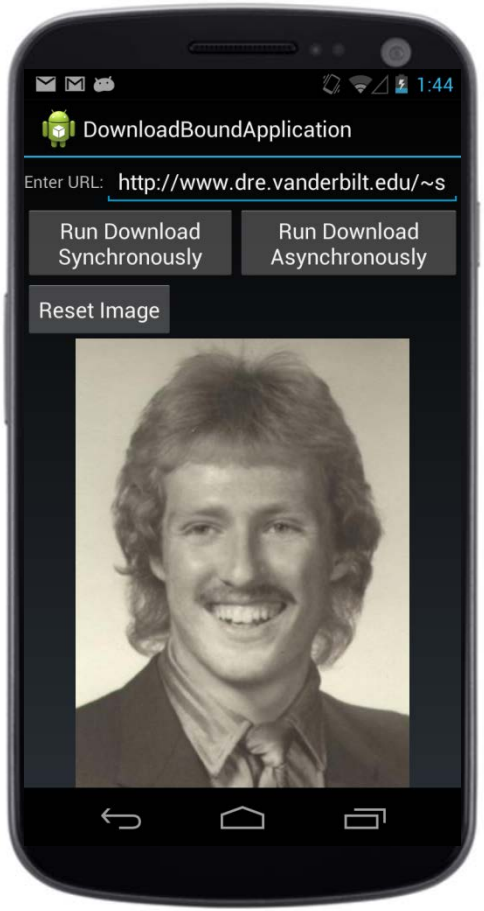
- Understand AIDL & Binder mechanisms Activities use to communicate with Bound Services synchronously & asynchronously

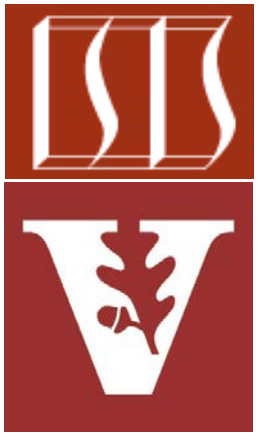


AIDL & Binder provide powerful mechanisms for typed object-oriented IPC

The Bounded Download Application

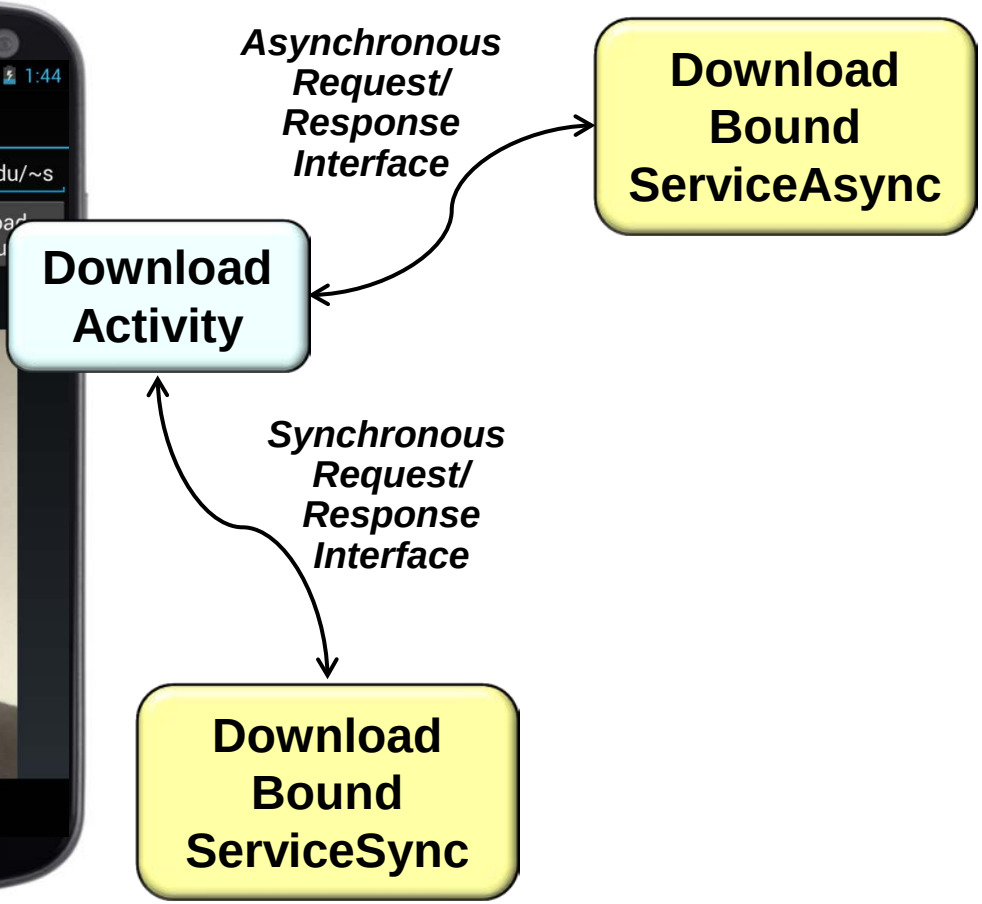
- BoundDownloadApplication uses AIDL & Binder mechanisms to interact with Bound Services to download/display images synchronously/asynchronously





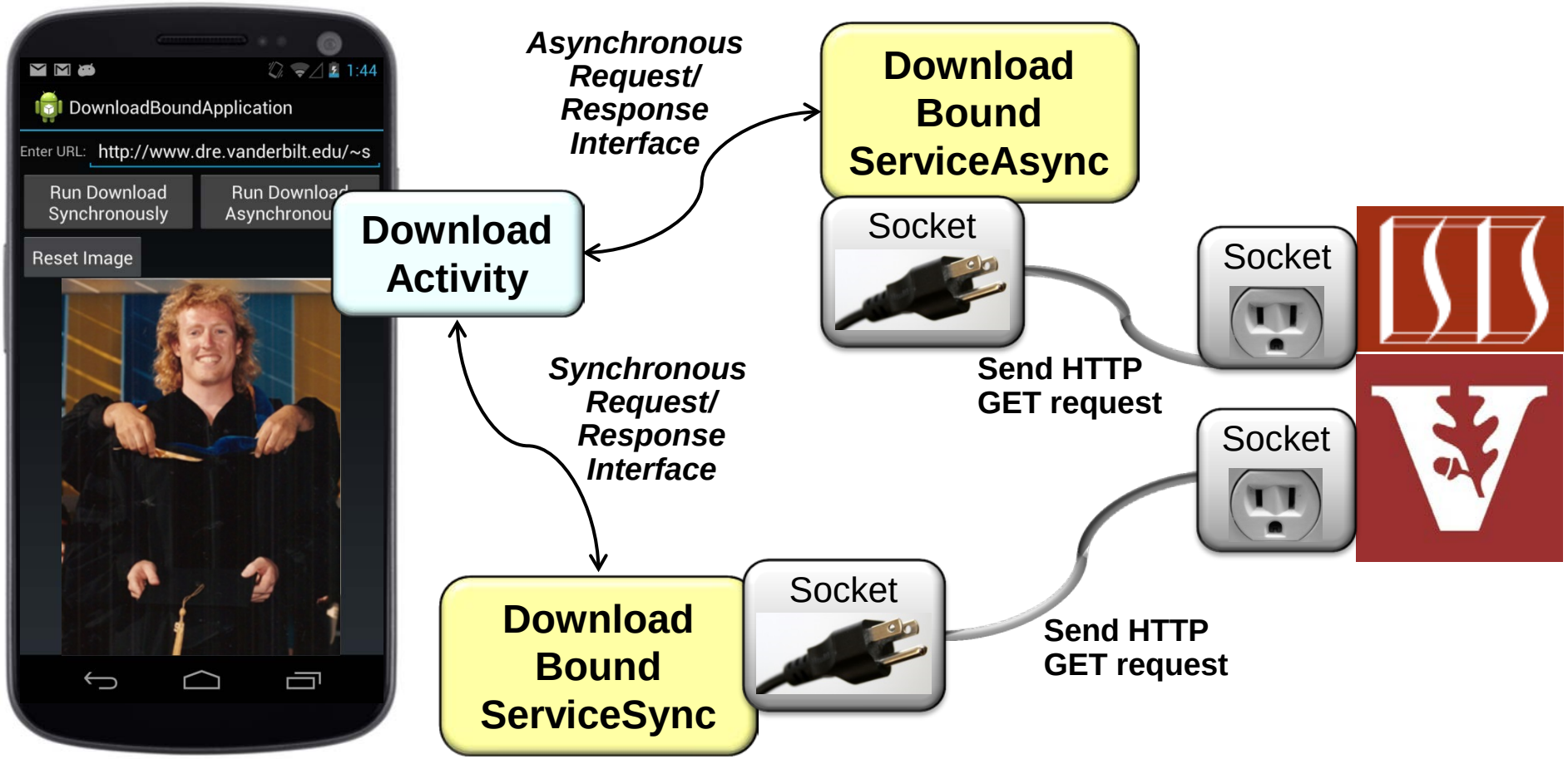
The Bounded Download Application

- BoundDownloadApplication uses AIDL & Binder mechanisms to interact with Bound Services to download/display images synchronously/asynchronously



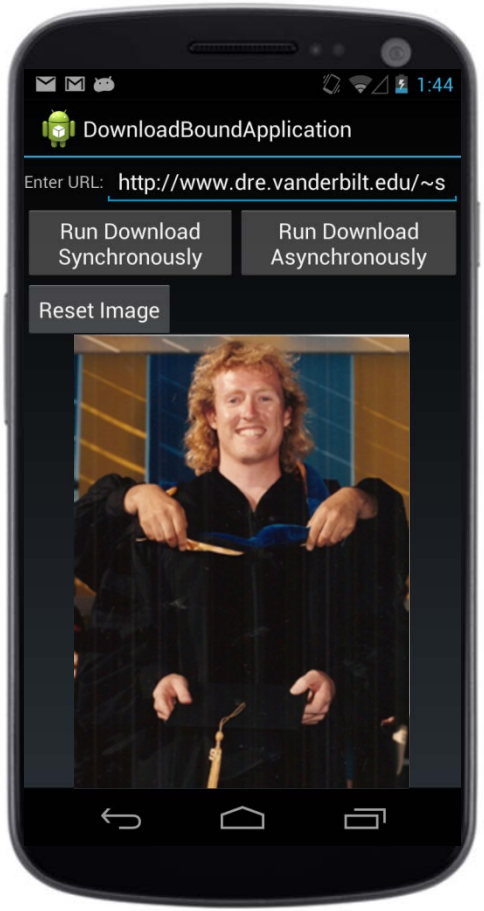
The Bounded Download Application

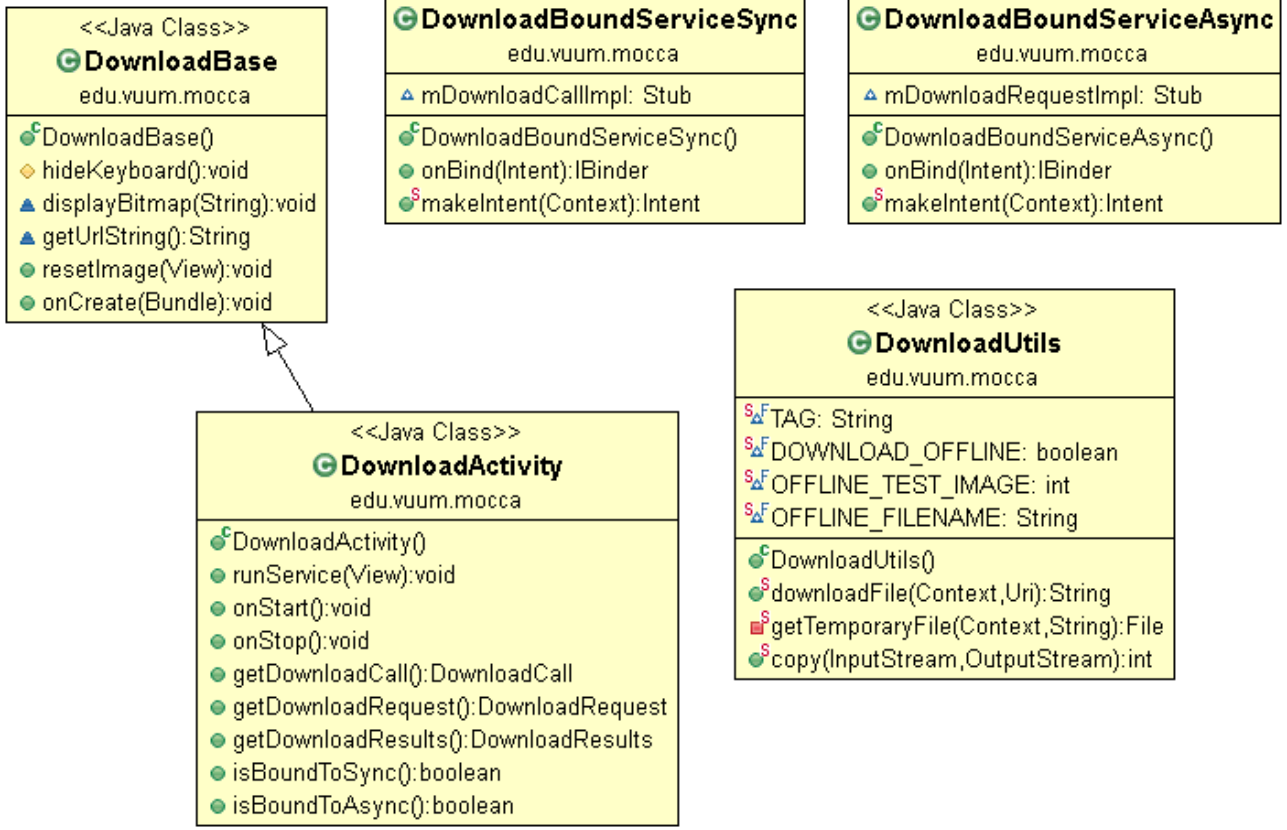
- BoundDownloadApplication uses AIDL & Binder mechanisms to interact with Bound Services to download/display images synchronously/asynchronously



The Bounded Download Application

- BoundDownloadApplication uses AIDL & Binder mechanisms to interact with Bound Services to download/display images synchronously/asynchronously





See github.com/douglascraigschmidt/POSA-14/tree/master/assignments/week-8-assignment-7

Supplemental Material on AIDL & Binder

- The AIDL & Binder mechanisms are powerful, but programmers must understand a number of constructs & steps to use them effectively

See videos on “AIDL Syntax & Supported Data Types” & “Implementing AIDL Interfaces”

Motivating the AIDL & Binder (Part 1)

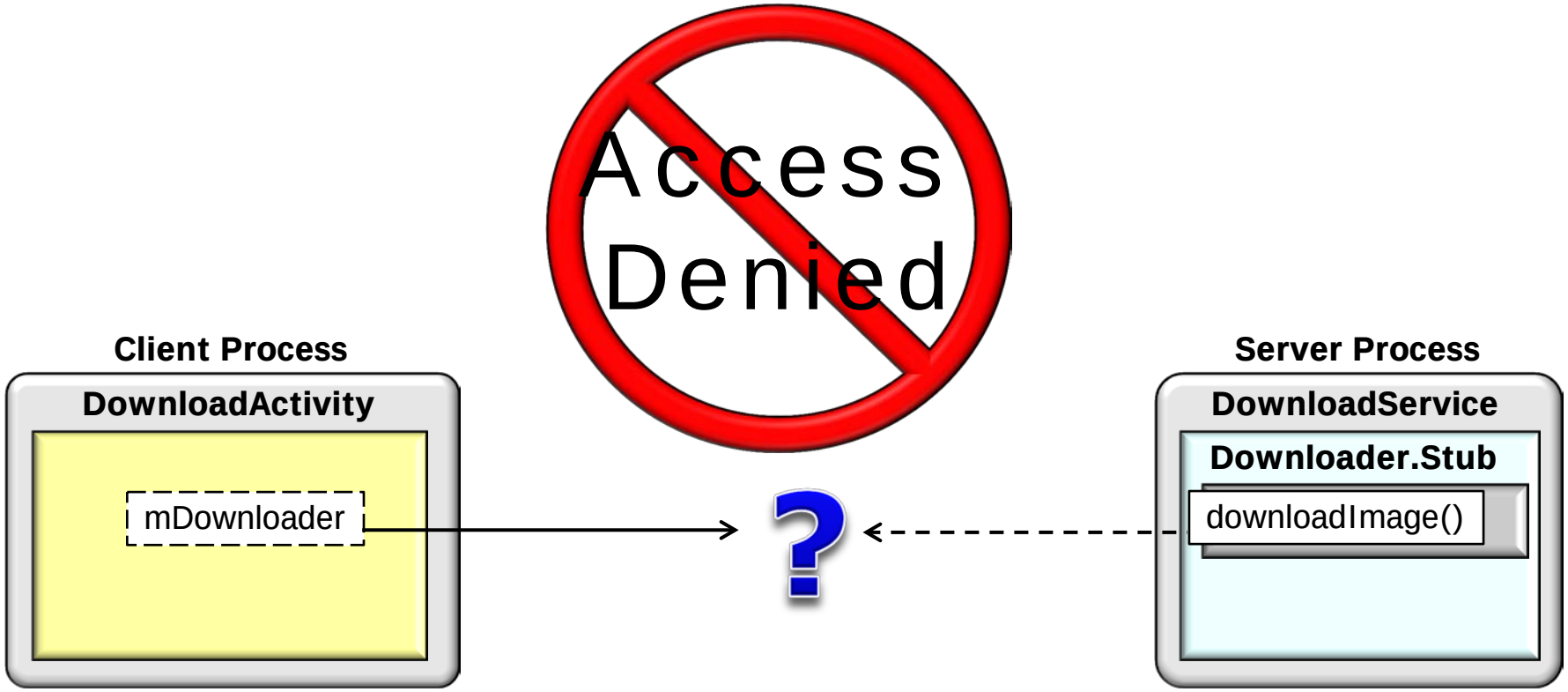
Motivating the AIDL & Binder

- Services may need to run in different processes than their clients

See previous part on “Activity & Service Communication”

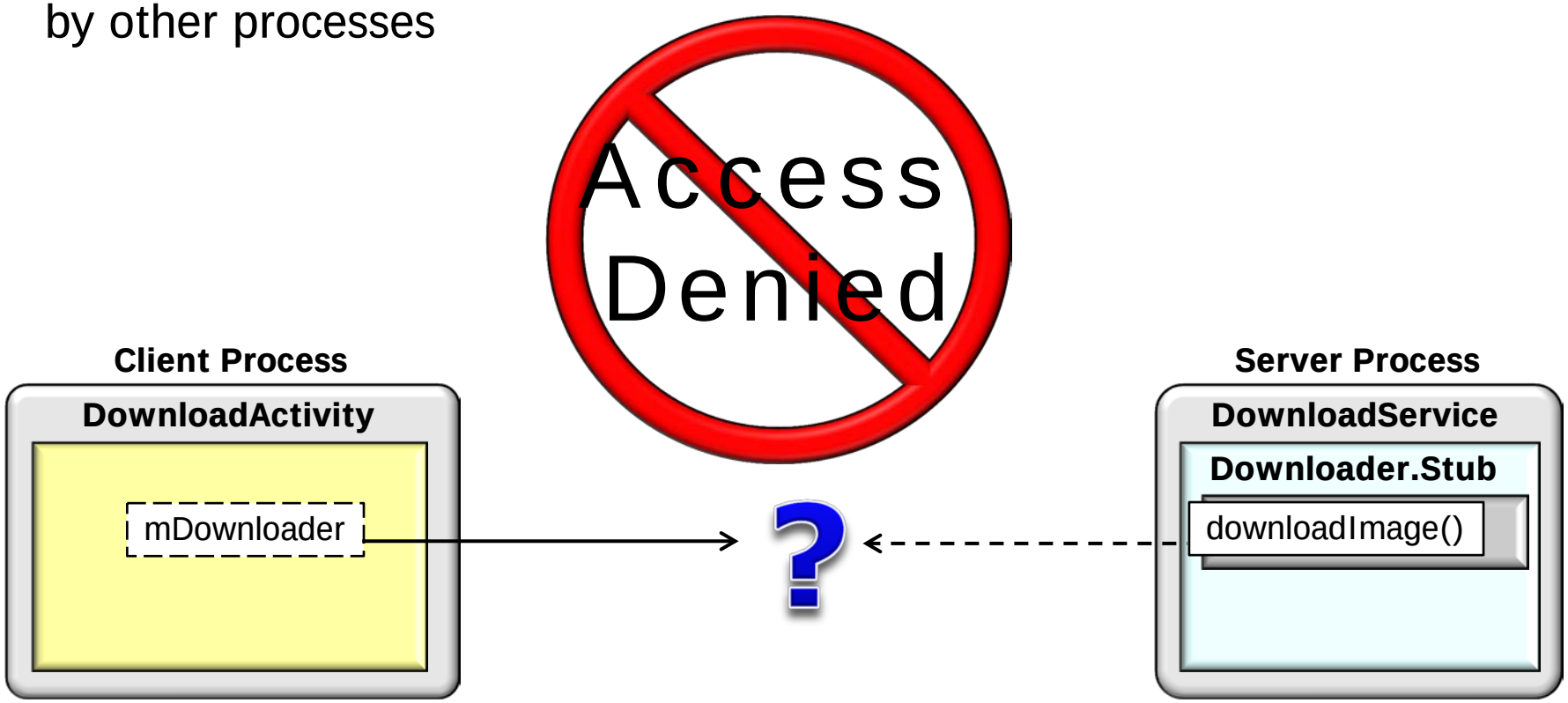
Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
  - Objects in one process on Android cannot normally access objects in the address space of another process



Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
  - Objects in one process on Android cannot normally access objects in the address space of another process
  - Processes are intentionally designed to protect their contents from access by other processes



