Extra Groovy Stuff about Processes in Android

Some Guy (who may or may not be Brian Ricks)

An excellent question was asked in class today over processes for inter-app components: more specifically does a component (for example an Activity) invoked via intent from a component in another app share the same process? To answer this question, let's take a look at Android's process organization.

By default, every component within a single application will run in the same process (yes, including services – there is nothing special about services). To have a component run in a different process under the same application, a new process name using *android:process* must be defined for that component in the application's manifest. If one takes a look at the list of processes running in Android at any given time, they will notice that process names are usually package names, corresponding to different applications. This is the default behavior. Essentially, each *application,* by default, is assigned its own process. Note that, as mentioned in class, an application is in essence a collection of components under the same package. So organizing processes in this way makes sense. Groovy, but can we have components from different apps run in the same process?

Yes, by doing a couple of things. First, both apps need to be signed with the same certificate. Second, both components need to have the same value for *android:process* in their respective manifests.

So that's that, but it really doesn't tell the whole story. Android's intent system works under the Inter-process Communication (IPC) paradigm. This makes sense when looking at the camera activity example from class: App A's activity needs a pic taken with the camera but doesn't include a component to do this, so it creates and sends an implicit intent, Android OS matches the intent criteria with some camera activity in another app, and that activity is launched. The user snaps a pic, and then the pic is sent back to App A's calling activity. This is all done using IPC. Great, but what about two activities within the same app (and process)? We start these activities using intents also, but this isn't IPC. Opps, in the Android world, these details are abstracted away from you (this is their 'intent'.... ba dum tss), but behind the scenes what goes on is very different. Remember, processes are containers with their own virtual address space (whereas threads are execution units that share the virtual address space of their parent process – it is implied that any process will contain at least one thread). Within a single app, components will most likely be sharing the same process, and even the same thread within it. This means that how the intent actually is used behind the scenes can be quite different.

Should you care? Maybe. Multiple processes introduce additional overhead. IPC itself also introduces overhead. Resource sharing between processes is not as trivial as between components within the same process. What if you want to share an object between Activity A in process A and Activity B in process B? Can you use the same techniques you would use if say both Activity A and B were in the same process? In the former case, IPC is needed, and you would most likely use intents containing serialized data. In the latter, you can use singleton objects (the Application object is a great place to look) or public static fields, both of which are more efficient than serializing objects. This distinction becomes especially important if you want to separate components of the same app into different processes, as since generally all components within an app share the same process, it may be easy to forget that IPC is now required.