Drawbacks of existing OS methods in smart phones:

* resource constrained smart phones -> high overhead for multiple complete OSs running on it.
* the hardwares are more directly related with different applications.

\*\*Cells\*\* solution: one OS instances, with virtual namespaces to run multiple virtual phones(VPs).

One foreground VP with full access to hardwares.

Multiple background VPs with shared access to hardwares.

configurations, download, creation of a VP is done in a PC. The access rights of a PC are configured during the PC, with no access, shared access and exclusive access.

Virtualization in Cells: private virtual namespaces.

* kernel-level device virtualization
  + device driver wrapper
  + modification to the event handler(device subsystem)
  + modification to the device driver(only allowing viewing of some certain processes)
* user-level device virtualization
  + one management environment
  + new VP

improve performance and ensure scalability

* share read-only file systems
* search for anonymous memory pages which have same contents to supply as a copy
* memory killer

## graphics

### framebuffer

framebuffer -> virtual framebuffer -> exclusive access to the foreground VP

Those background VPs send the output related to display to a virtual screen memory buffer in system RAM.

### GPU

VP can have direct access to GPU. But the final display is isolated because the VPs need FB to display its scene.

### power management

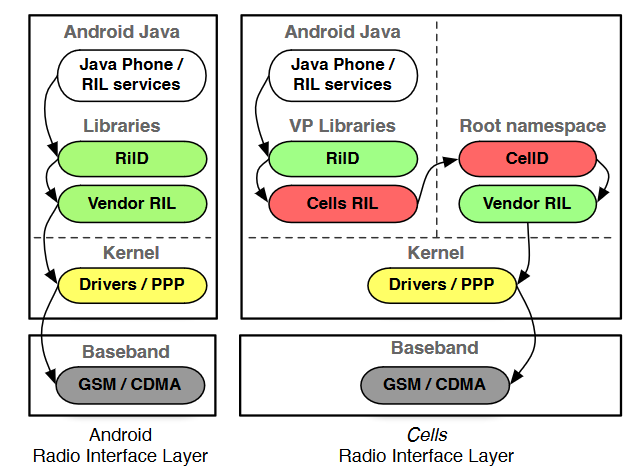
* background always not in low-power mode
* background not stopping the foreground in low-power mode

power management subsystems:

* early suspend: disallow the background VPs from initializing suspend
* fbearlysuspend: display device suspend and resume states into user space
* wake locks: when all the locks are inactive, the system goes into lowpower mode; all the foreground and background VPs can take the control of the same locks. While the low power mode is associated with the locks related to foreground VP.

## telephony

independent call logs and phone numbers for VPs



VPs have their own RiIDs, but communicate and utilize the hardware by connecting to the CellD.

input: the messages are multiplexed through cellD.

output: only foreground VP can make calls

Limit the phony commands into some categories that some of them can only be used by foreground VPs such as making a dial.

=> provide full configurations on the phony policies

multiple numbers: VoIP server, each VP has an appending digit to indicate.

## networking

just like a virtual machine in NAT

Each VP has a virtual Ethernet pair with the root namespace. Only the configuration requests sent from the foreground VP will be processes.