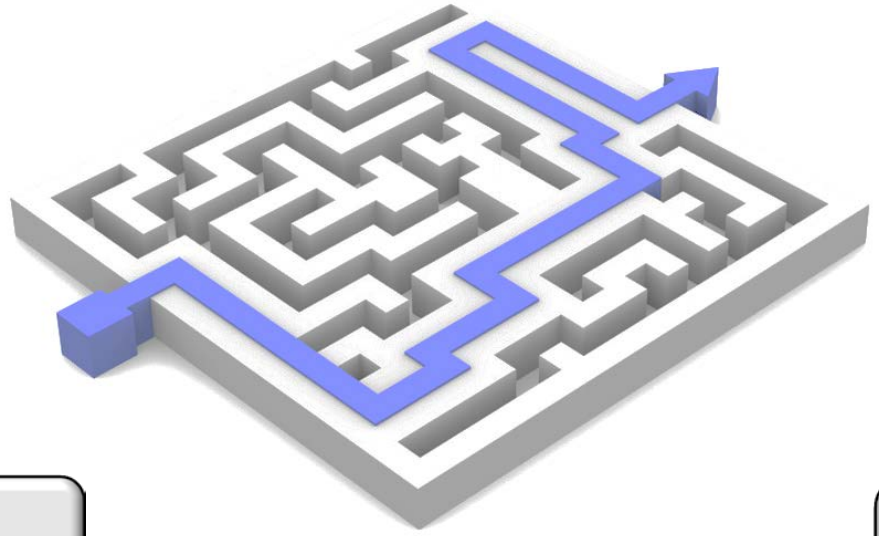
Motivating the AIDL & Binder

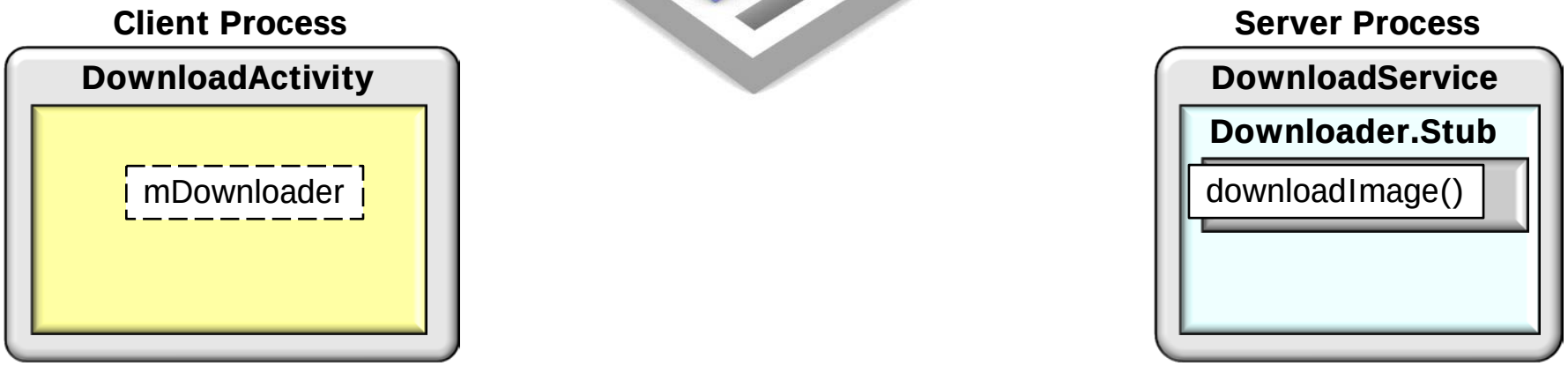
- Services may need to run in different processes than their clients
  - Objects in one process on Android cannot normally access objects in the address space of another process
  - Processes are intentionally designed to protect their contents from access by other processes

en.wikipedia.org/wiki/Process_(computing) has more info

Motivating the AIDL & Binder

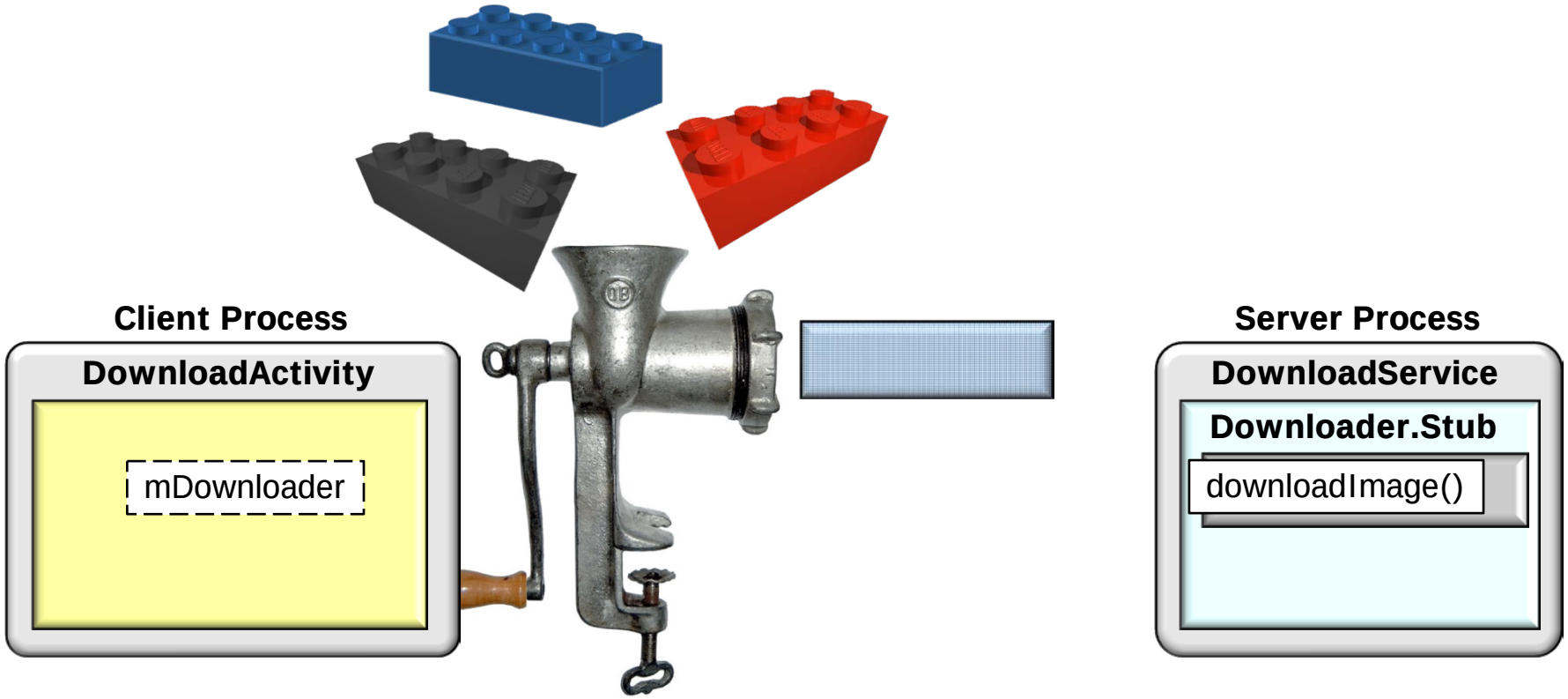
- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps





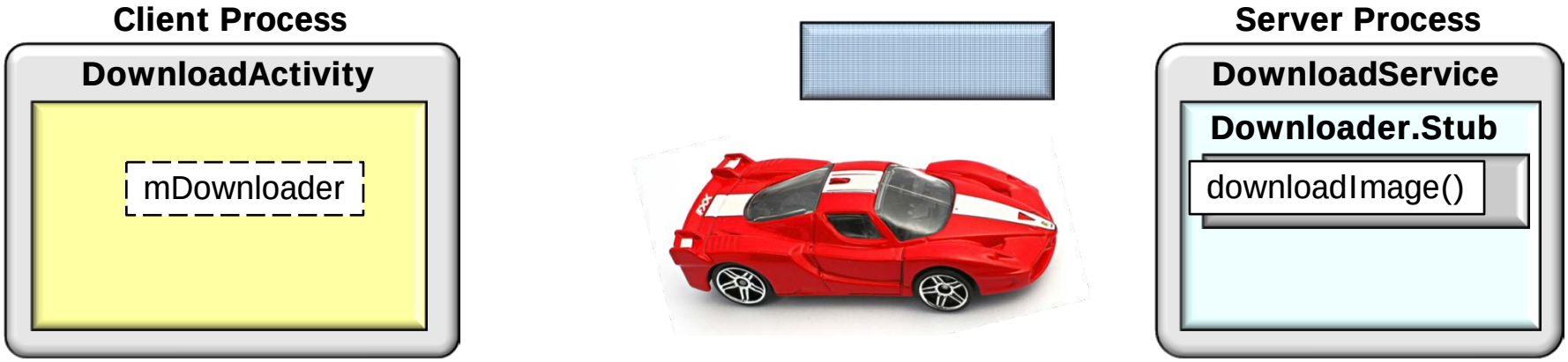
Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
  - Decompose & transform objects into bytestreams that Linux understands



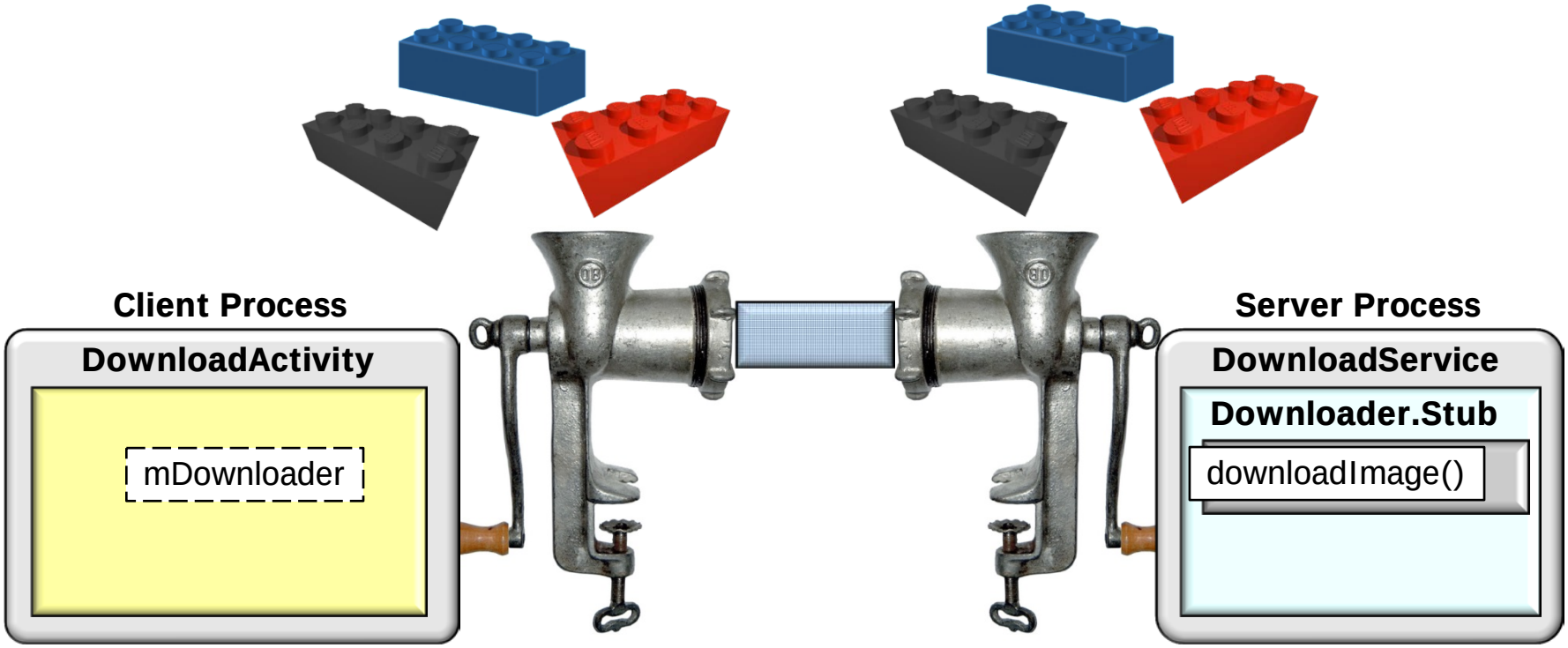
Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
  - Decompose & transform objects into bytestreams that Linux understands
  - Exchange these byte streams via the Linux IPC mechanisms



Motivating the AIDL & Binder

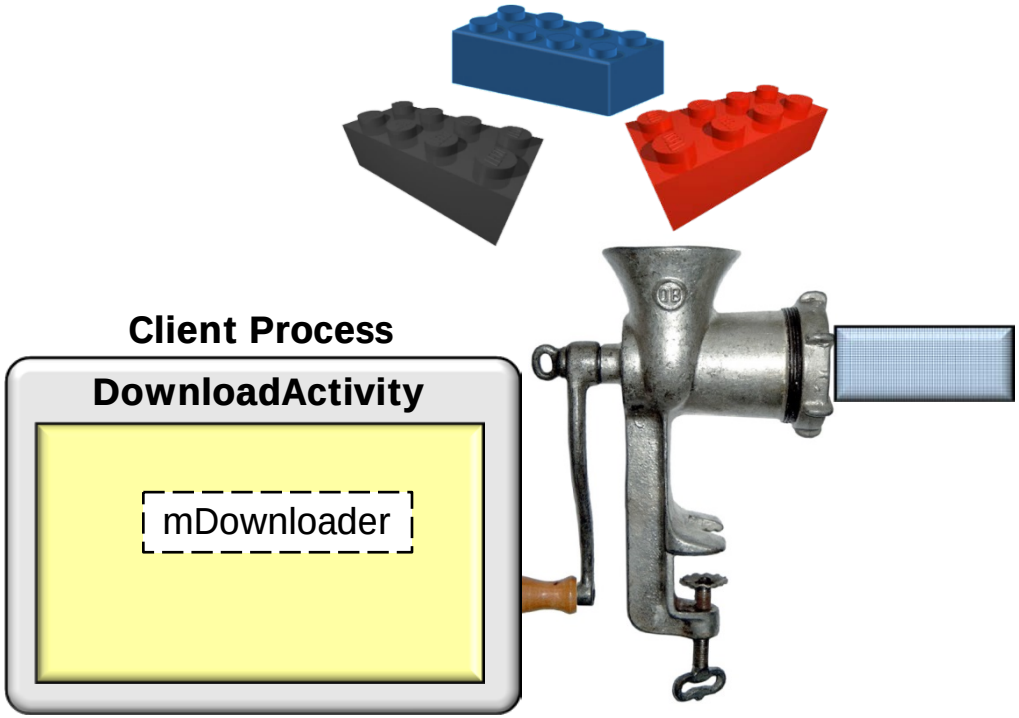
- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms



en.wikipedia.org/wiki/Marshalling_(computer_science) has more info

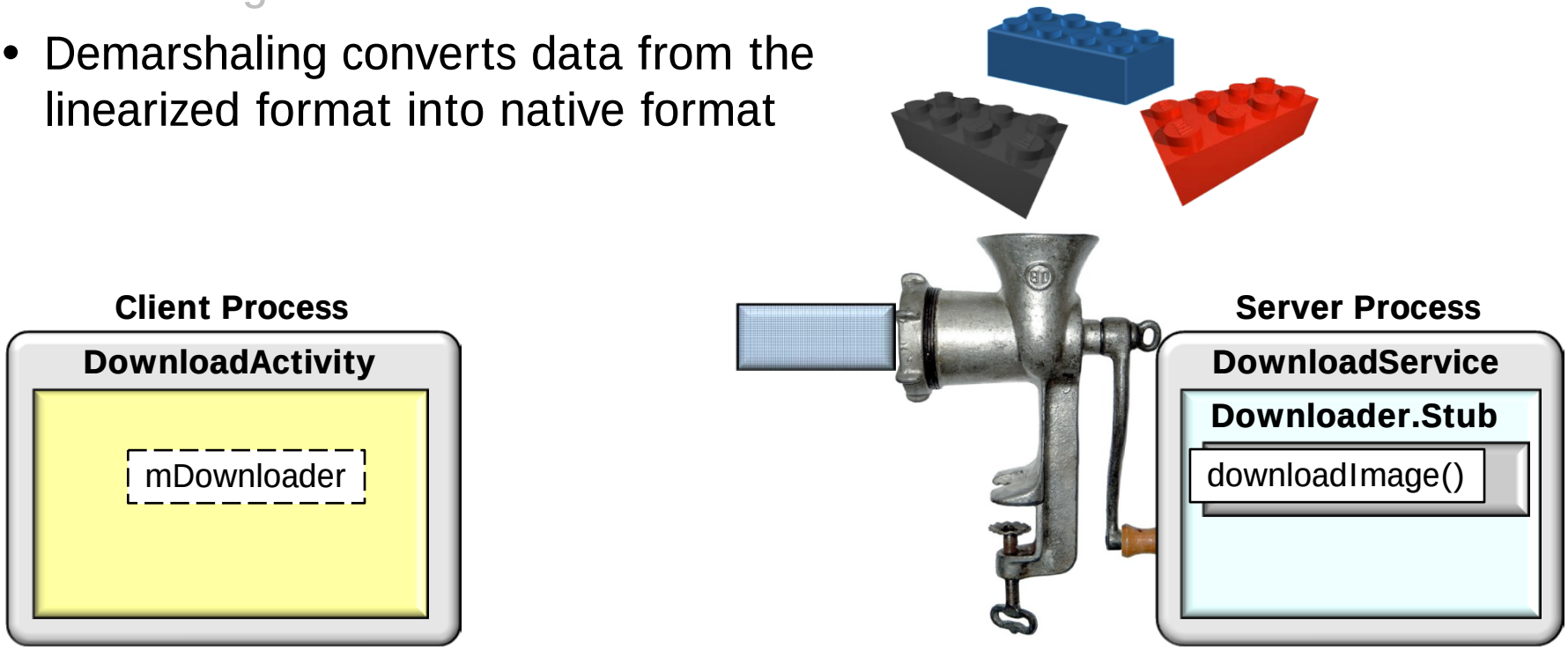
Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
  - Marshaling converts data from native format into a linearized format



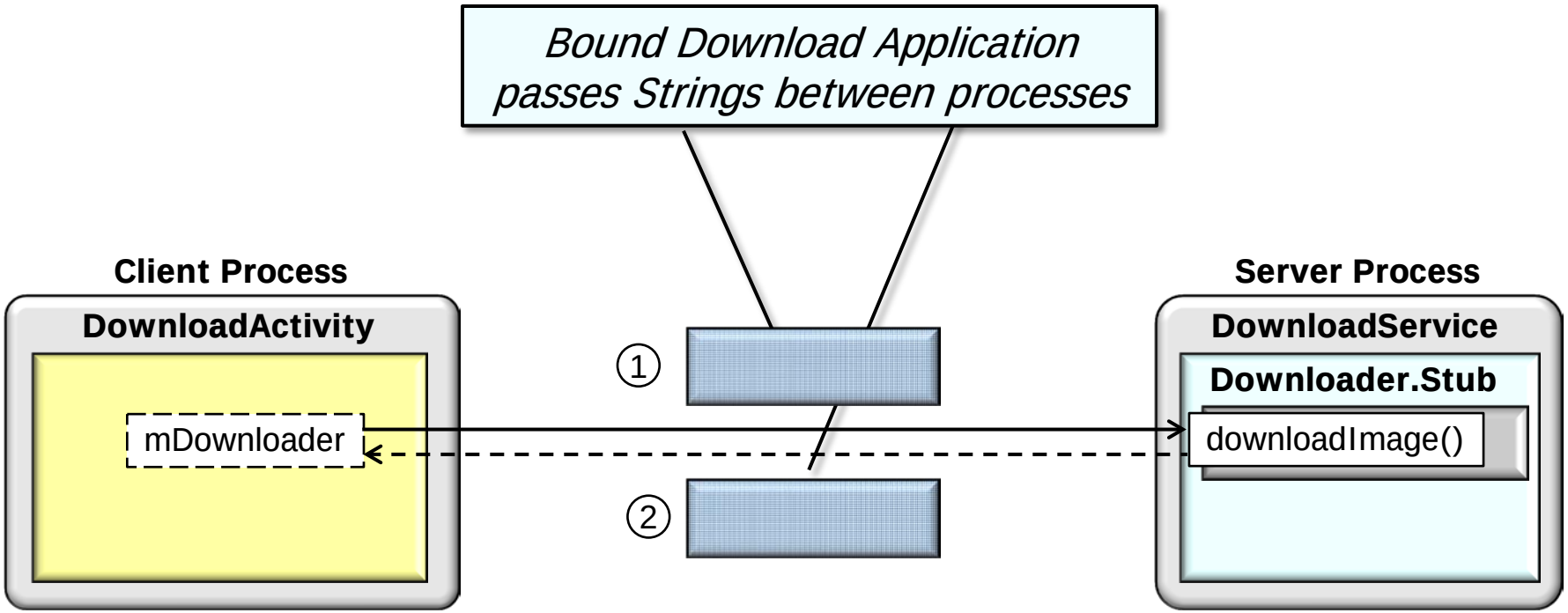
Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
  - Marshaling converts data from native format into a linearized format
  - Demarshaling converts data from the linearized format into native format



Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms



AIDL & Binder support marshaling & demarshaling of a wide range of data types

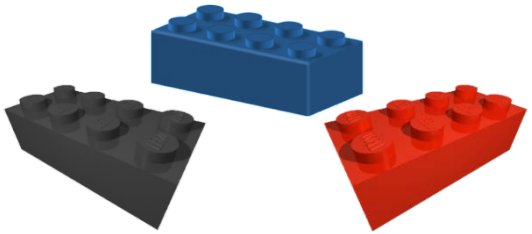
Motivating the AIDL & Binder (Part 2)

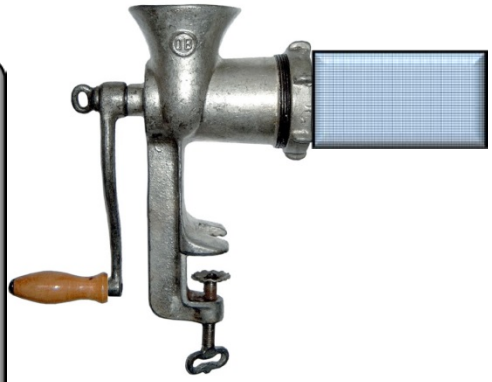
Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone

Motivating the AIDL & Binder

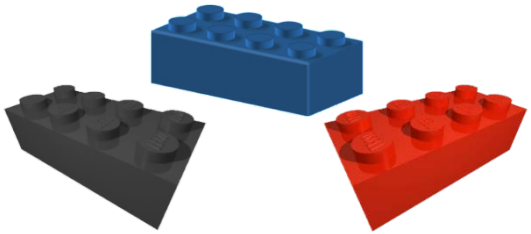
- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone

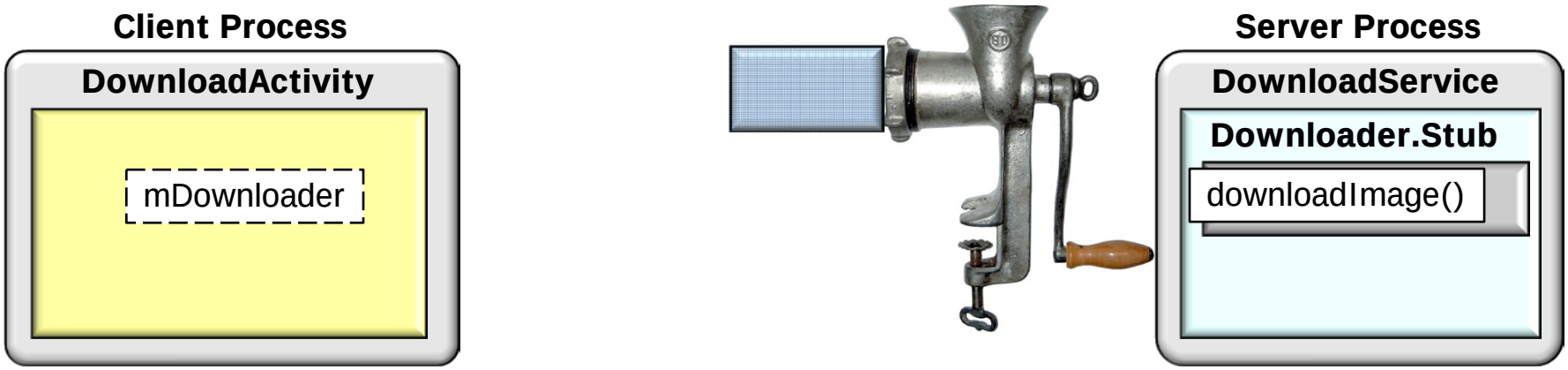




Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone





Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone, so Android automates it with the AIDL & an associated compiler

Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone, so Android automates it with the AIDL & an associated compiler

Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone, so Android automates it with the AIDL & an associated compiler

developer.android.com/guide/components/aidl.html has AIDL overview

Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone, so Android automates it with the AIDL & an associated compiler
  - AIDL is similar to Java interfaces

class.coursera.org/android-001/lecture/85 has more info on AIDL

Motivating the AIDL & Binder

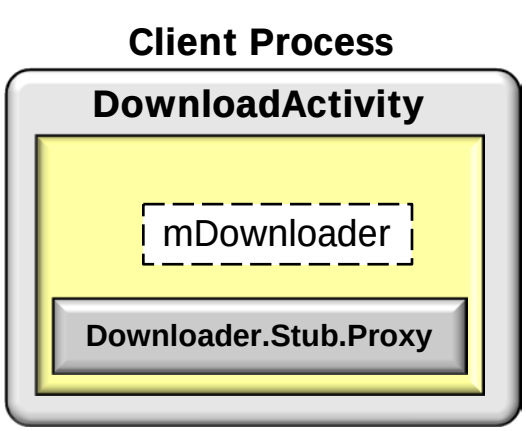
- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone, so Android automates it with the AIDL & an associated compiler
  - AIDL is similar to Java interfaces

