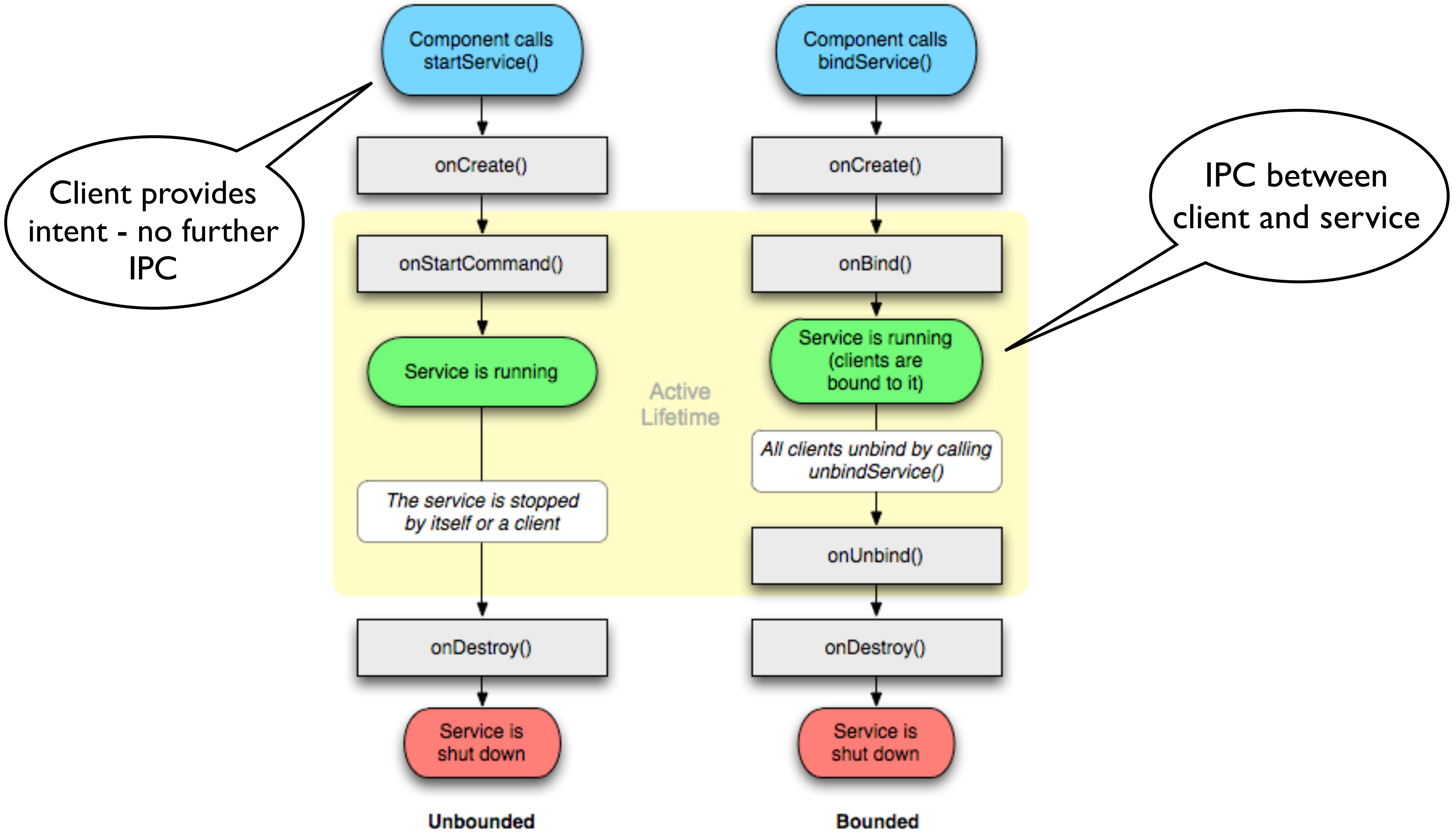
Service

- Run long-running operations with no UI
    - Notifications and toasts, though, are possible (but from main thread only)
- Service lifecycle can be independent of any activity
- Two ways to run service:
    - Unbounded: explicit `startService()` and `stopSelf()` or `stopService()`
    - Bounded: started with first `bind()`, ended with last `unbind()` - IPC with a defined interface between service and components that use it
- start/stop, bind/unbind can be combined
- Note: service by default runs in app main thread - threading is needed to keep UI responsive and avoid ANR

Service lifecycle


Client provides intent - no further IPC
Component calls startService()
onCreate()
onStartCommand()
Service is running
The service is stopped by itself or a client
onDestroy()
Service is shut down
Unbounded
Active Lifetime
Component calls bindService()
onCreate()
onBind()
Service is running (clients are bound to it)
All clients unbind by calling unbindService()
onUnbind()
onDestroy()
Service is shut down
Bounded
IPC between client and service

Implementing started service - the flexible way

- Inherit `Service` and implement
  - `onCreate()`
  - `onStartCommand()` - `Intent` object is available here, use for example extras to get input values for the service
  - `onBind()` - return `nil` if the service does not support bound operation
- Note:
  - service runs in main thread, not good!
  - in practice you must stop service yourself (`stopSelf()`)

Implementing started service - an easy way

- Inherit `IntentService` and implement
  - `onHandleIntent()`
- `IntentService` provides implementation for
  - processing each intent in a worker thread (not the application main thread) - but one at a time
  - setting up a queue for `Intent` objects received when clients call `startService()`
  - stopping the service when it is no more needed

Broadcast Receiver

- Broadcast receiver is a component that does nothing but receive and react to broadcast announcements.
- Broadcast receivers do not display a user interface. However, they may start an activity in response to the information they receive.
- Many broadcasts originate in system code (the timezone has changed, the battery is low or a picture has been taken).
- Applications (esp. Service components) can initiate broadcasts - for example to let other applications know that some data has been downloaded to the device and is available for them to use.
- An application can have any number of broadcast receivers to respond to any announcements it considers important. All receivers extend the `BroadcastReceiver` base class.