Motivating the AIDL & Binder

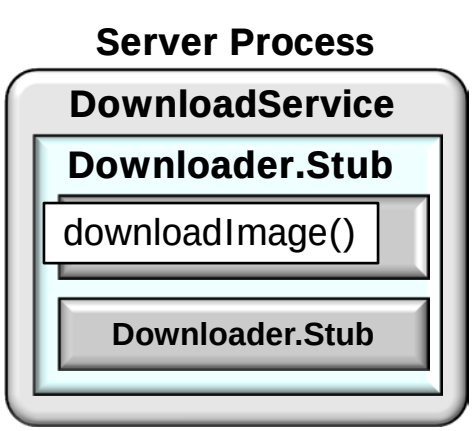
- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone, so Android automates it with the AIDL & an associated compiler
  - AIDL is similar to Java interfaces

Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone, so Android automates it with the AIDL & an associated compiler
  - AIDL is similar to Java interfaces

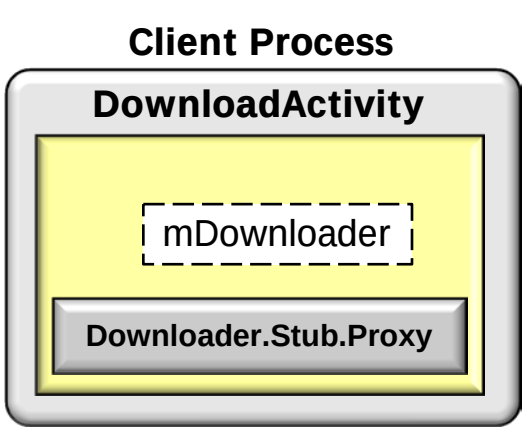



```
interface Downloader {
  String downloadImage
    (String url);
}
```

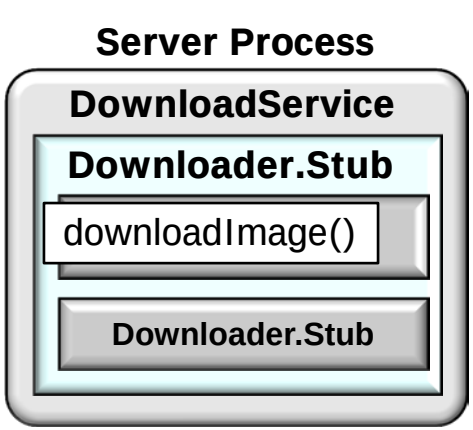


Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone, so Android automates it with the AIDL & an associated compiler
  - AIDL is similar to Java interfaces

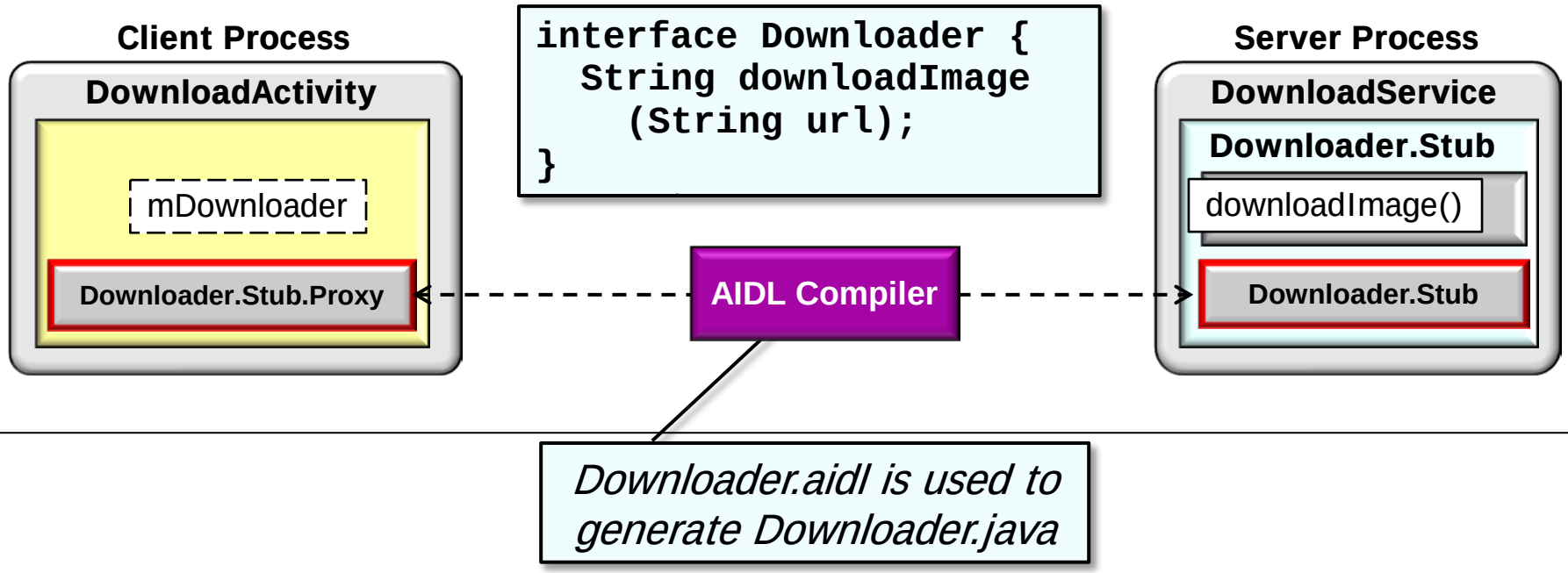



```
interface Downloader {
  String downloadImage
    (String url);
}
```



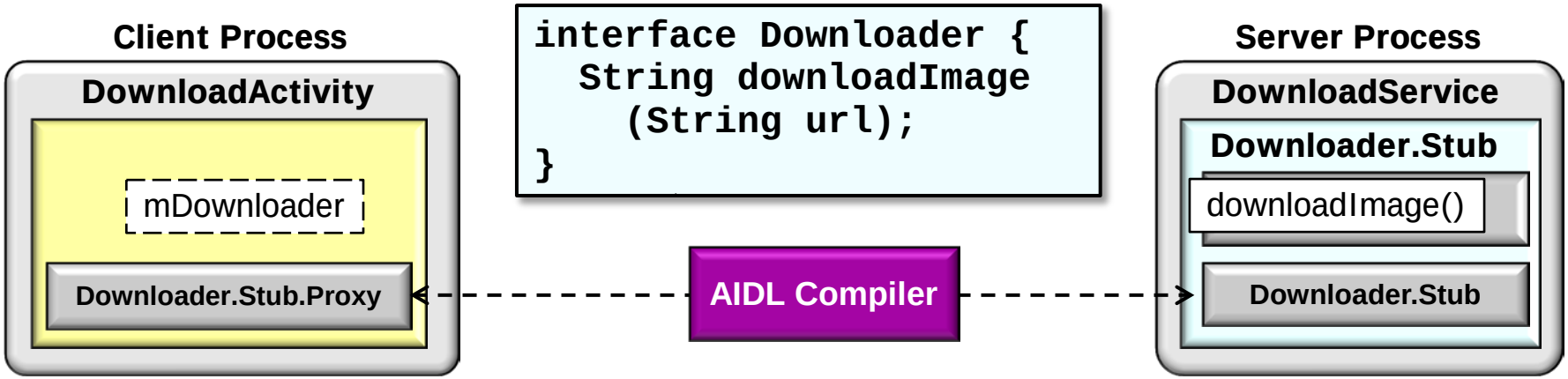
Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone, so Android automates it with the AIDL & an associated compiler
  - AIDL is similar to Java interfaces
  - Compilation is handled automatically by Eclipse



Motivating the AIDL & Binder

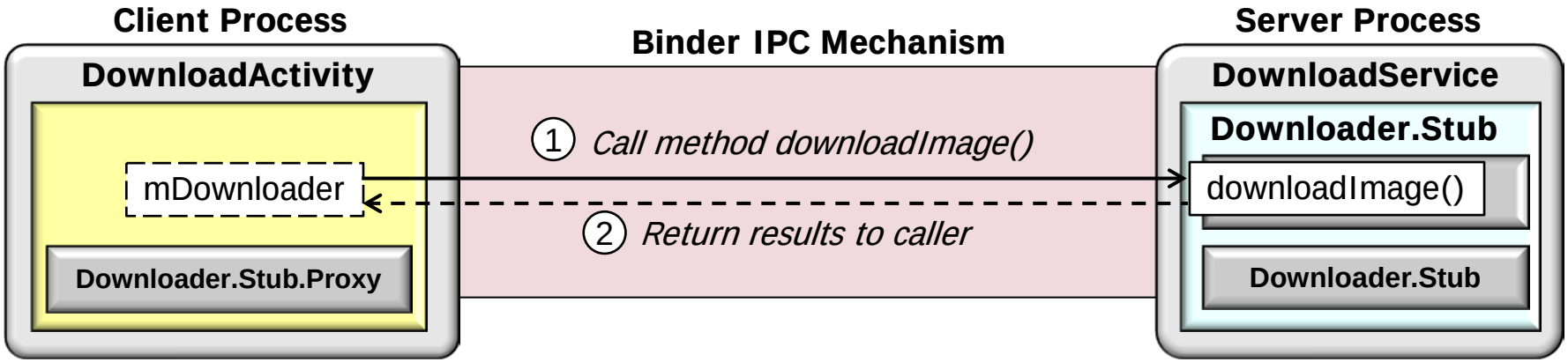
- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone, so Android automates it with the AIDL & an associated compiler
  - AIDL is similar to Java interfaces
  - Compilation is handled automatically by Eclipse



See developer.android.com/guide/components/aidl.html#Defining

Motivating the AIDL & Binder

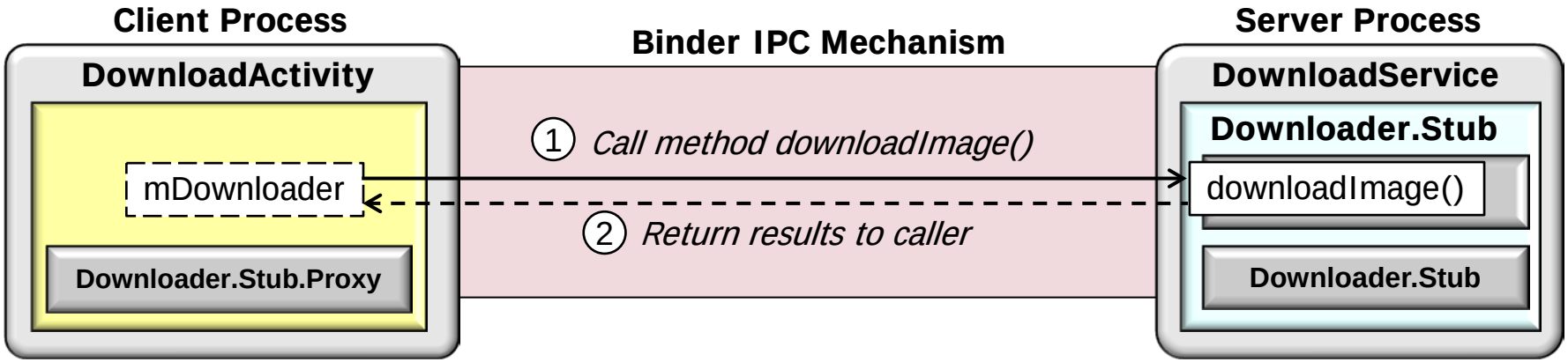
- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone, so Android automates it with the AIDL & an associated compiler
- The Android Binder optimizes communication for cross-process calls



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

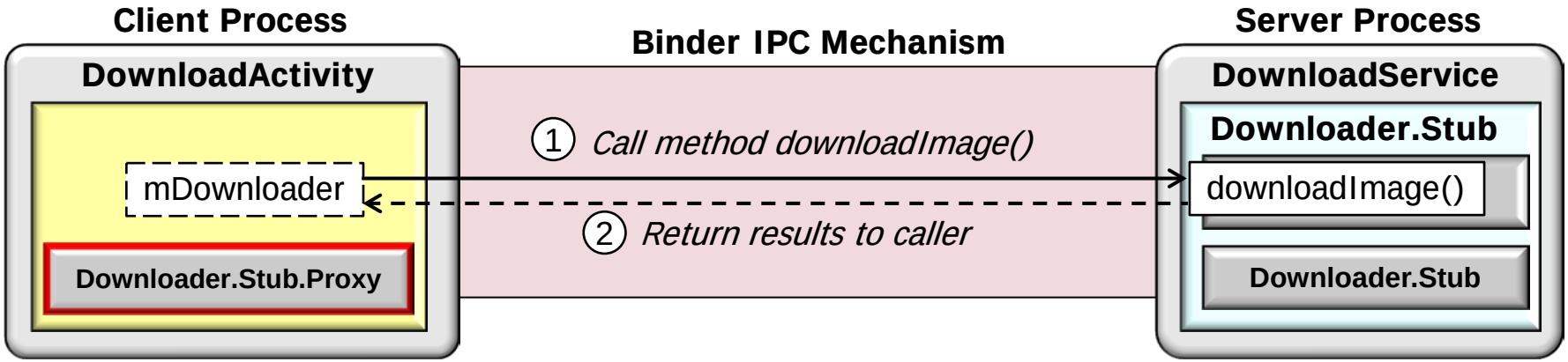
Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone, so Android automates it with the AIDL & an associated compiler
- The Android Binder optimizes communication for cross-process calls
  - Apps rarely access the Binder directly, but instead use AIDL Stubs & Proxies



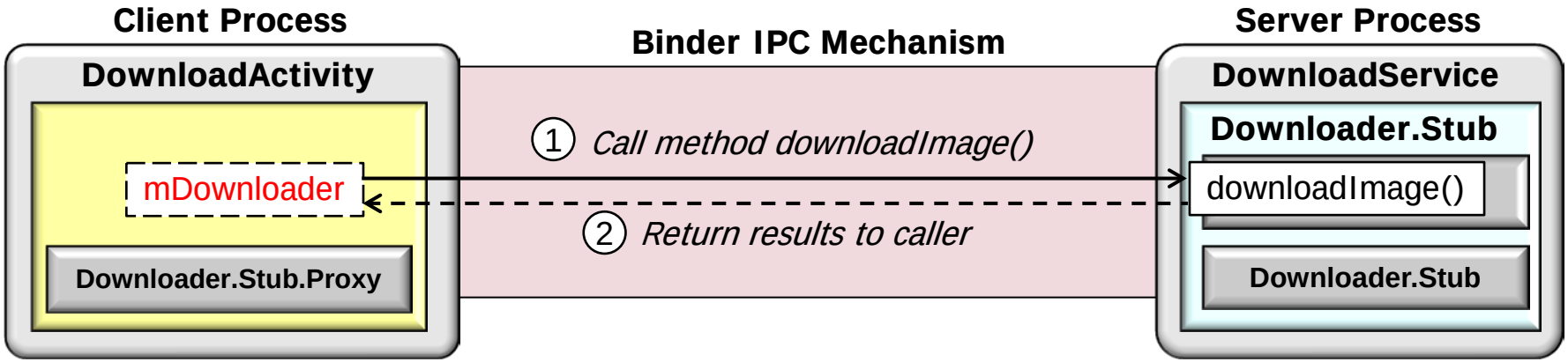
Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone, so Android automates it with the AIDL & an associated compiler
- The Android Binder optimizes communication for cross-process calls
  - Apps rarely access the Binder directly, but instead use AIDL Stubs & Proxies



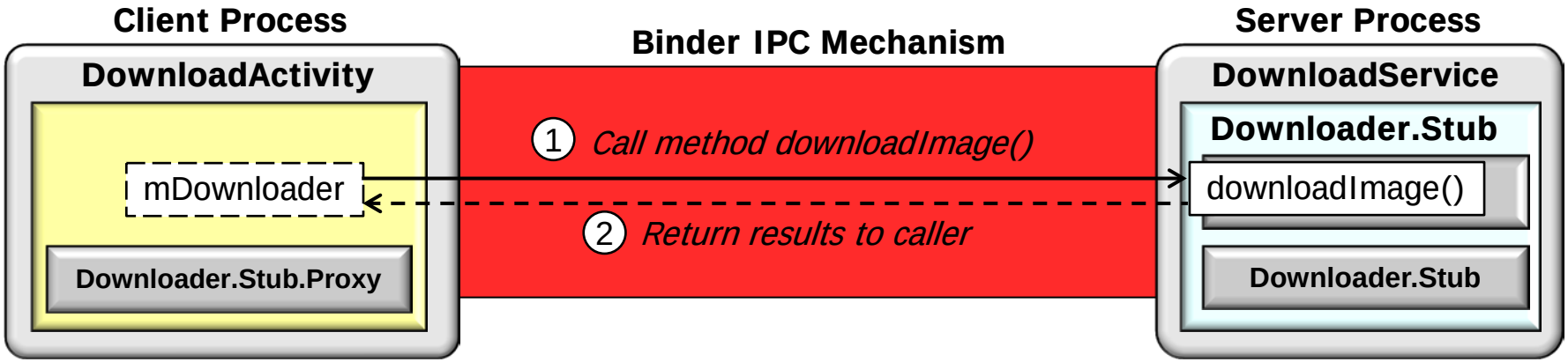
Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone, so Android automates it with the AIDL & an associated compiler
- The Android Binder optimizes communication for cross-process calls
  - Apps rarely access the Binder directly, but instead use AIDL Stubs & Proxies



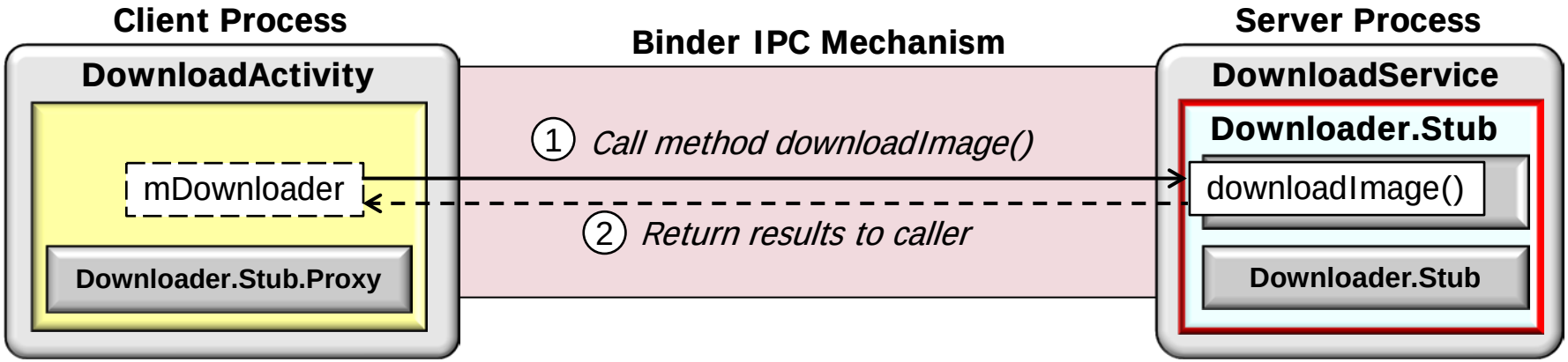
Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone, so Android automates it with the AIDL & an associated compiler
- The Android Binder optimizes communication for cross-process calls
  - Apps rarely access the Binder directly, but instead use AIDL Stubs & Proxies



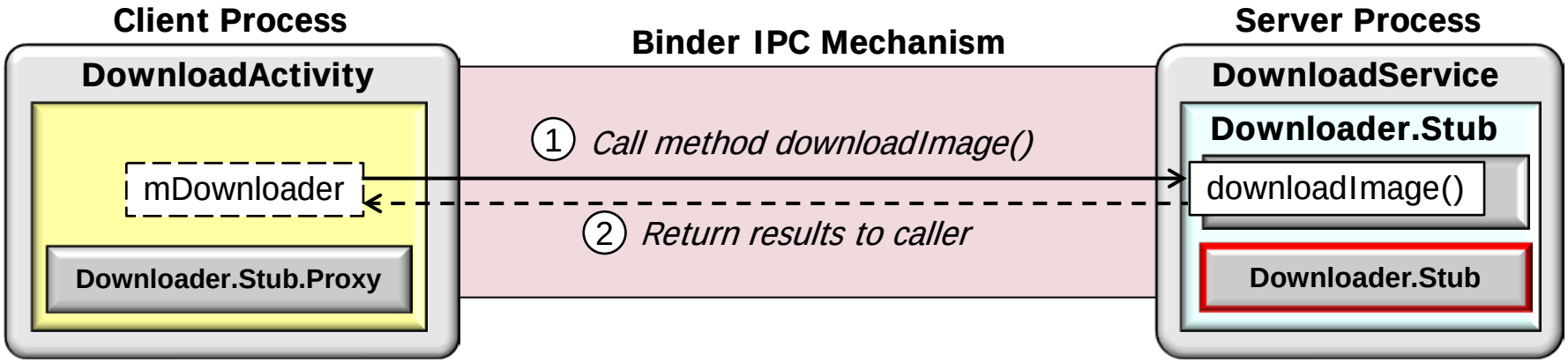
Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone, so Android automates it with the AIDL & an associated compiler
- The Android Binder optimizes communication for cross-process calls
  - Apps rarely access the Binder directly, but instead use AIDL Stubs & Proxies



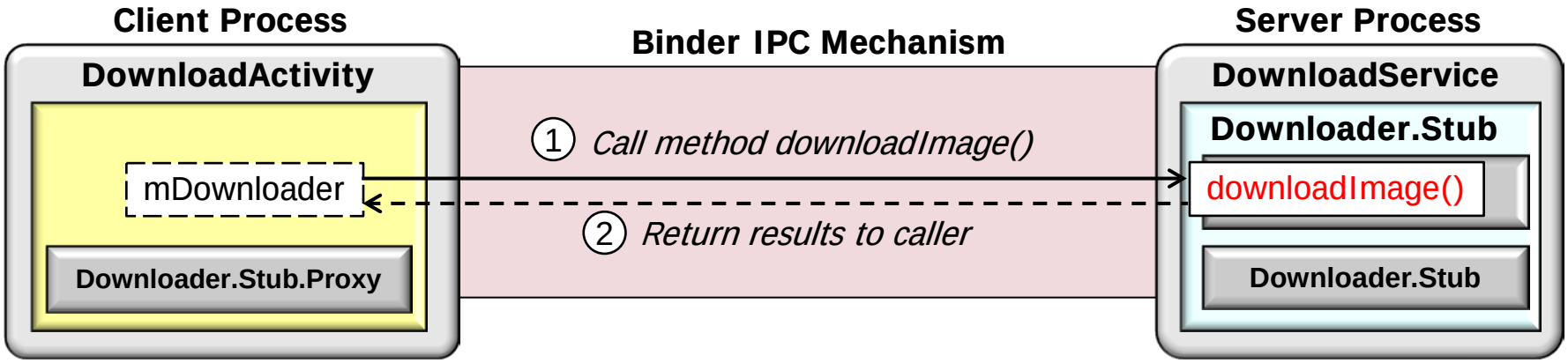
Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone, so Android automates it with the AIDL & an associated compiler
- The Android Binder optimizes communication for cross-process calls
  - Apps rarely access the Binder directly, but instead use AIDL Stubs & Proxies



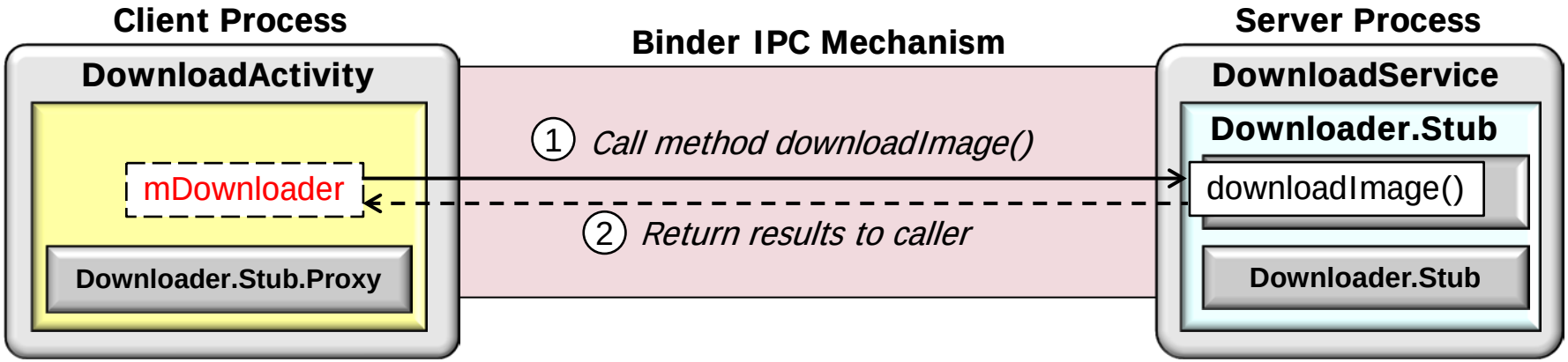
Motivating the AIDL & Binder

- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone, so Android automates it with the AIDL & an associated compiler
- The Android Binder optimizes communication for cross-process calls
  - Apps rarely access the Binder directly, but instead use AIDL Stubs & Proxies



Motivating the AIDL & Binder

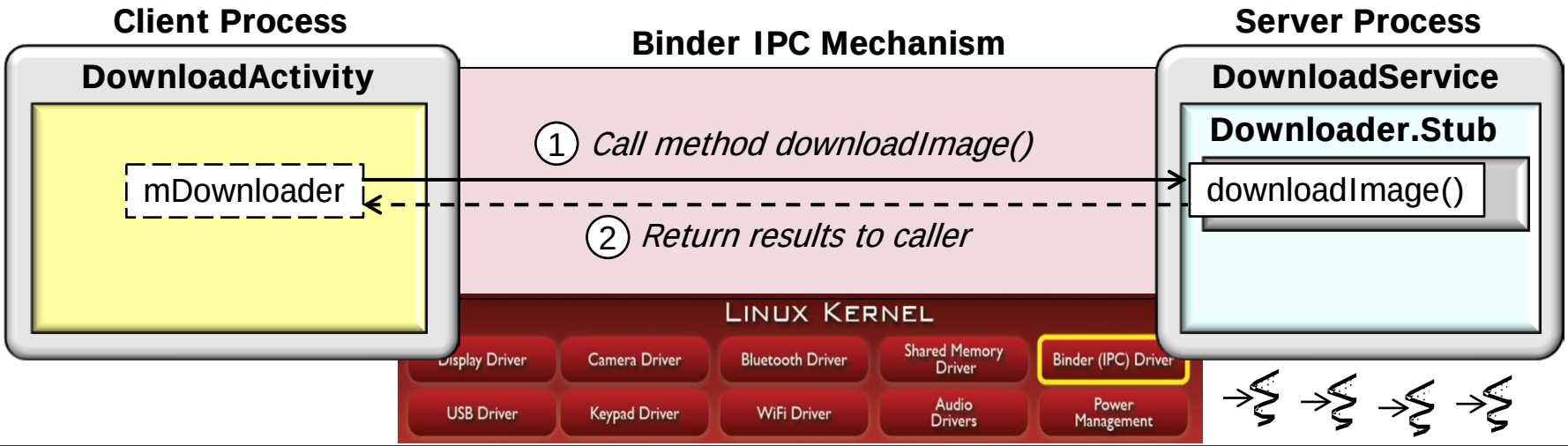
- Services may need to run in different processes than their clients
- Activities that communicate with objects in Services residing in other processes must perform additional steps
- At the heart of these steps are *marshaling* & *demarshaling* mechanisms
- Manually writing code to (de)marshal is tedious & error-prone, so Android automates it with the AIDL & an associated compiler
- The Android Binder optimizes communication for cross-process calls
  - Apps rarely access the Binder directly, but instead use AIDL Stubs & Proxies



Overview of Android Binder & AIDL (Part 1)

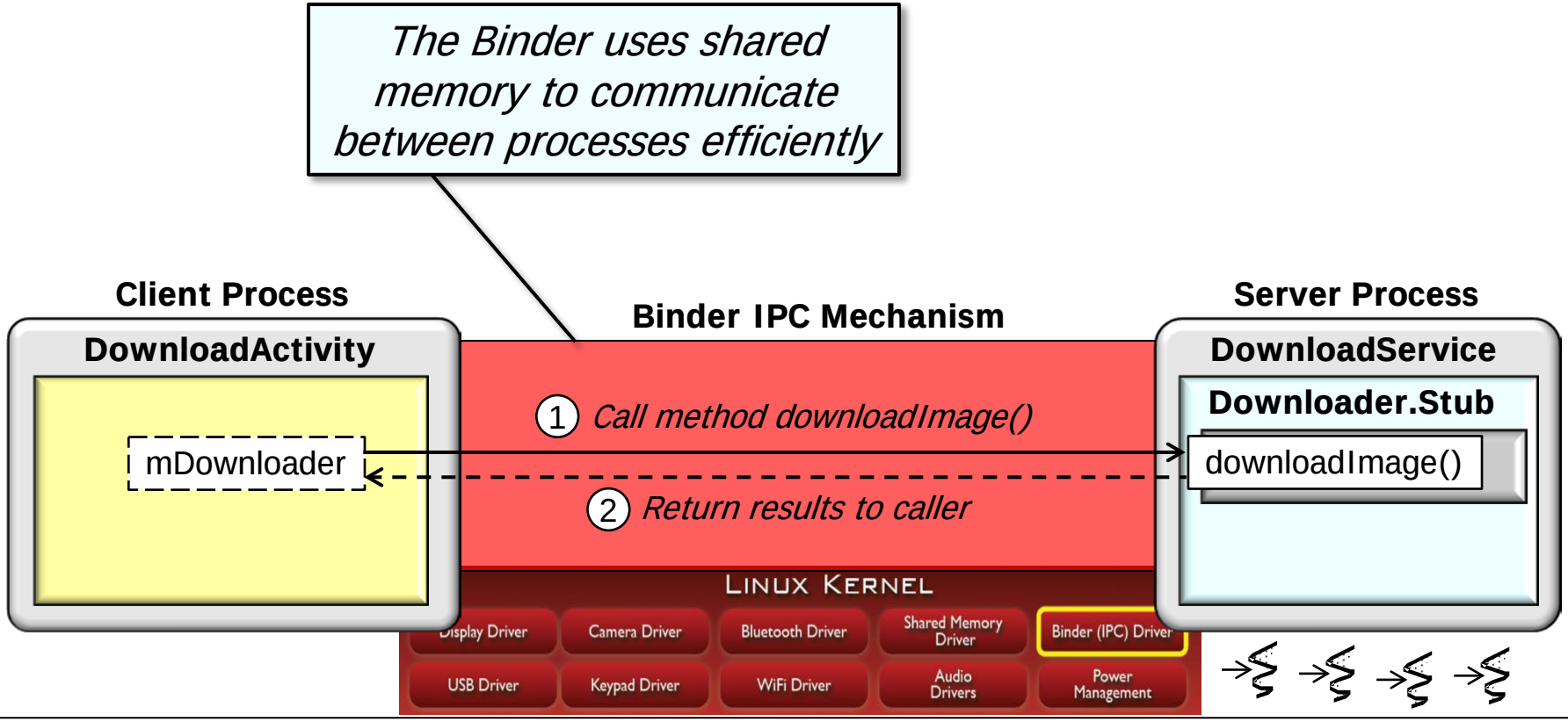
Overview of Android Binder & AIDL

- The Binder Driver runs in the Linux kernel to optimize IPC on Android devices



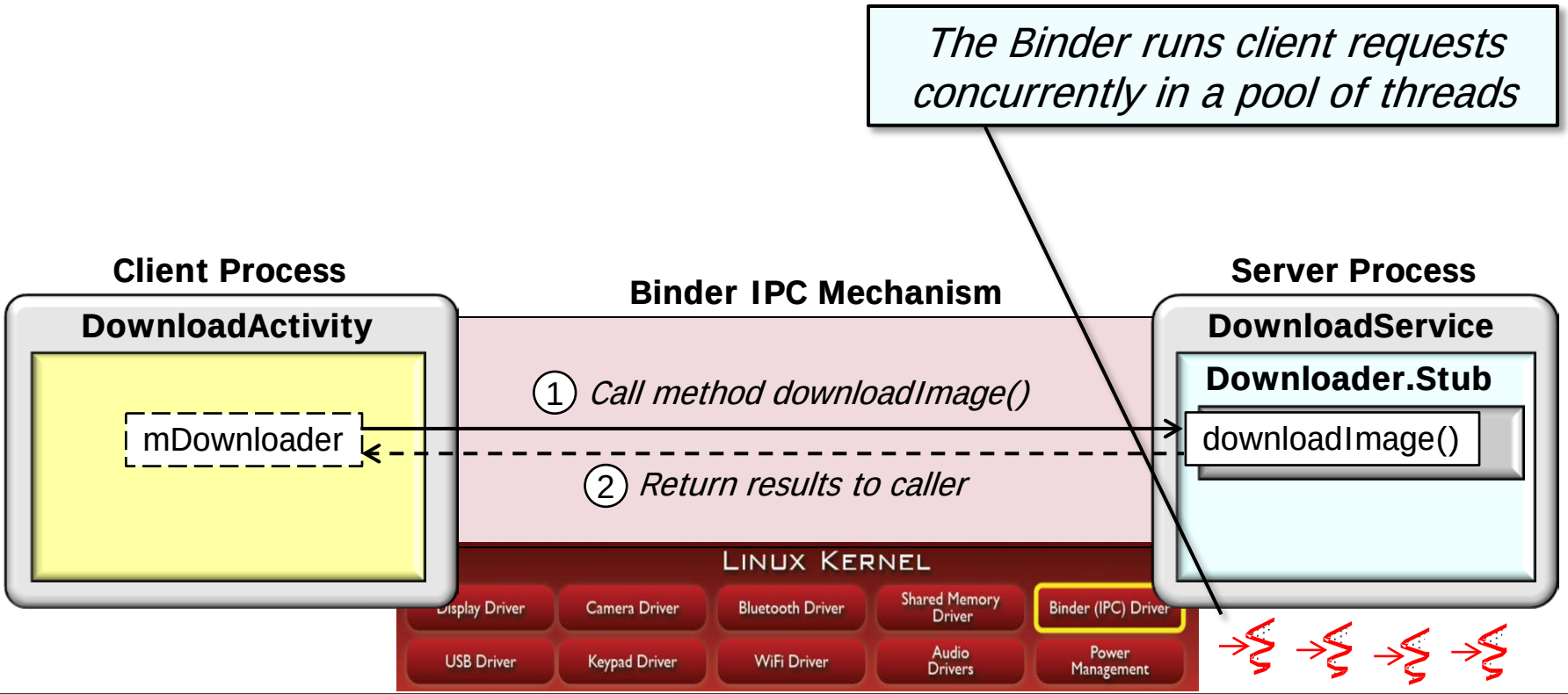
Overview of Android Binder & AIDL

- The Binder Driver runs in the Linux kernel to optimize IPC on Android devices
  - It uses shared memory to minimize data copying



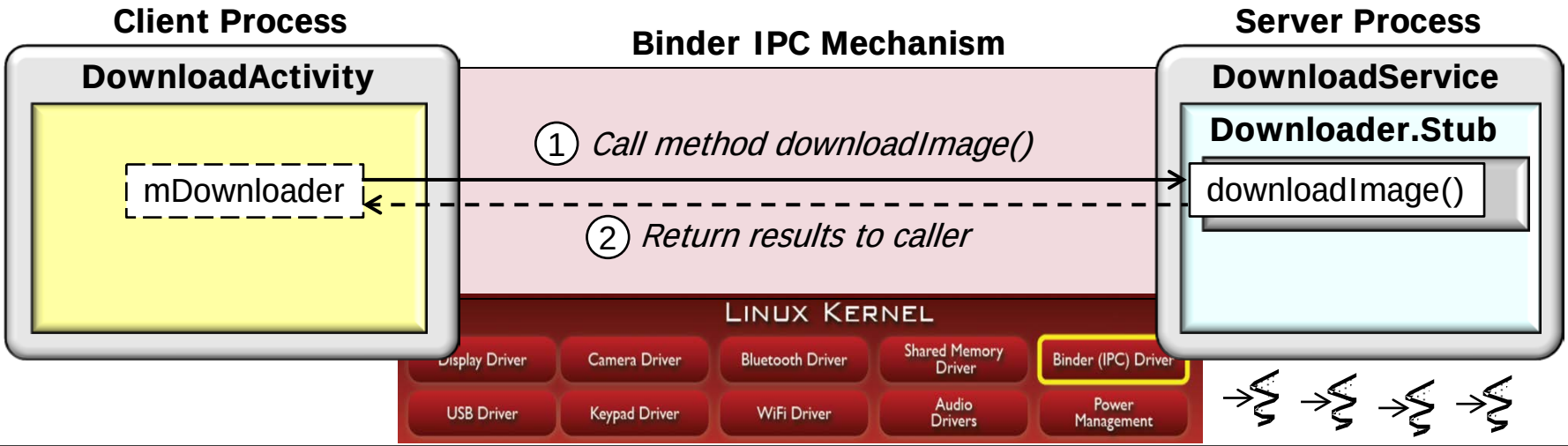
Overview of Android Binder & AIDL

- The Binder Driver runs in the Linux kernel to optimize IPC on Android devices
  - It uses shared memory to minimize data copying
  - It provides a per-process pool of threads



Overview of Android Binder & AIDL

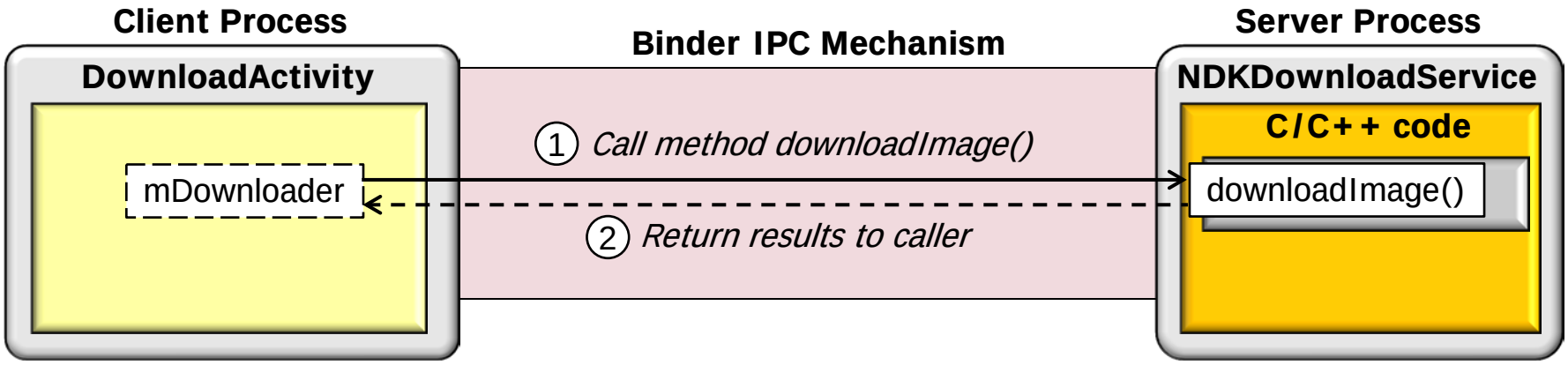
- The Binder Driver runs in the Linux kernel to optimize IPC on Android devices
  - It uses shared memory to minimize data copying
  - It provides a per-process pool of threads



elinux.org/Android_Binder has more info on Android Binder

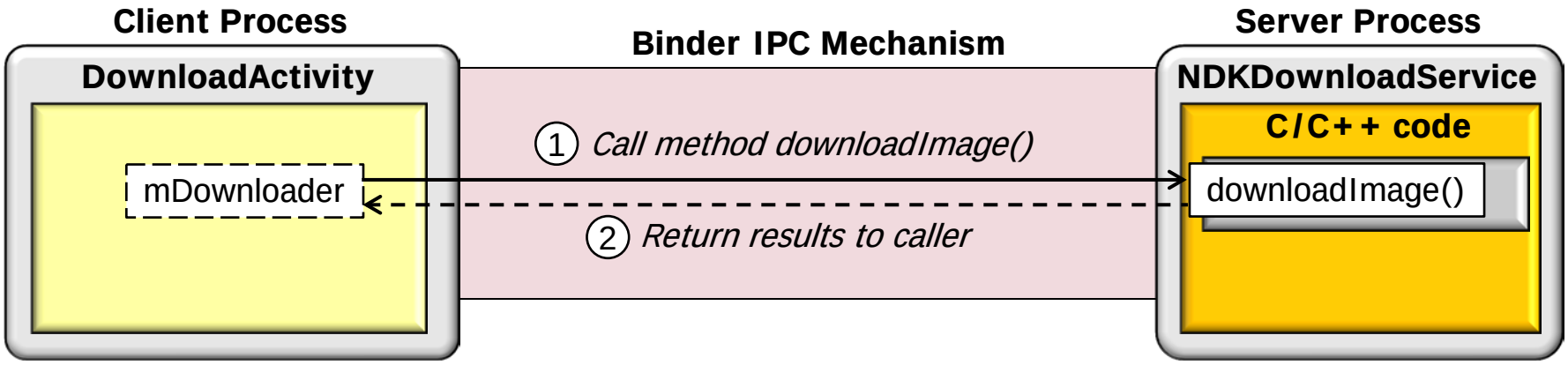
Overview of Android Binder & AIDL

- The Binder Driver runs in the Linux kernel to optimize IPC on Android devices
- AIDL also simplifies inter language communication since Services can be written in Java, as well as C/C++



Overview of Android Binder & AIDL

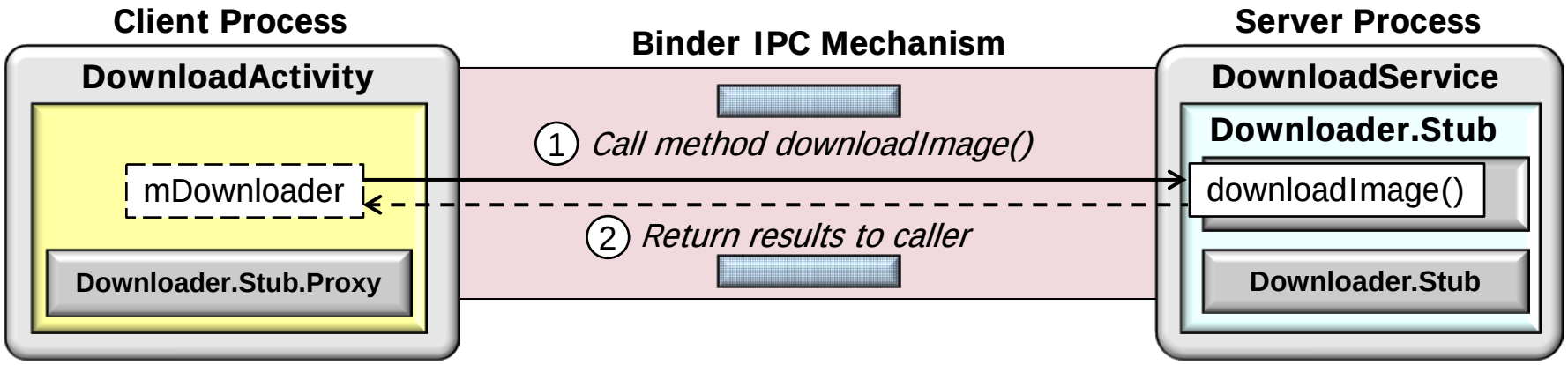
- The Binder Driver runs in the Linux kernel to optimize IPC on Android devices
- AIDL also simplifies inter language communication since Services can be written in Java, as well as C/C++
  - Many Android system services are written in C++, including the Media Player Service, Sensor Service, & Audio/Surface Flinger Sevices



sites.google.com/site/io/anatomy—physiology-of-an-android has more info

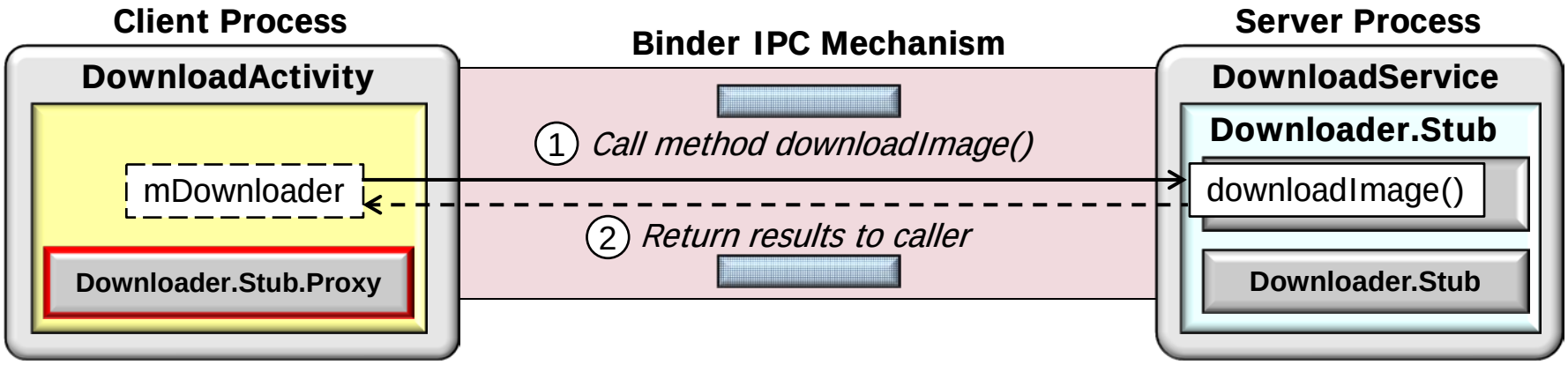
Overview of Android Binder & AIDL

- The Binder Driver runs in the Linux kernel to optimize IPC on Android devices
- AIDL also simplifies inter language communication since Services can be written in Java, as well as C/C++
- The Proxies & Stubs generated by the AIDL compiler are used at several points in the path from caller to callee



Overview of Android Binder & AIDL

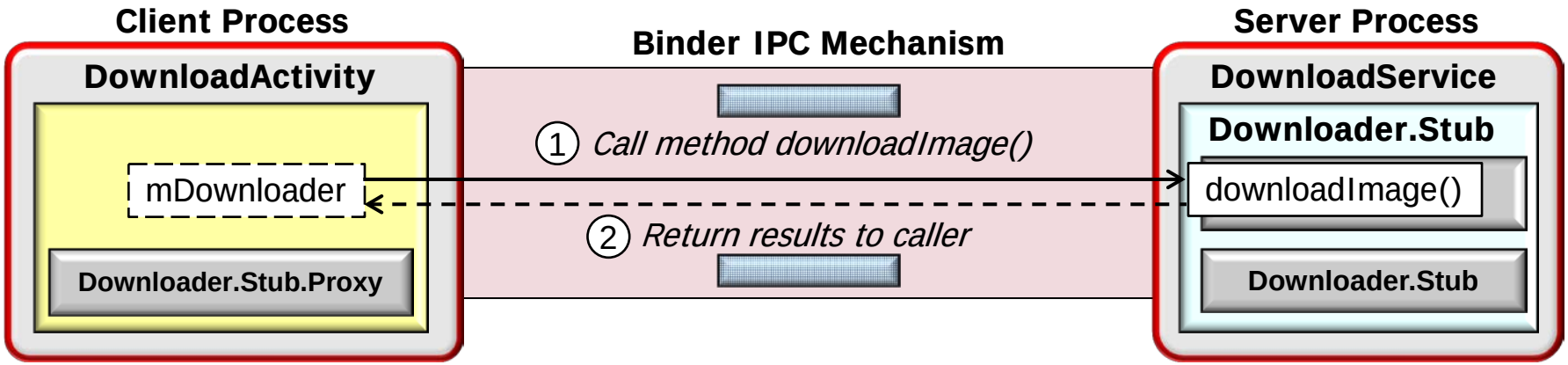
- The Binder Driver runs in the Linux kernel to optimize IPC on Android devices
- AIDL also simplifies inter language communication since Services can be written in Java, as well as C/C++
- The Proxies & Stubs generated by the AIDL compiler are used at several points in the path from caller to callee
  - Caller's data is marshaled into parcels



developer.android.com/reference/android/os/Parcel.html has more info

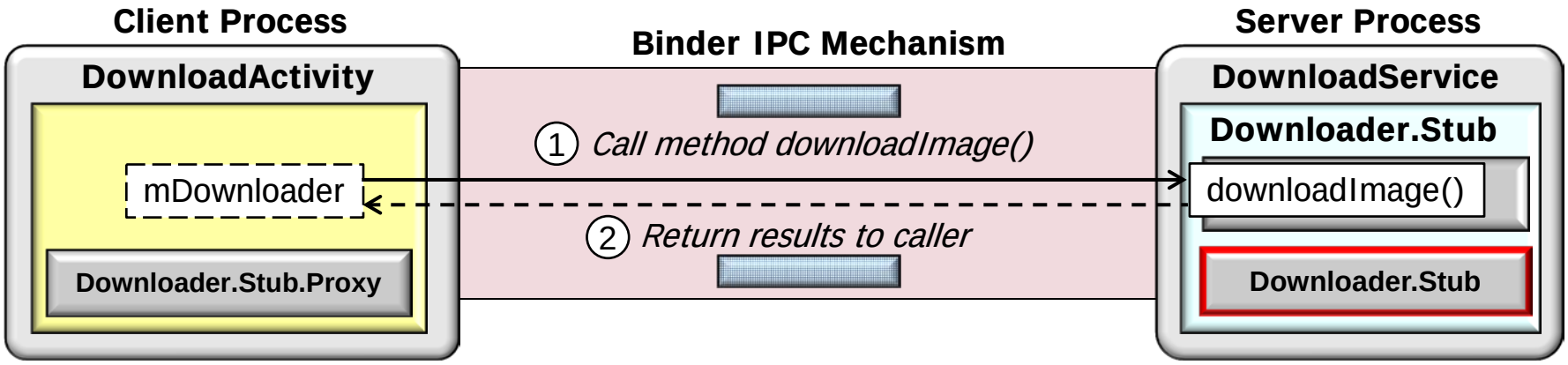
Overview of Android Binder & AIDL

- The Binder Driver runs in the Linux kernel to optimize IPC on Android devices
- AIDL also simplifies inter language communication since Services can be written in Java, as well as C/C++
- The Proxies & Stubs generated by the AIDL compiler are used at several points in the path from caller to callee
  - Caller's data is marshaled into parcels
  - Copied from caller's process to callee's process



Overview of Android Binder & AIDL

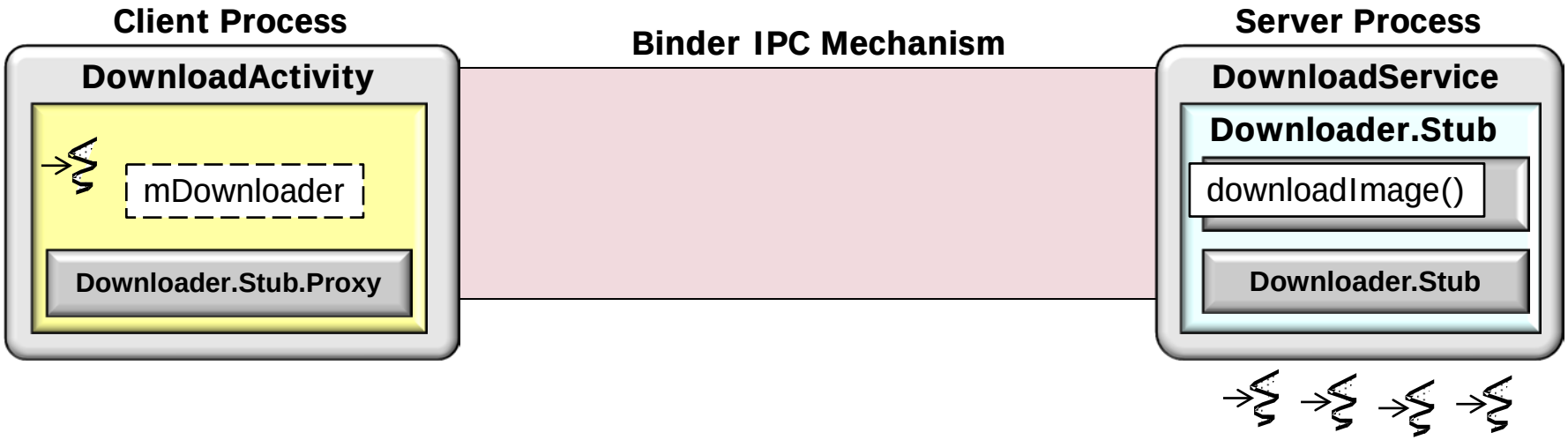
- The Binder Driver runs in the Linux kernel to optimize IPC on Android devices
- AIDL also simplifies inter language communication since Services can be written in Java, as well as C/C++
- The Proxies & Stubs generated by the AIDL compiler are used at several points in the path from caller to callee
  - Caller's data is marshaled into parcels
  - Copied from caller's process to callee's process
  - Demarshaled into the format expected by the callee



Overview of Android Binder & AIDL (Part 2)

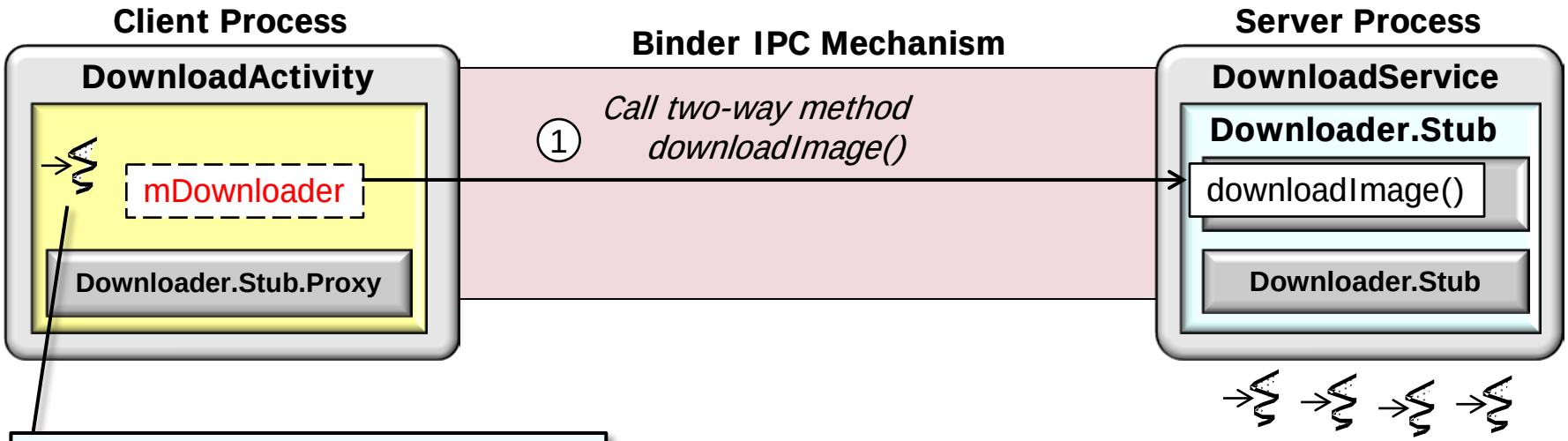
Overview of Android Binder & AIDL

- The Binder Driver is installed in the Linux kernel to accelerate IPC
- Android (system) Services can be written in Java, as well as C/C++
- Caller's data is marshaled into parcels, copied to callee's process, & demarshaled into what callee expects
- Two-way method invocations are synchronous (block the caller)



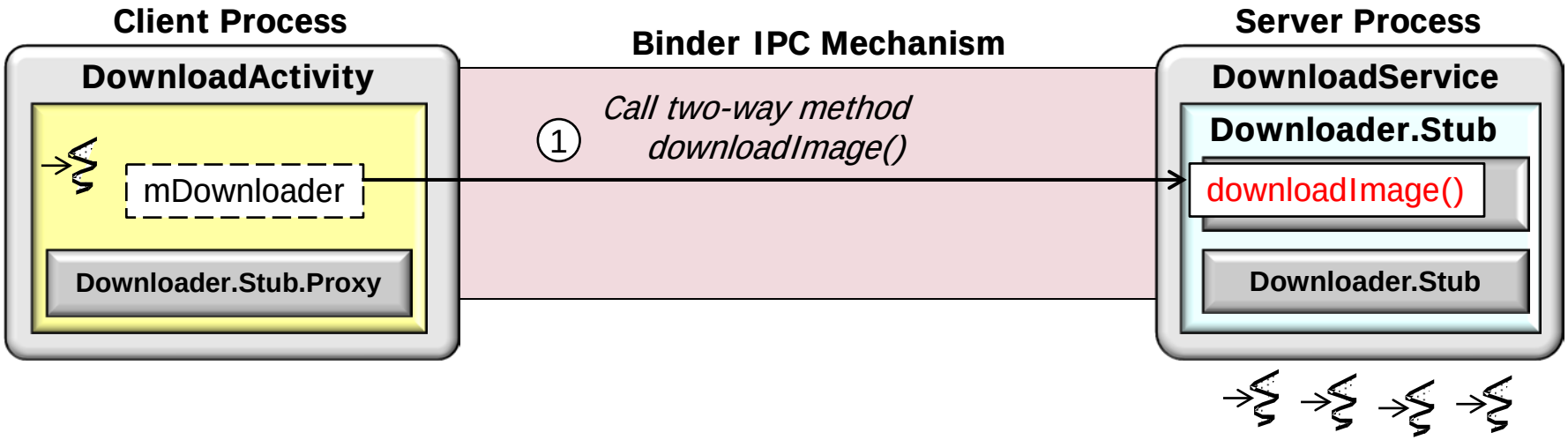
Overview of Android Binder & AIDL

- The Binder Driver is installed in the Linux kernel to accelerate IPC
- Android (system) Services can be written in Java, as well as C/C++
- Caller's data is marshaled into parcels, copied to callee's process, & demarshaled into what callee expects
- Two-way method invocations are synchronous (block the caller)



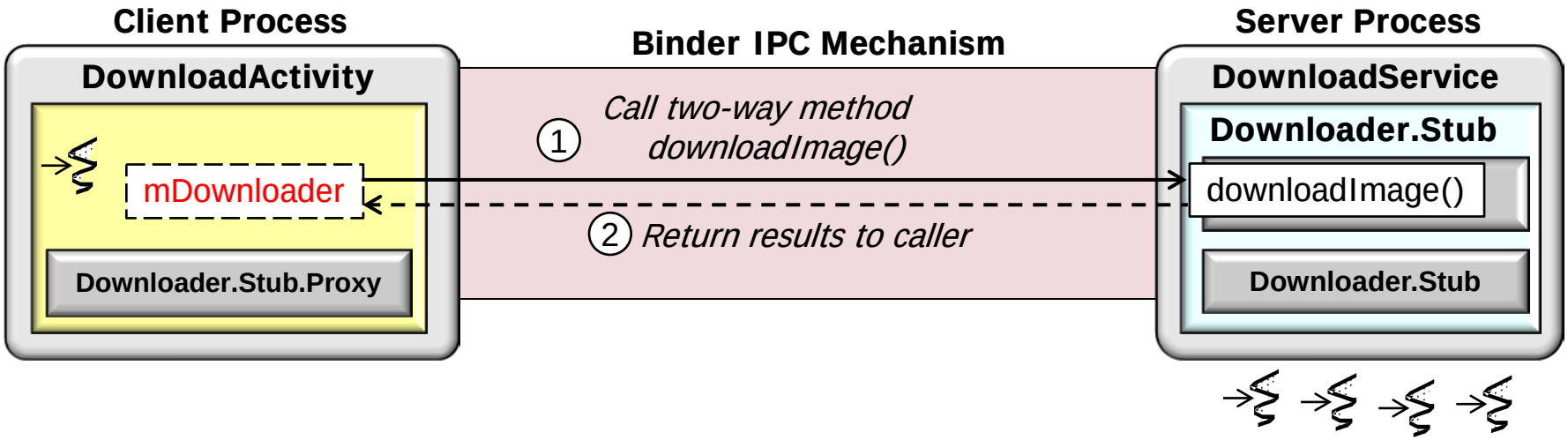
Overview of Android Binder & AIDL

- The Binder Driver is installed in the Linux kernel to accelerate IPC
- Android (system) Services can be written in Java, as well as C/C++
- Caller's data is marshaled into parcels, copied to callee's process, & demarshaled into what callee expects
- Two-way method invocations are synchronous (block the caller)



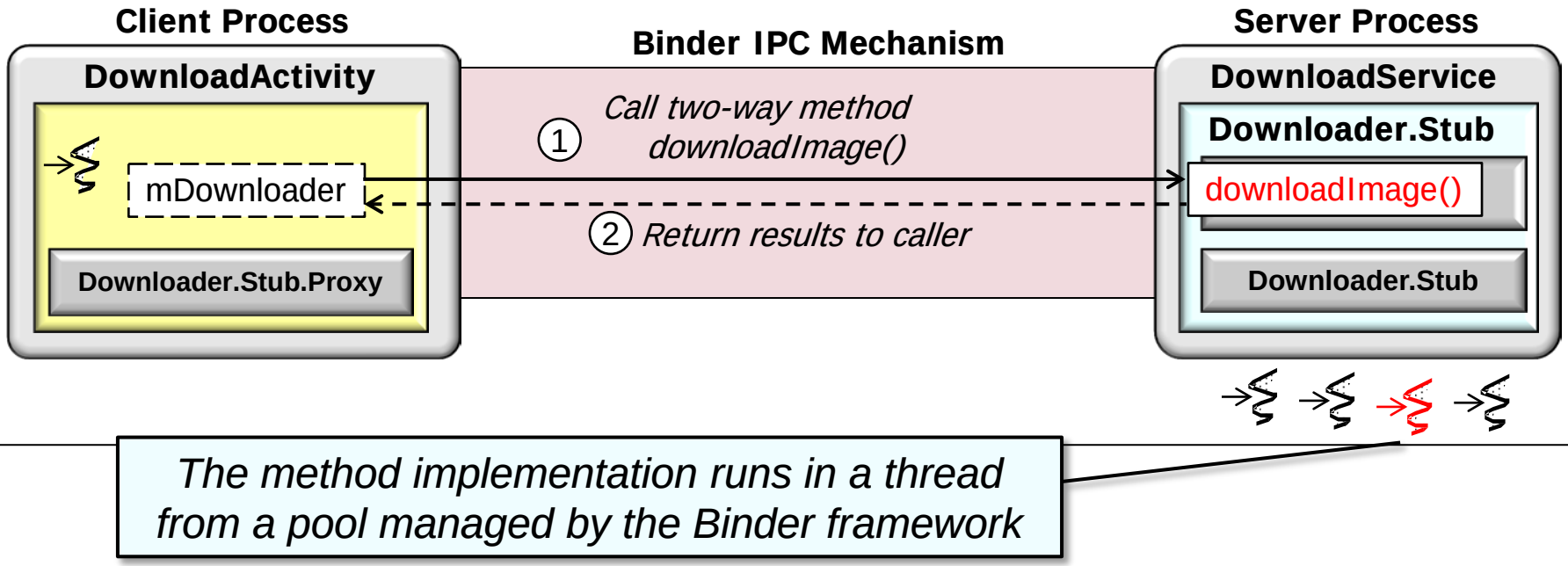
Overview of Android Binder & AIDL

- The Binder Driver is installed in the Linux kernel to accelerate IPC
- Android (system) Services can be written in Java, as well as C/C++
- Caller's data is marshaled into parcels, copied to callee's process, & demarshaled into what callee expects
- Two-way method invocations are synchronous (block the caller)



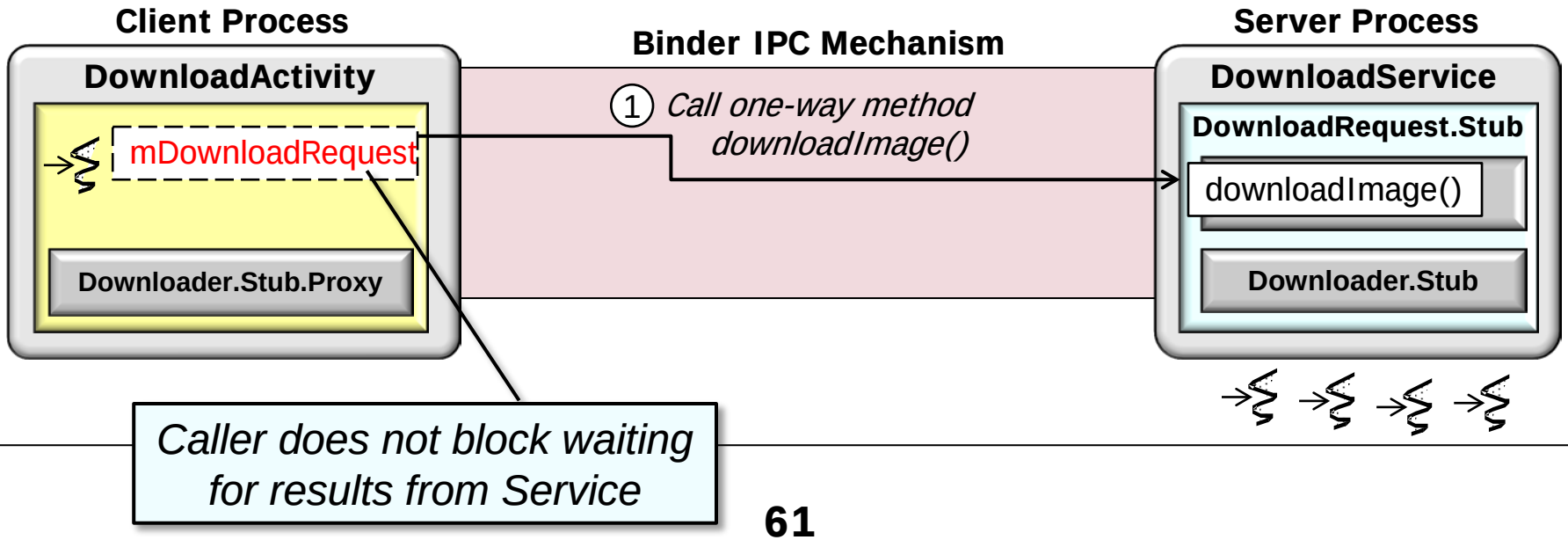
Overview of Android Binder & AIDL

- The Binder Driver is installed in the Linux kernel to accelerate IPC
- Android (system) Services can be written in Java, as well as C/C++
- Caller's data is marshaled into parcels, copied to callee's process, & demarshaled into what callee expects
- Two-way method invocations are synchronous (block the caller)



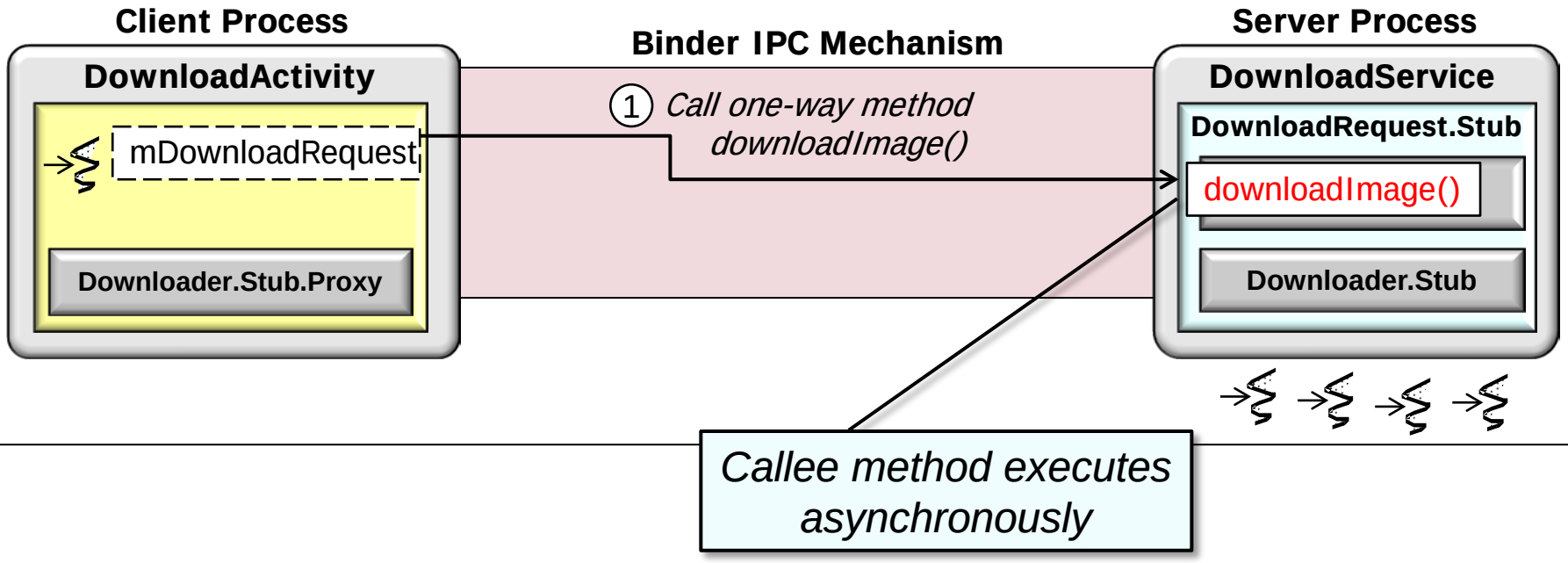
Overview of Android Binder & AIDL

- The Binder Driver is installed in the Linux kernel to accelerate IPC
- Android (system) Services can be written in Java, as well as C/C++
- Caller's data is marshaled into parcels, copied to callee's process, & demarshaled into what callee expects
- Two-way method invocations are synchronous (block the caller)
  - One-way method invocations behave differently than two-way methods



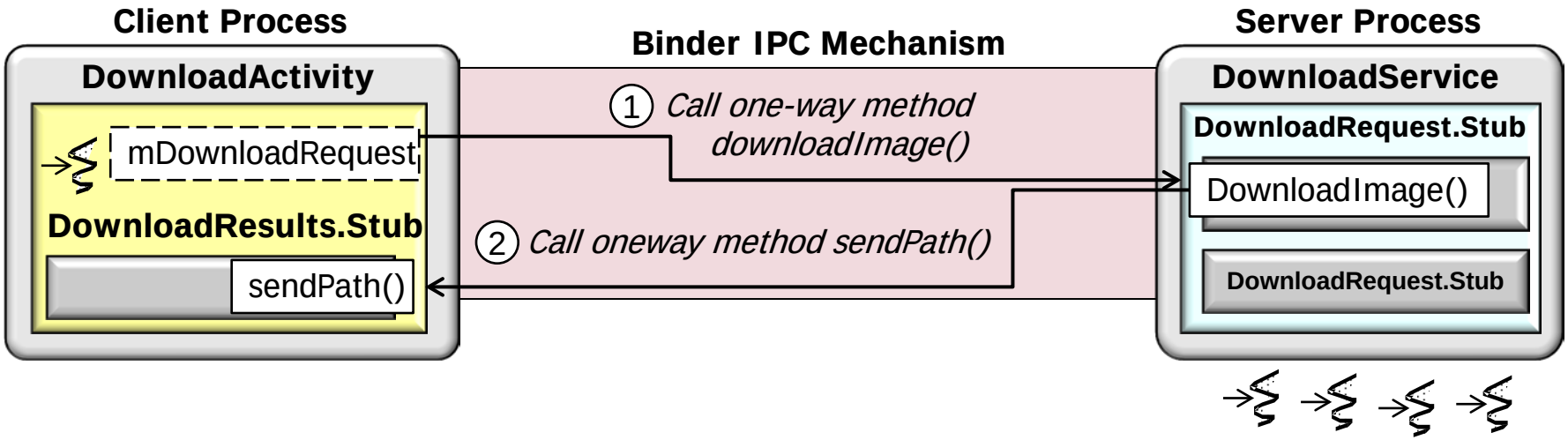
Overview of Android Binder & AIDL

- The Binder Driver is installed in the Linux kernel to accelerate IPC
- Android (system) Services can be written in Java, as well as C/C++
- Caller's data is marshaled into parcels, copied to callee's process, & demarshaled into what callee expects
- Two-way method invocations are synchronous (block the caller)
  - One-way method invocations behave differently than two-way methods



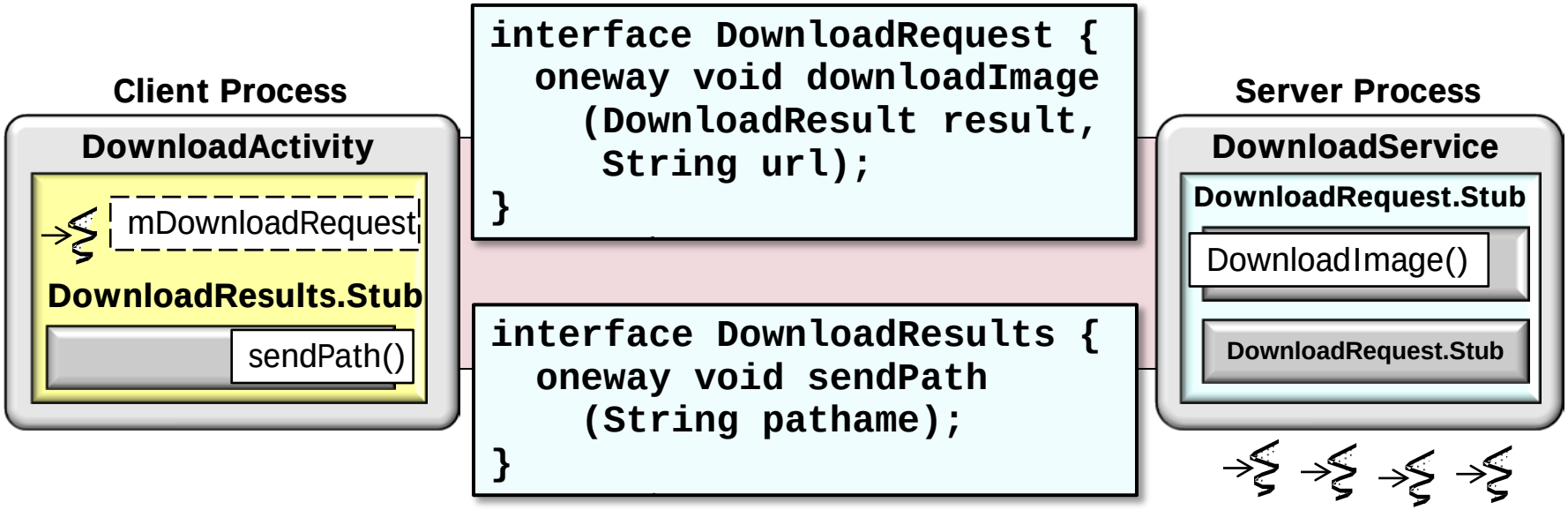
Overview of Android Binder & AIDL

- The Binder Driver is installed in the Linux kernel to accelerate IPC
- Android (system) Services can be written in Java, as well as C/C++
- Caller's data is marshaled into parcels, copied to callee's process, & demarshaled into what callee expects
- Two-way method invocations are synchronous (block the caller)
- Android also supports two-way asynchronous calls between processes



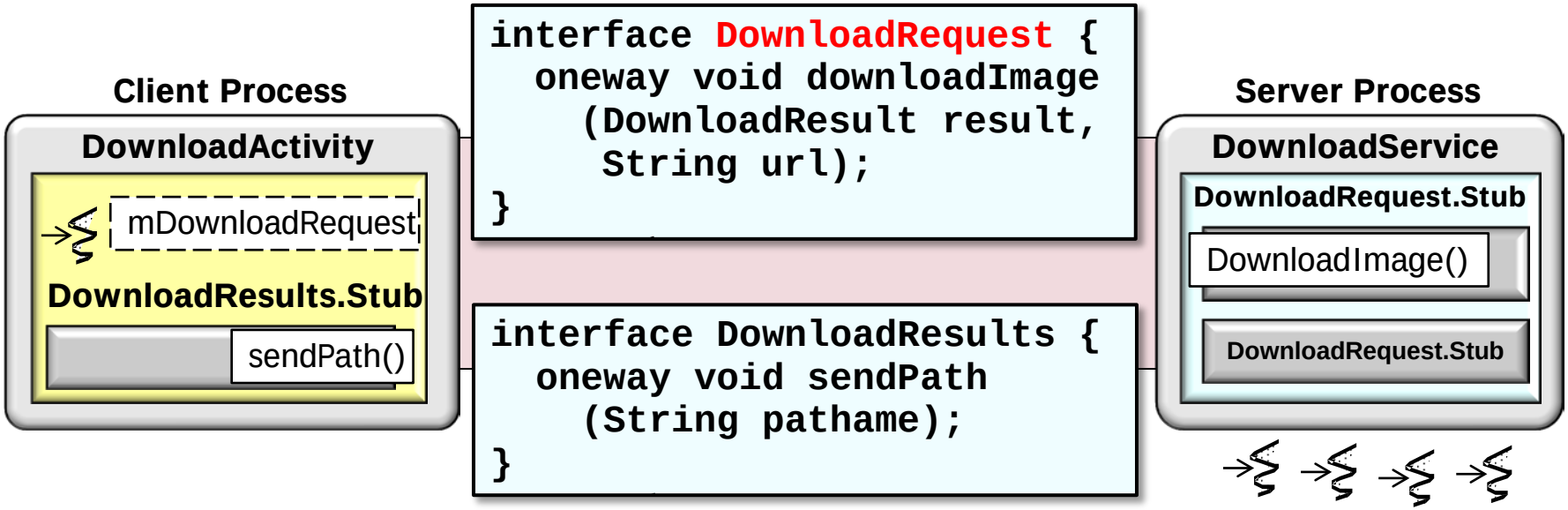
Overview of Android Binder & AIDL

- The Binder Driver is installed in the Linux kernel to accelerate IPC
- Android (system) Services can be written in Java, as well as C/C++
- Caller's data is marshaled into parcels, copied to callee's process, & demarshaled into what callee expects
- Two-way method invocations are synchronous (block the caller)
- Android also supports two-way asynchronous calls between processes
  - Implemented using two AIDL interfaces with one-way methods



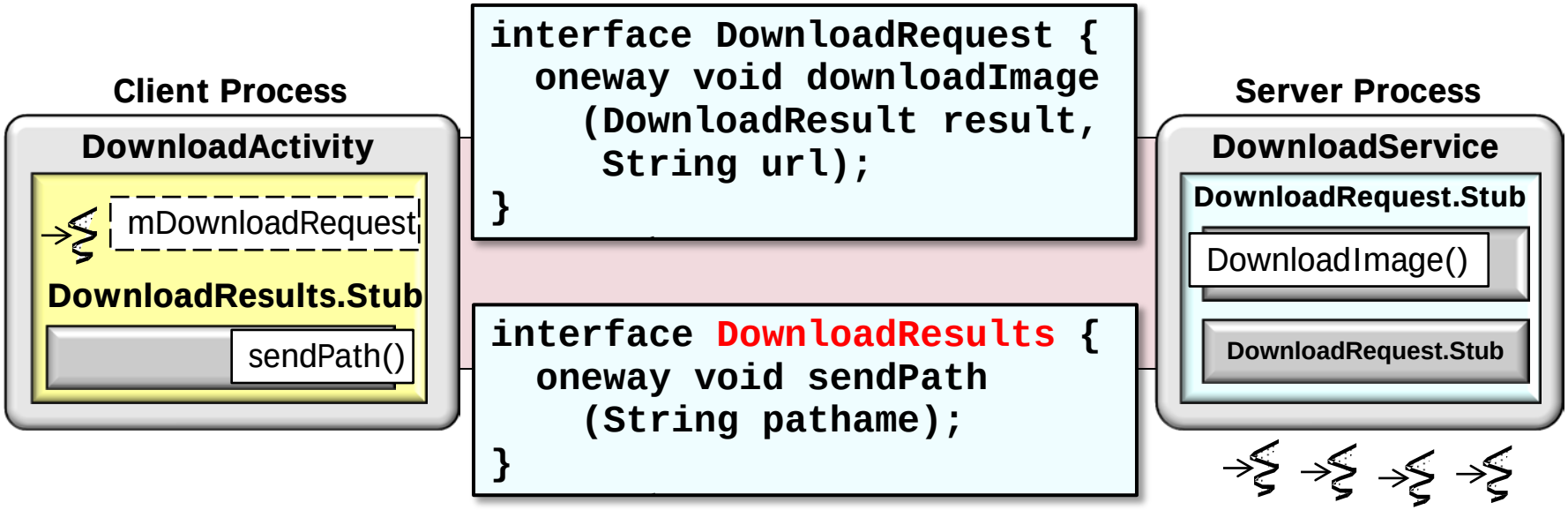
Overview of Android Binder & AIDL

- The Binder Driver is installed in the Linux kernel to accelerate IPC
- Android (system) Services can be written in Java, as well as C/C++
- Caller's data is marshaled into parcels, copied to callee's process, & demarshaled into what callee expects
- Two-way method invocations are synchronous (block the caller)
- Android also supports two-way asynchronous calls between processes
  - Implemented using two AIDL interfaces with one-way methods



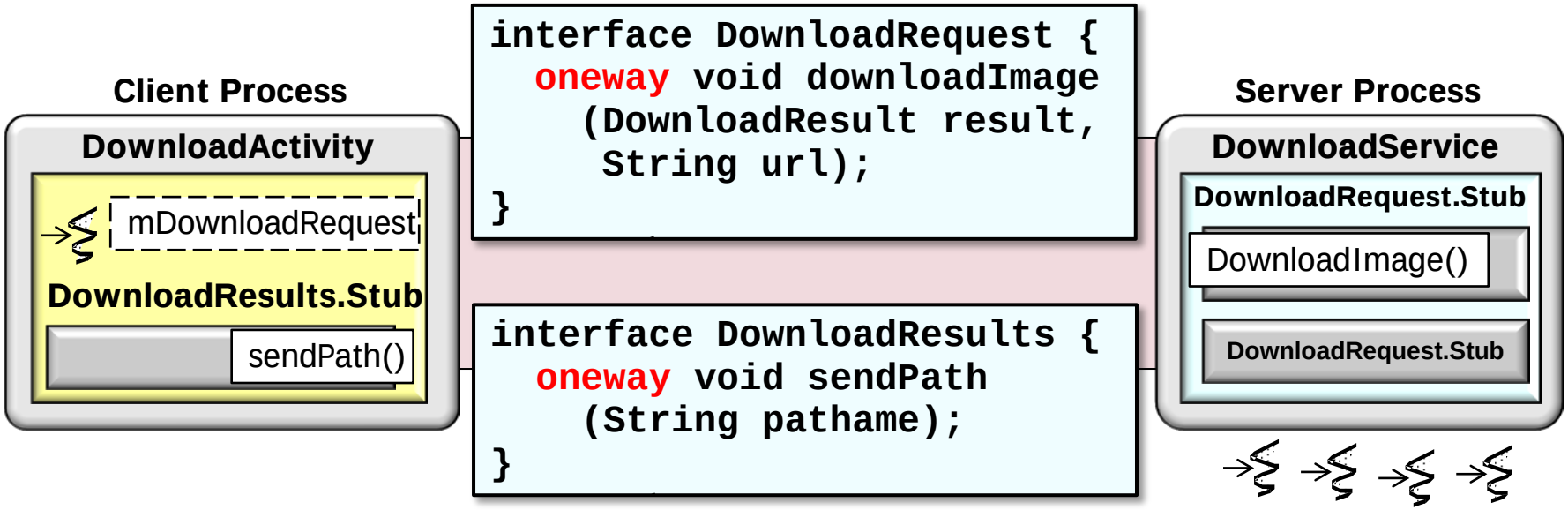
Overview of Android Binder & AIDL

- The Binder Driver is installed in the Linux kernel to accelerate IPC
- Android (system) Services can be written in Java, as well as C/C++
- Caller's data is marshaled into parcels, copied to callee's process, & demarshaled into what callee expects
- Two-way method invocations are synchronous (block the caller)
- Android also supports two-way asynchronous calls between processes
  - Implemented using two AIDL interfaces with one-way methods



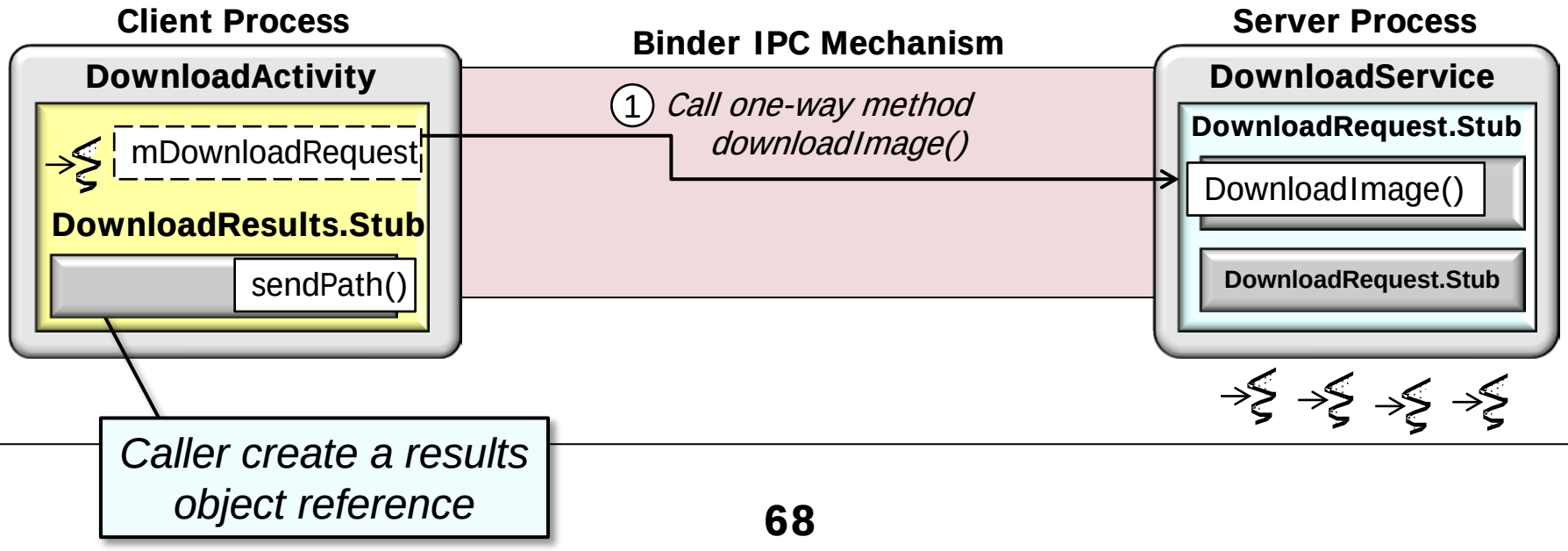
Overview of Android Binder & AIDL

- The Binder Driver is installed in the Linux kernel to accelerate IPC
- Android (system) Services can be written in Java, as well as C/C++
- Caller's data is marshaled into parcels, copied to callee's process, & demarshaled into what callee expects
- Two-way method invocations are synchronous (block the caller)
- Android also supports two-way asynchronous calls between processes
  - Implemented using two AIDL interfaces with one-way methods



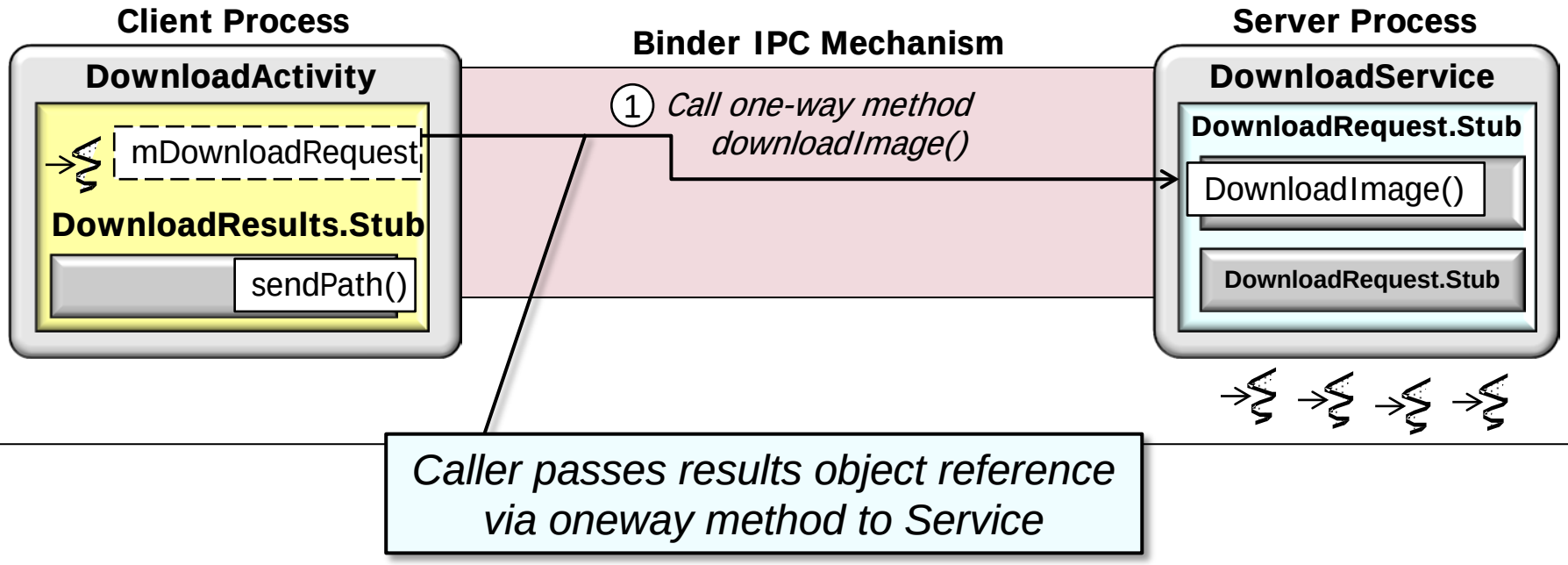
Overview of Android Binder & AIDL

- The Binder Driver is installed in the Linux kernel to accelerate IPC
- Android (system) Services can be written in Java, as well as C/C++
- Caller's data is marshaled into parcels, copied to callee's process, & demarshaled into what callee expects
- Two-way method invocations are synchronous (block the caller)
- Android also supports two-way asynchronous calls between processes
  - Implemented using two AIDL interfaces with one-way methods



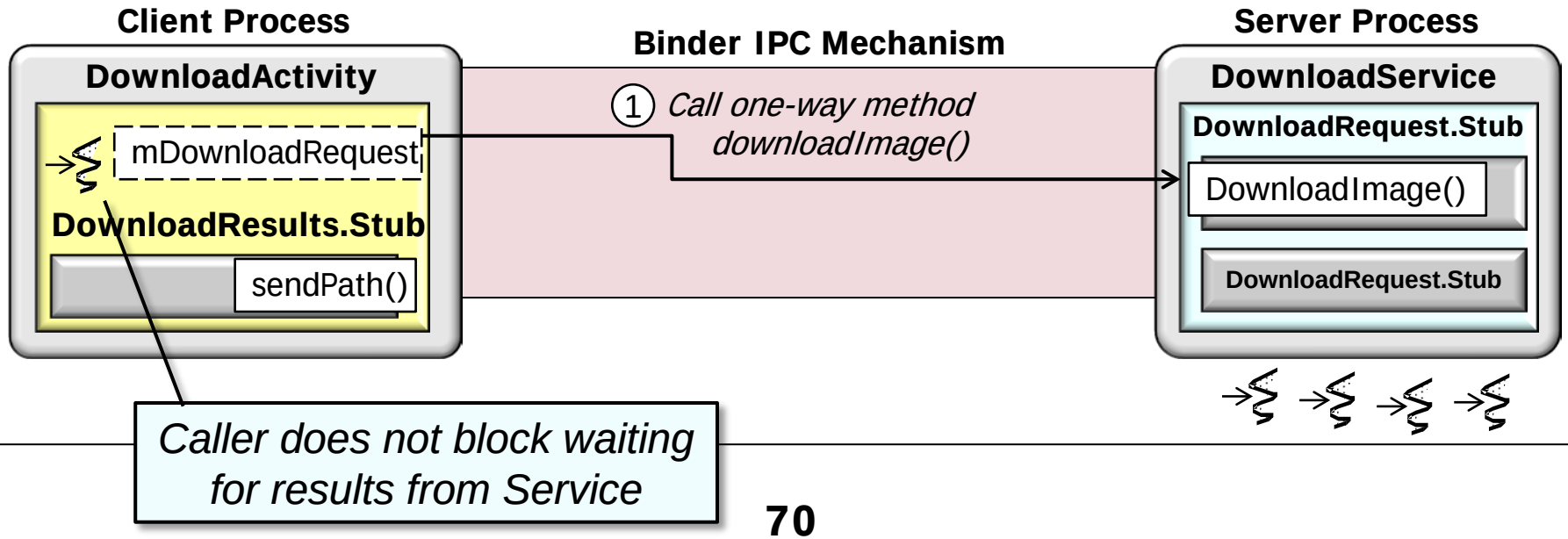
Overview of Android Binder & AIDL

- The Binder Driver is installed in the Linux kernel to accelerate IPC
- Android (system) Services can be written in Java, as well as C/C++
- Caller's data is marshaled into parcels, copied to callee's process, & demarshaled into what callee expects
- Two-way method invocations are synchronous (block the caller)
- Android also supports two-way asynchronous calls between processes
  - Implemented using two AIDL interfaces with one-way methods



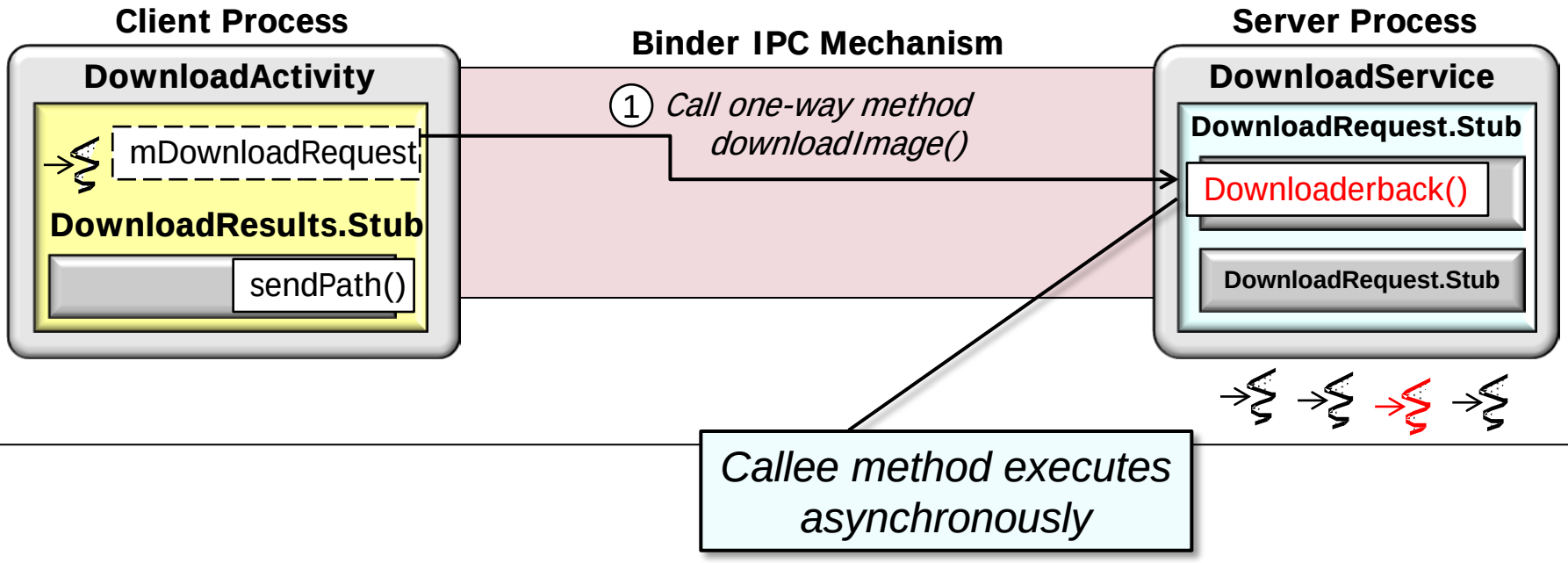
Overview of Android Binder & AIDL

- The Binder Driver is installed in the Linux kernel to accelerate IPC
- Android (system) Services can be written in Java, as well as C/C++
- Caller's data is marshaled into parcels, copied to callee's process, & demarshaled into what callee expects
- Two-way method invocations are synchronous (block the caller)
- Android also supports two-way asynchronous calls between processes
  - Implemented using two AIDL interfaces with one-way methods



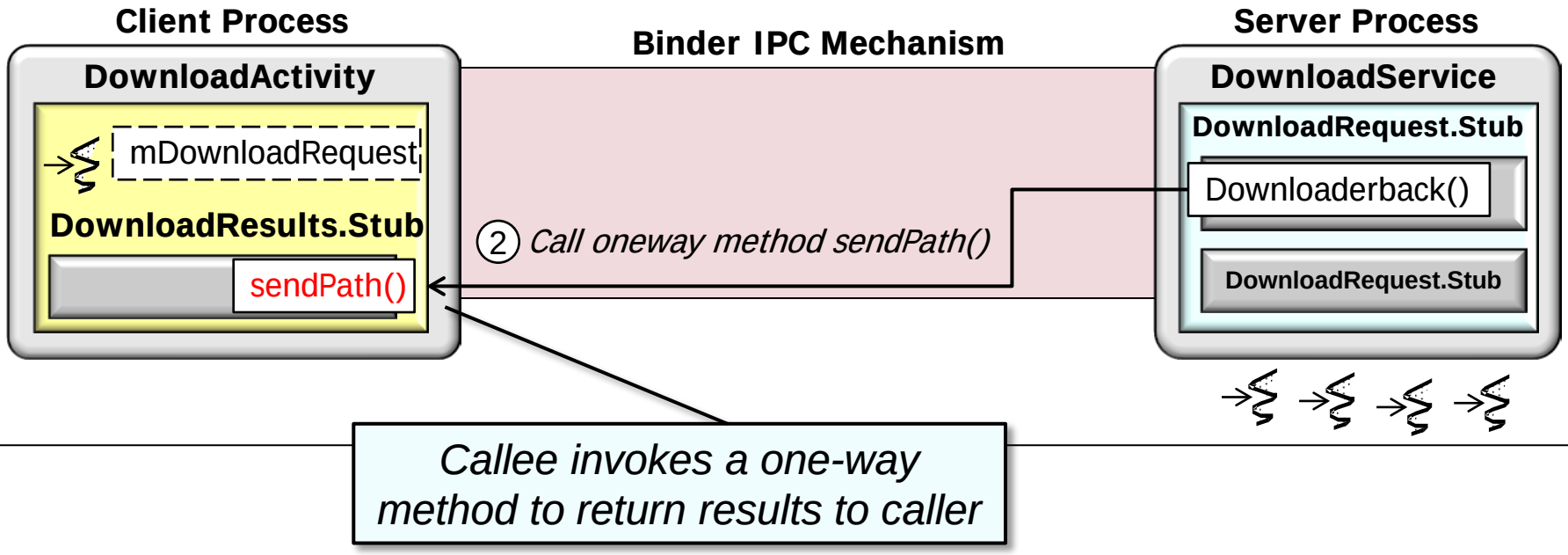
Overview of Android Binder & AIDL

- The Binder Driver is installed in the Linux kernel to accelerate IPC
- Android (system) Services can be written in Java, as well as C/C++
- Caller's data is marshaled into parcels, copied to callee's process, & demarshaled into what callee expects
- Two-way method invocations are synchronous (block the caller)
- Android also supports two-way asynchronous calls between processes
  - Implemented using two AIDL interfaces with one-way methods



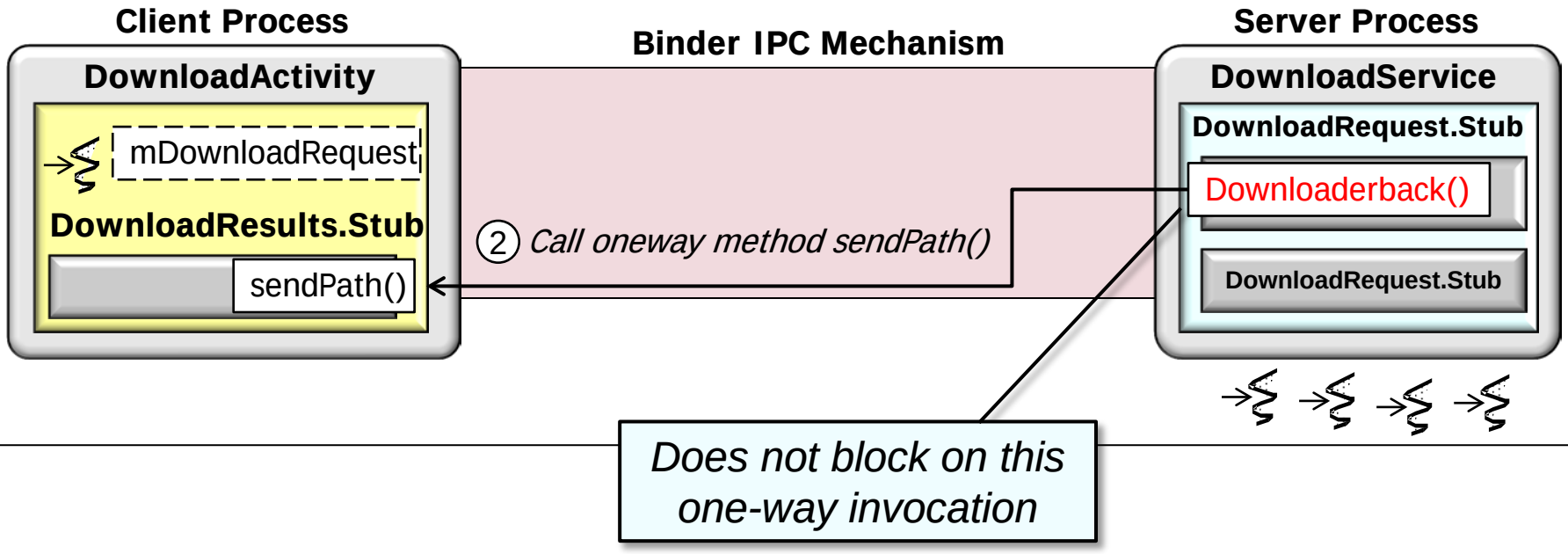
Overview of Android Binder & AIDL

- The Binder Driver is installed in the Linux kernel to accelerate IPC
- Android (system) Services can be written in Java, as well as C/C++
- Caller's data is marshaled into parcels, copied to callee's process, & demarshaled into what callee expects
- Two-way method invocations are synchronous (block the caller)
- Android also supports two-way asynchronous calls between processes
  - Implemented using two AIDL interfaces with one-way methods



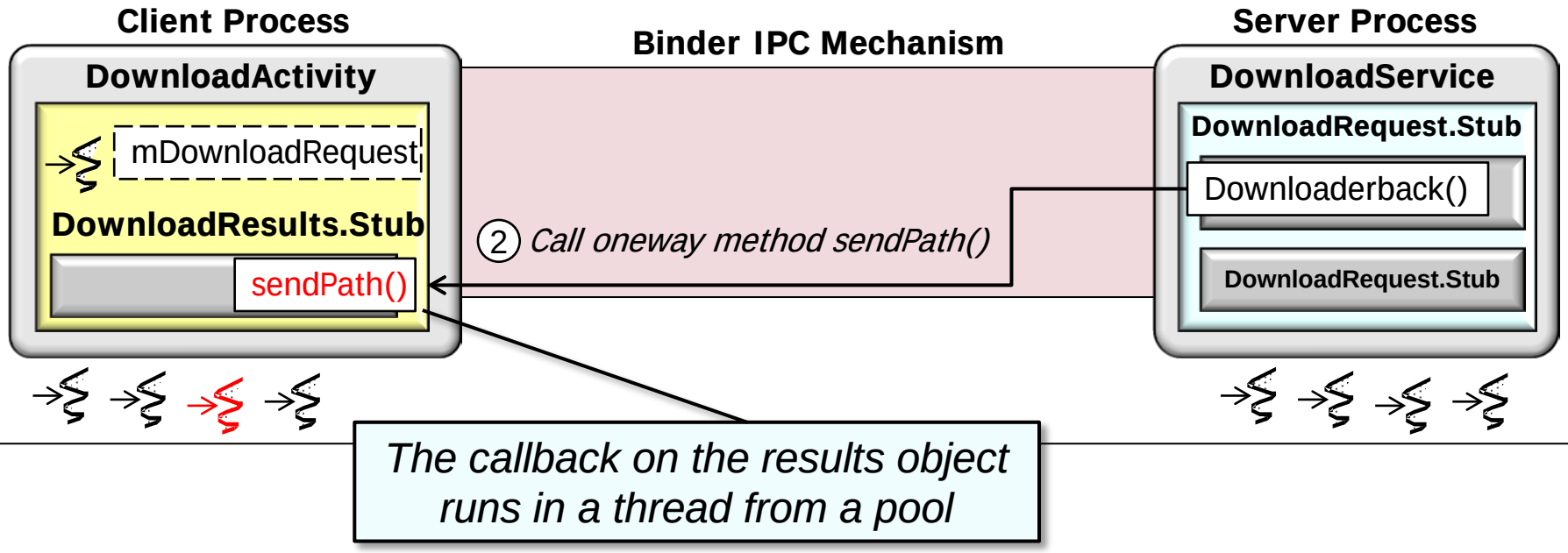
Overview of Android Binder & AIDL

- The Binder Driver is installed in the Linux kernel to accelerate IPC
- Android (system) Services can be written in Java, as well as C/C++
- Caller's data is marshaled into parcels, copied to callee's process, & demarshaled into what callee expects
- Two-way method invocations are synchronous (block the caller)
- Android also supports two-way asynchronous calls between processes
  - Implemented using two AIDL interfaces with one-way methods



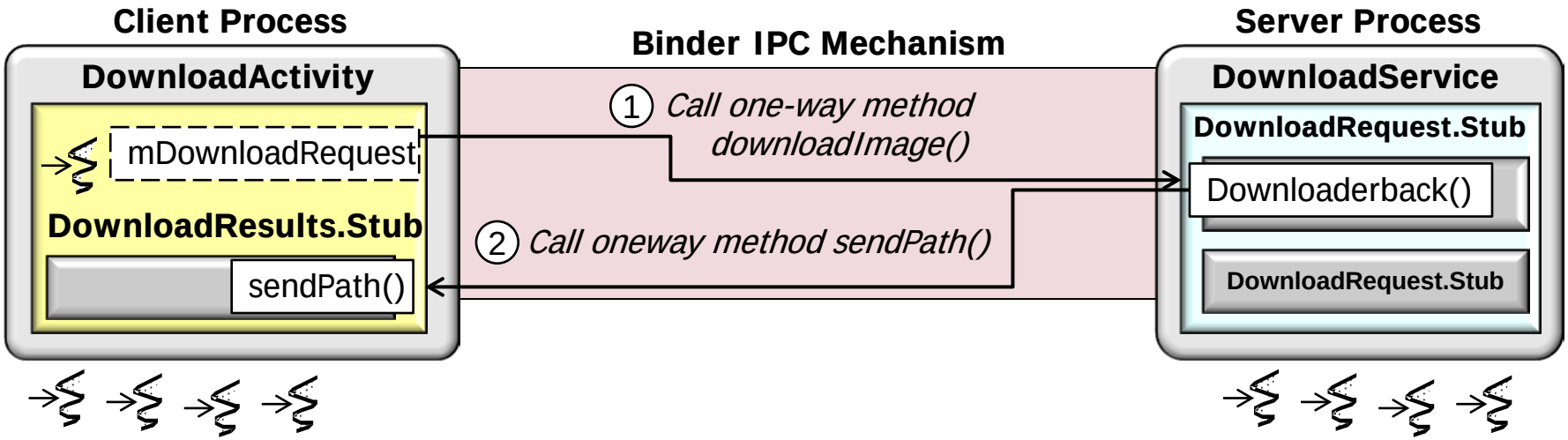
Overview of Android Binder & AIDL

- The Binder Driver is installed in the Linux kernel to accelerate IPC
- Android (system) Services can be written in Java, as well as C/C++
- Caller's data is marshaled into parcels, copied to callee's process, & demarshaled into what callee expects
- Two-way method invocations are synchronous (block the caller)
- Android also supports two-way asynchronous calls between processes
  - Implemented using two AIDL interfaces with one-way methods



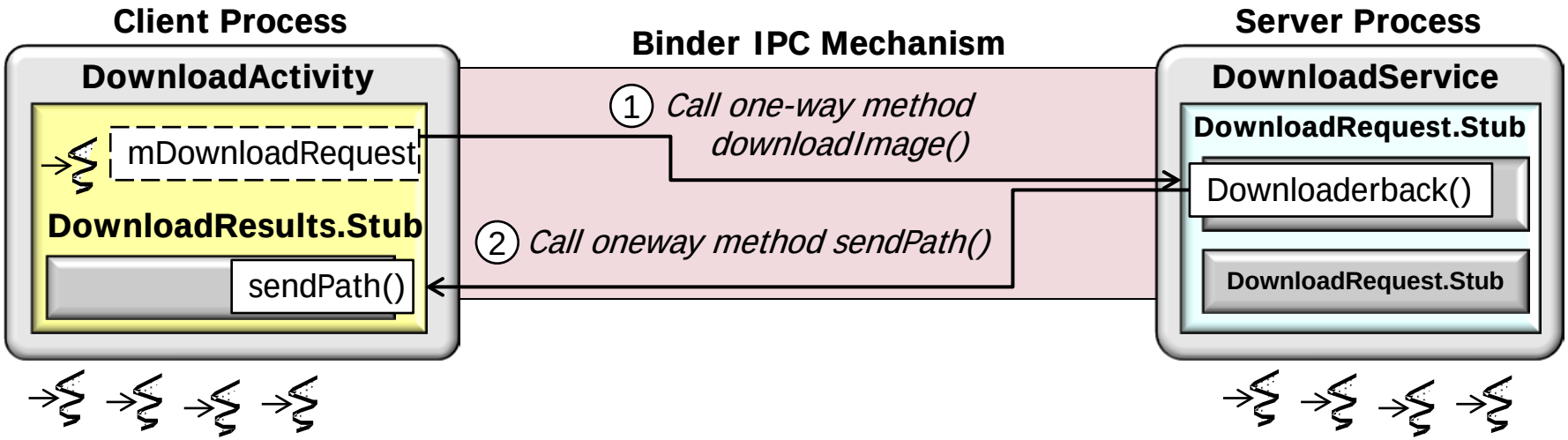
Overview of Android Binder & AIDL

- The Binder Driver is installed in the Linux kernel to accelerate IPC
- Android (system) Services can be written in Java, as well as C/C++
- Caller's data is marshaled into parcels, copied to callee's process, & demarshaled into what callee expects
- Two-way method invocations are synchronous (block the caller)
- Android also supports two-way asynchronous calls between processes
- AIDL invocations that span process boundaries run in a thread pool



Overview of Android Binder & AIDL

- The Binder Driver is installed in the Linux kernel to accelerate IPC
- Android (system) Services can be written in Java, as well as C/C++
- Caller's data is marshaled into parcels, copied to callee's process, & demarshaled into what callee expects
- Two-way method invocations are synchronous (block the caller)
- Android also supports two-way asynchronous calls between processes
- AIDL invocations that span process boundaries run in a thread pool
  - Objects must therefore be synchronized to avoid race conditions



Summary

Summary

- AIDL is a language for defining Binder-based interfaces to Bound Services

Android Interface Definition Language (AIDL)

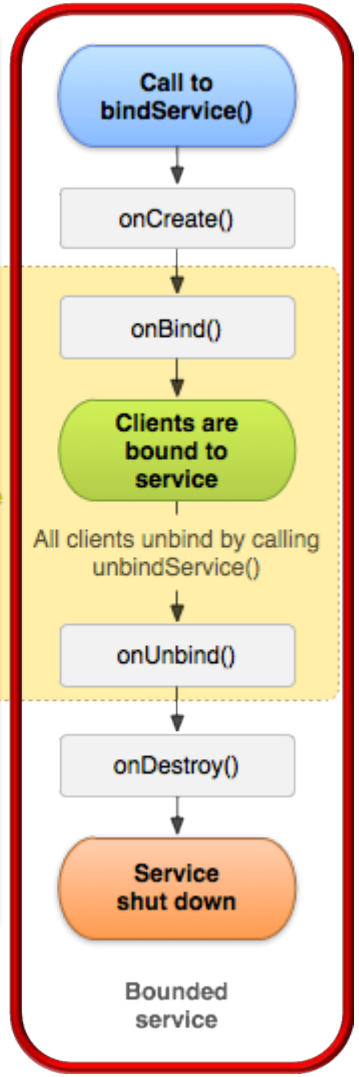
AIDL (Android Interface Definition Language) is similar to other IDLs you might have worked with. It allows you to define the programming interface that both the client and service agree upon in order to communicate with each other using interprocess communication (IPC). On Android, one process cannot normally access the memory of another process. So to talk, they need to decompose their objects into primitives that the operating system can understand, and marshall the objects across that boundary for you. The code to do that marshalling is tedious to write, so Android handles it for you with AIDL.

IN THIS DOCUMENT

SEE ALSO

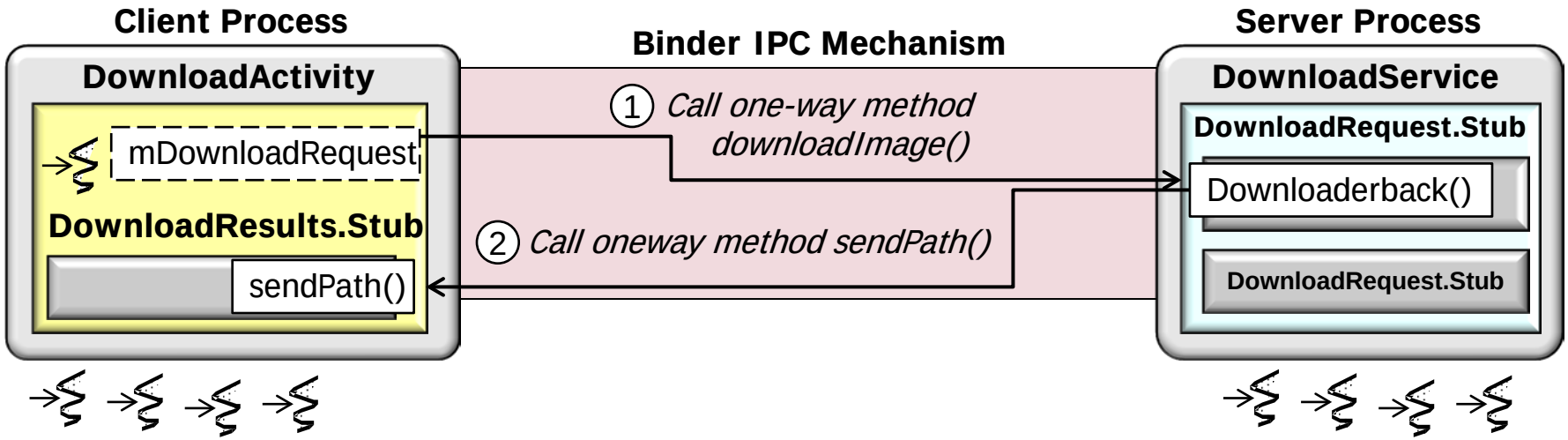
Bound Services

> **Note:** Using AIDL is necessary only if you allow clients from different applications to access your service for IPC and want to handle multithreading in your service. If you do not need to perform concurrent IPC across different applications, you should create your interface by implementing a Binder or, if you want to perform IPC, but do *not* need to handle multithreading, implement your interface using a Messenger. Regardless, be sure that you understand Bound Services before implementing an AIDL.



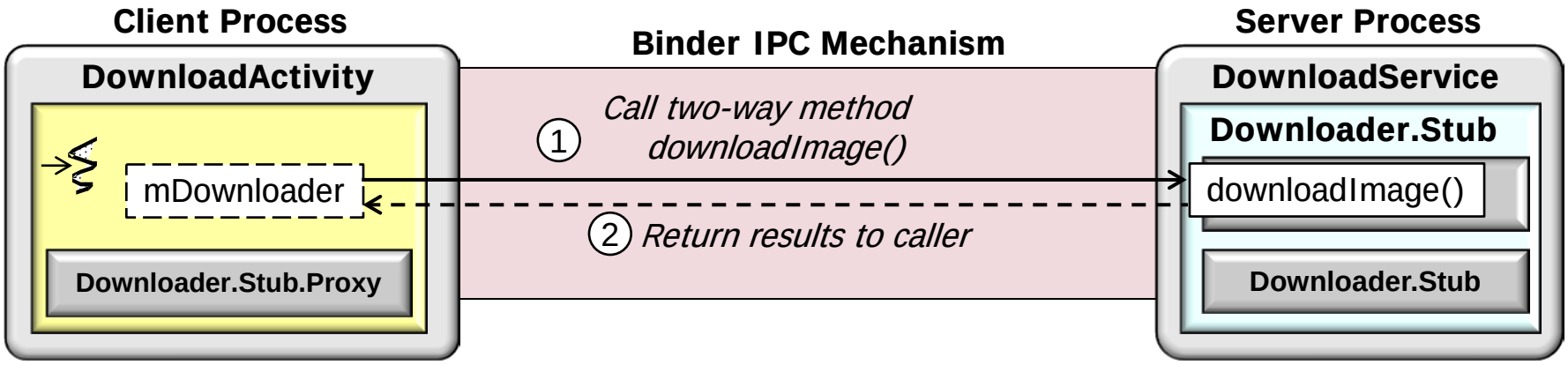
Summary

- AIDL is a language for defining Binder-based interfaces to Bound Services
- The AIDL & Binder framework provide several capabilities to applications that communicate with Bound Services residing in separate processes



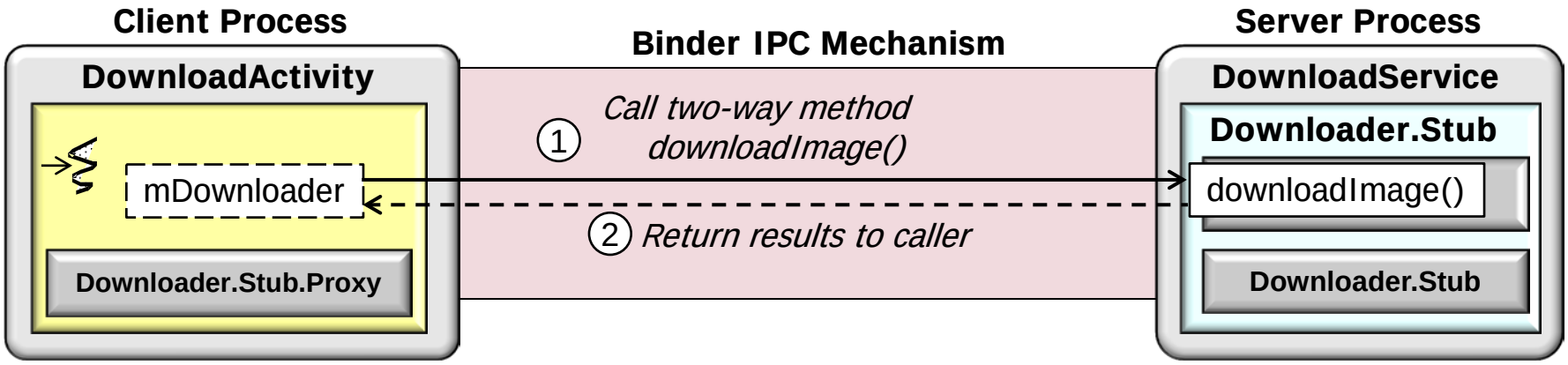
Summary

- AIDL is a language for defining Binder-based interfaces to Bound Services
- The AIDL & Binder framework provide several capabilities to applications that communicate with Bound Services residing in separate processes
  - The provide an object-oriented IPC mechanism



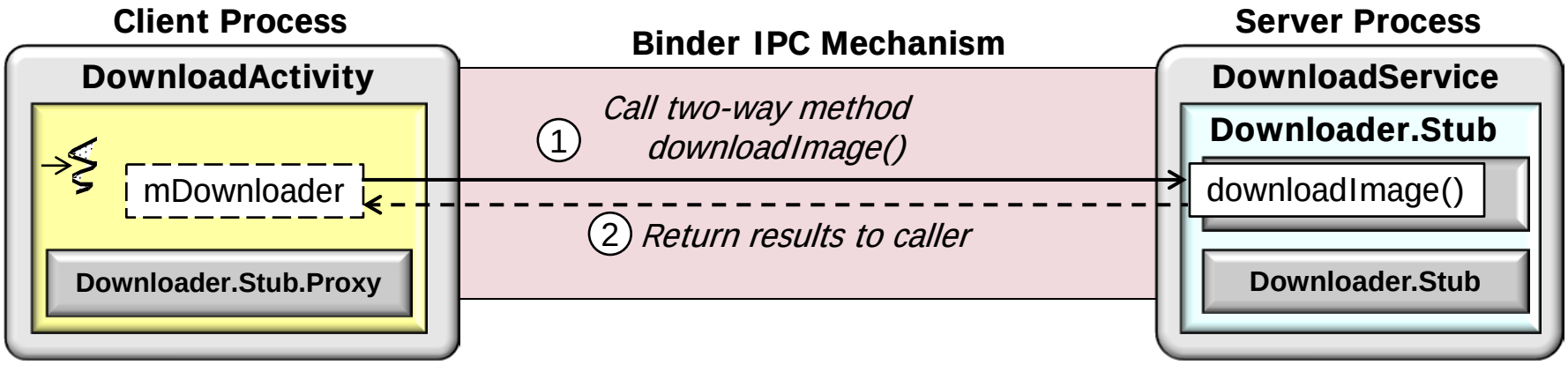
Summary

- AIDL is a language for defining Binder-based interfaces to Bound Services
- The AIDL & Binder framework provide several capabilities to applications that communicate with Bound Services residing in separate processes
  - The provide an object-oriented IPC mechanism
    - Mimics typed method invocations between Java objects



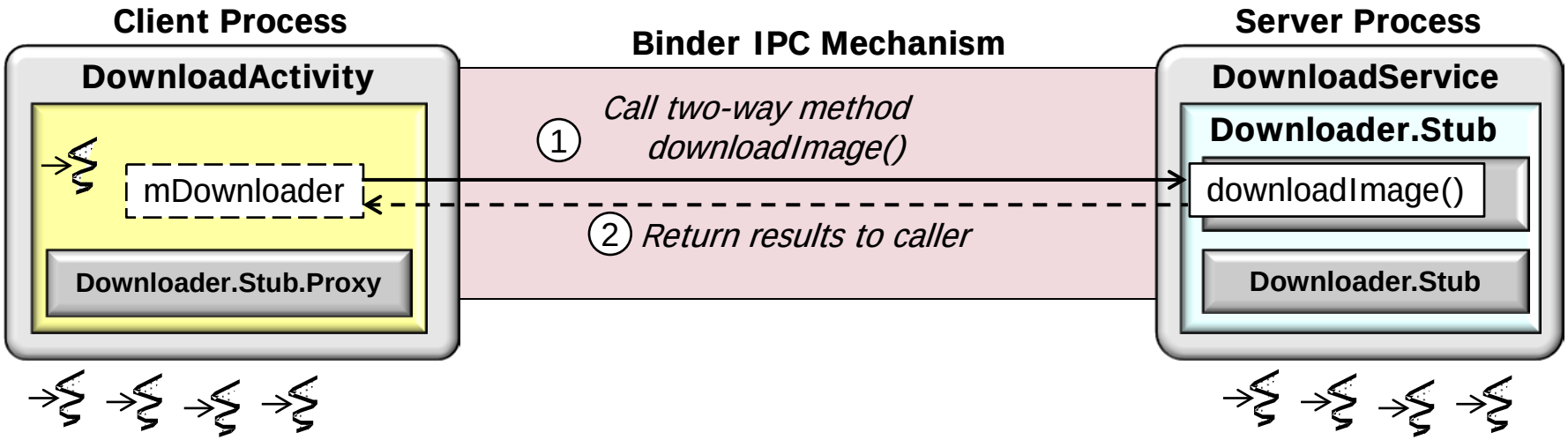
Summary

- AIDL is a language for defining Binder-based interfaces to Bound Services
- The AIDL & Binder framework provide several capabilities to applications that communicate with Bound Services residing in separate processes
  - The provide an object-oriented IPC mechanism
    - Mimics typed method invocations between Java objects
    - Explicitly typed interfaces can be less tedious & error-prone than the Messages sent via the generic Messenger mechanism



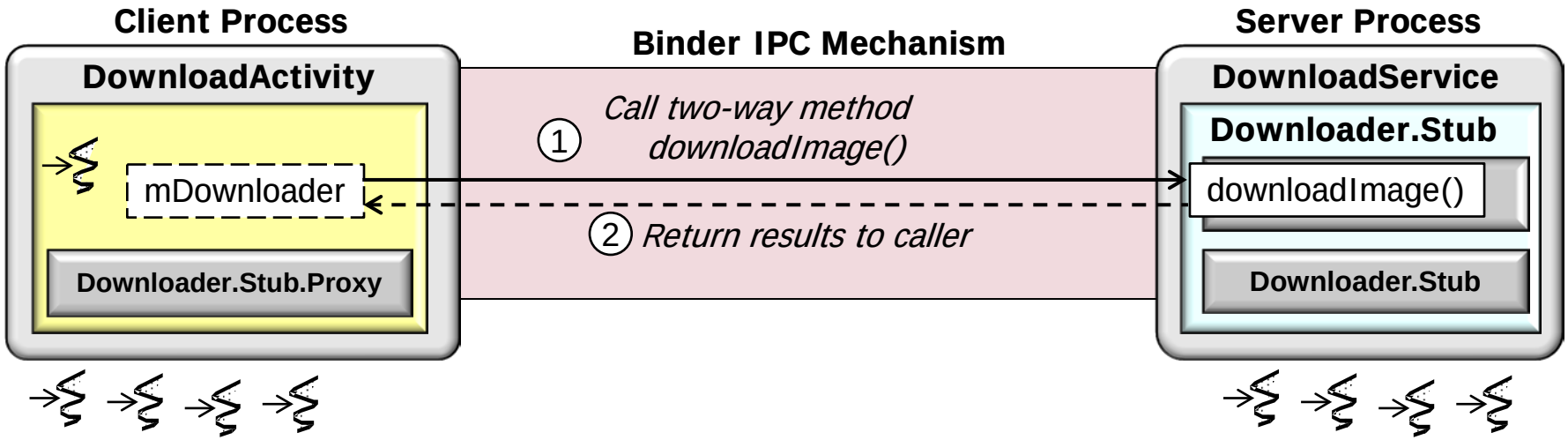
Summary

- AIDL is a language for defining Binder-based interfaces to Bound Services
- The AIDL & Binder framework provide several capabilities to applications that communicate with Bound Services residing in separate processes
  - The provide an object-oriented IPC mechanism
  - Method calls spanning process boundaries run concurrently in Thread pools



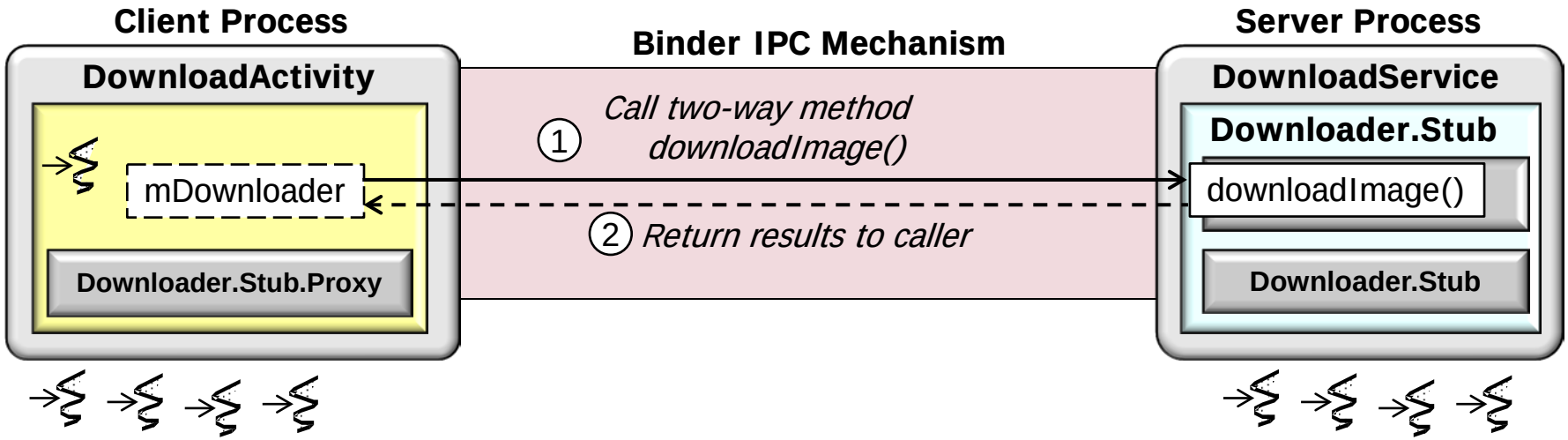
Summary

- AIDL is a language for defining Binder-based interfaces to Bound Services
- The AIDL & Binder framework provide several capabilities to applications that communicate with Bound Services residing in separate processes
  - The provide an object-oriented IPC mechanism
  - Method calls spanning process boundaries run concurrently in Thread pools
    - These Threads are managed by the Binder framework



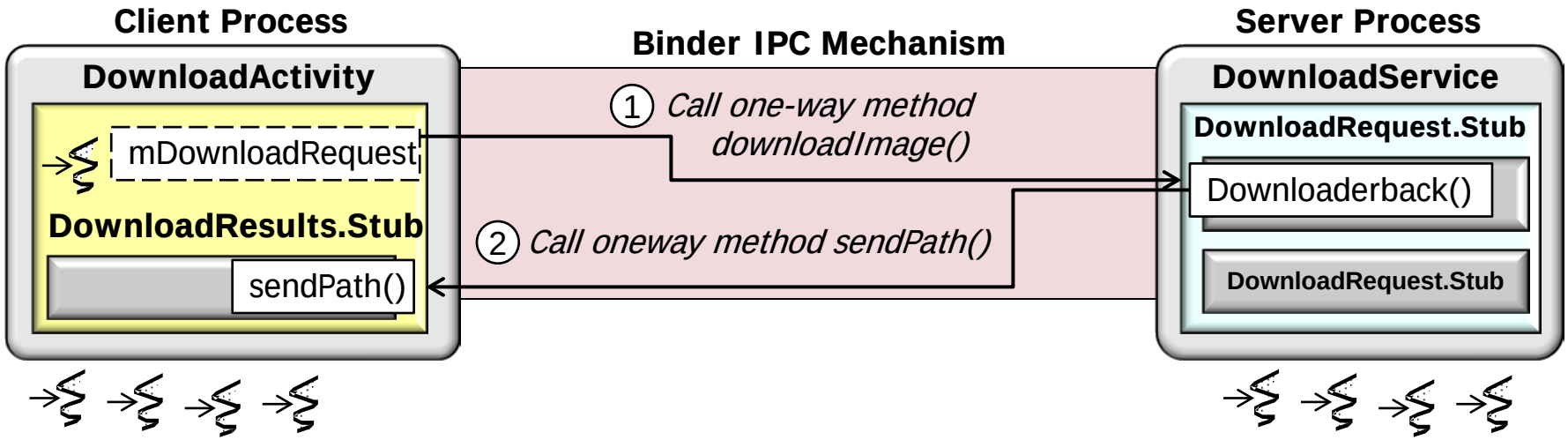
Summary

- AIDL is a language for defining Binder-based interfaces to Bound Services
- The AIDL & Binder framework provide several capabilities to applications that communicate with Bound Services residing in separate processes
  - The provide an object-oriented IPC mechanism
  - Method calls spanning process boundaries run concurrently in Thread pools
    - These Threads are managed by the Binder framework
    - Application developers needn't explicitly manipulate Threads, Handlers, Runnables, or Messages



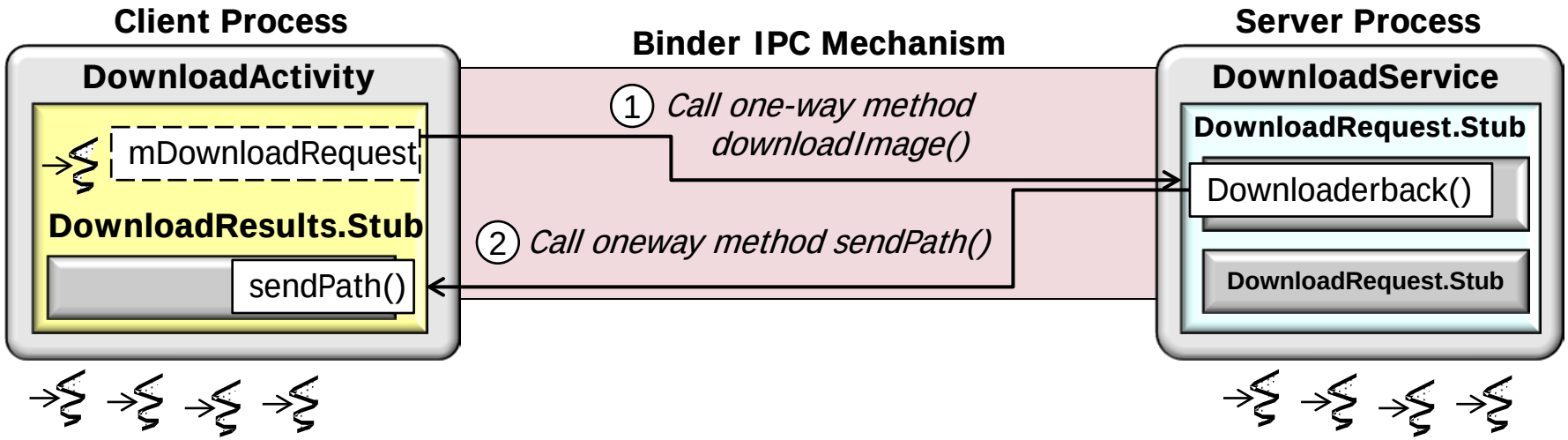
Summary

- AIDL is a language for defining Binder-based interfaces to Bound Services
- The AIDL & Binder framework provide several capabilities to applications that communicate with Bound Services residing in separate processes
  - The provide an object-oriented IPC mechanism
  - Method calls spanning process boundaries run concurrently in Thread pools
  - Activities/Services can communicate via two-way asynchronous interactions



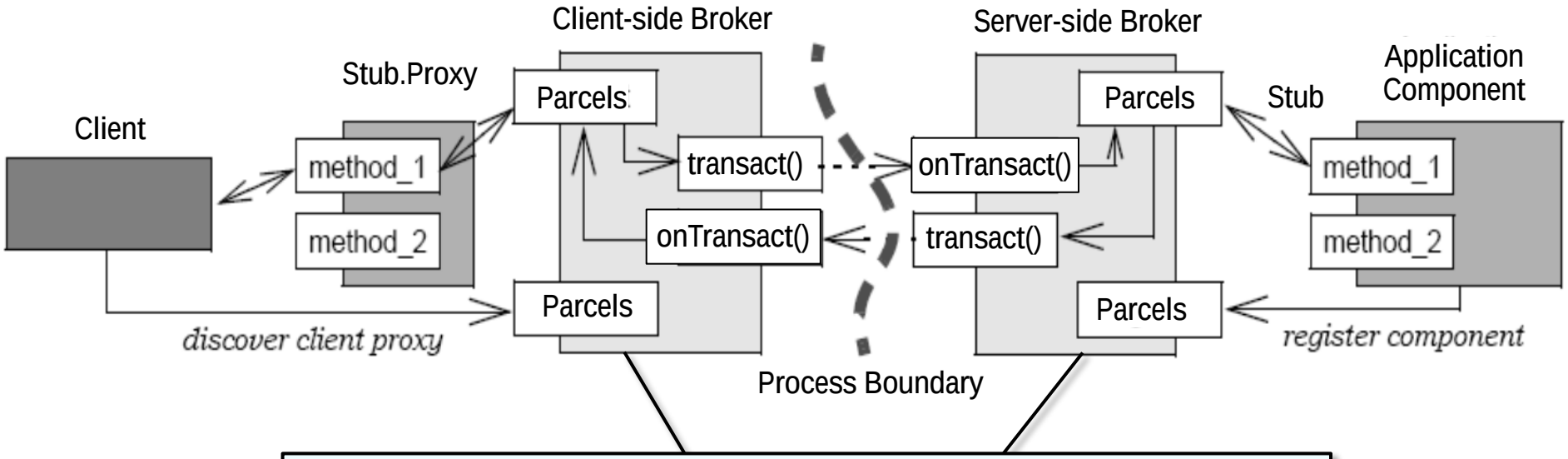
Summary

- AIDL is a language for defining Binder-based interfaces to Bound Services
- The AIDL & Binder framework provide several capabilities to applications that communicate with Bound Services residing in separate processes
  - The provide an object-oriented IPC mechanism
  - Method calls spanning process boundaries run concurrently in Thread pools
  - Activities/Services can communicate via two-way asynchronous interactions
    - Asynchronous programs can be robust & scalable if developers understand key patterns



Summary

- AIDL is a language for defining Binder-based interfaces to Bound Services
- The AIDL & Binder framework provide several capabilities to applications that communicate with Bound Services residing in separate processes
- The AIDL & Binder implementations apply the *Broker* pattern

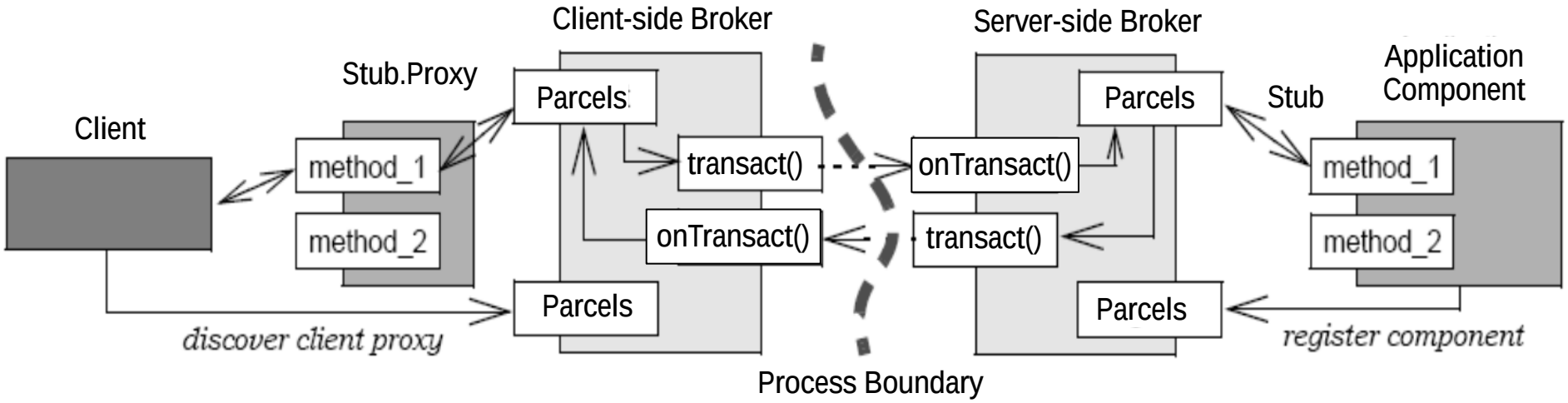


Broker connects clients with objects residing in Bound Services in different address spaces & mediates method invocations from clients to these objects

www.kircher-schwanninger.de/michael/publications/BrokerRevisited.pdf

Summary

- AIDL is a language for defining Binder-based interfaces to Bound Services
- The AIDL & Binder framework provide several capabilities to applications that communicate with Bound Services residing in separate processes
- The AIDL & Binder implementations apply the *Broker* pattern



- Many other patterns are used to implement AIDL & Binder framework
  - e.g., *Proxy*, *Activator*, etc.

See upcoming section on “Concurrency & Communication Patterns in Android”

Summary

- AIDL is a language for defining Binder-based interfaces to Bound Services
- The AIDL & Binder framework provide several capabilities to applications that communicate with Bound Services residing in separate processes
- The AIDL & Binder implementations apply the *Broker* pattern
- The AIDL & Binder are used extensively throughout Android's frameworks & packaged applications

  frameworks/base/core/java/android/app/IAlarmManager.aidl
  frameworks/base/core/java/android/app/INotificationManager.aidl
  frameworks/base/core/java/android/app/IProcessObserver.aidl
  frameworks/base/core/java/android/app/ISearchManager.aidl
  frameworks/base/core/java/android/app/IServiceConnection.aidl
  frameworks/base/core/java/android/app/PendingIntent.aidl
  frameworks/base/core/java/android/content/pm/ActivityInfo.aidl
  frameworks/base/core/java/android/database/IContentObserver.aidl
  frameworks/base/core/java/android/net/NetworkInfo.aidl
  frameworks/base/core/java/android/os/Bundle.aidl
  frameworks/base/core/java/android/os/IMessenger.aidl
  frameworks/base/core/java/android/os/Messenger.aidl
  frameworks/base/core/java/android/os/ParcelFileDescriptor.aidl
  frameworks/base/core/java/android/view/WindowManager.aidl
  frameworks/base/core/java/android/widget/RemoteViews.aidl