
aprilzosia

Multi-Persona Android

Oren Laadan
orenl@cellrox.com

Android Builders 2014

cellrox

www.cellrox.com

Mobile devices have multiple uses -
- the device needs to reflect that.

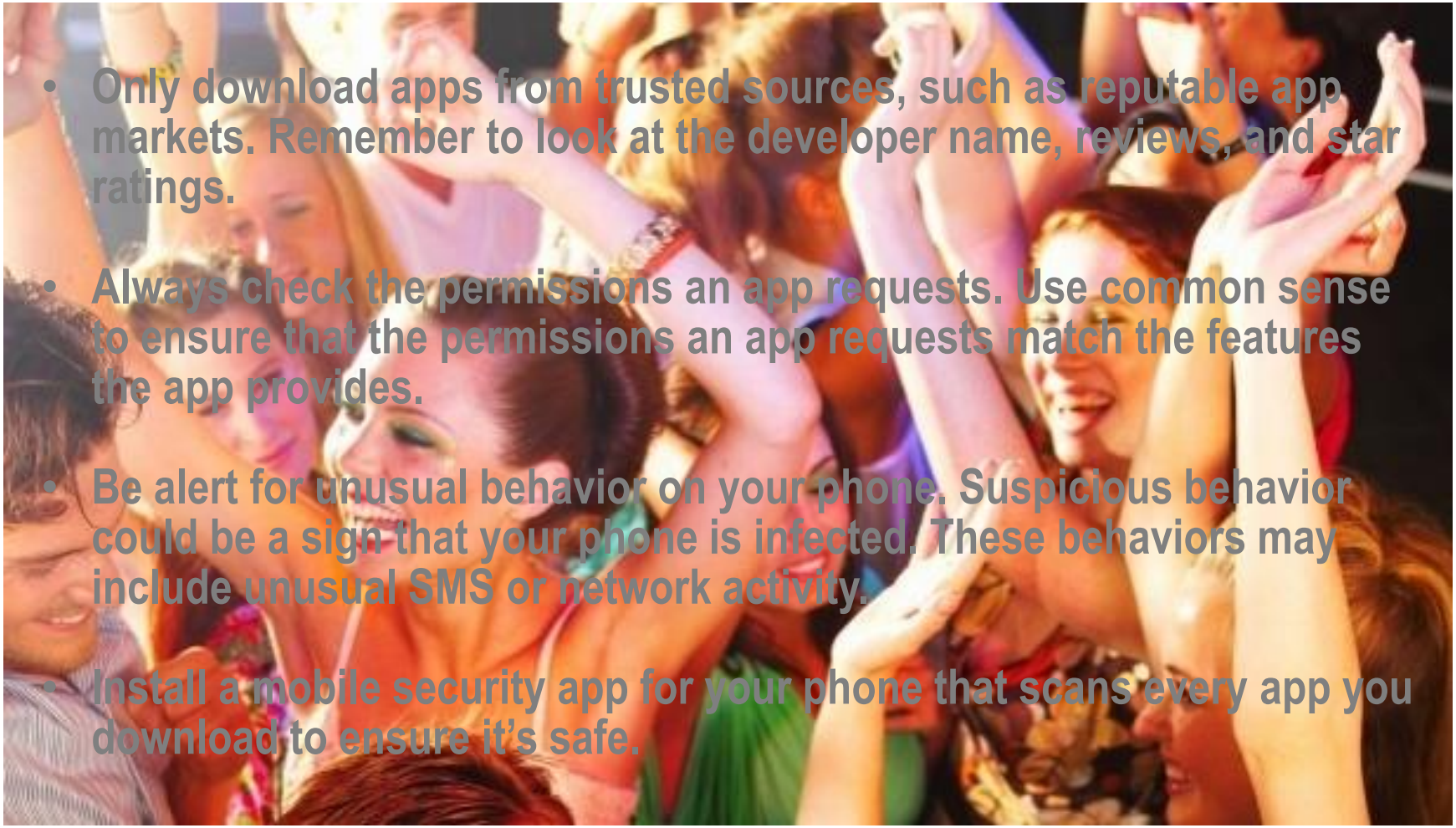
Security Use Case

Personal Phone
Business Phone

Do People Remember?

- Only download apps from trusted sources, such as reputable app markets. Remember to look at the developer name, reviews, and star ratings.
- Always check the permissions an app requests. Use common sense to ensure that the permissions an app requests match the features the app provides.
- Be alert for unusual behavior on your phone. Suspicious behavior could be a sign that your phone is infected. These behaviors may include unusual SMS or network activity.
- Install a mobile security app for your phone that scans every app you download to ensure it's safe.

No, They Don't!



- Only download apps from trusted sources, such as reputable app markets. Remember to look at the developer name, reviews, and star ratings.
- Always check the permissions an app requests. Use common sense to ensure that the permissions an app requests match the features the app provides.
- Be alert for unusual behavior on your phone. Suspicious behavior could be a sign that your phone is infected. These behaviors may include unusual SMS or network activity.
- Install a mobile security app for your phone that scans every app you download to ensure it's safe.

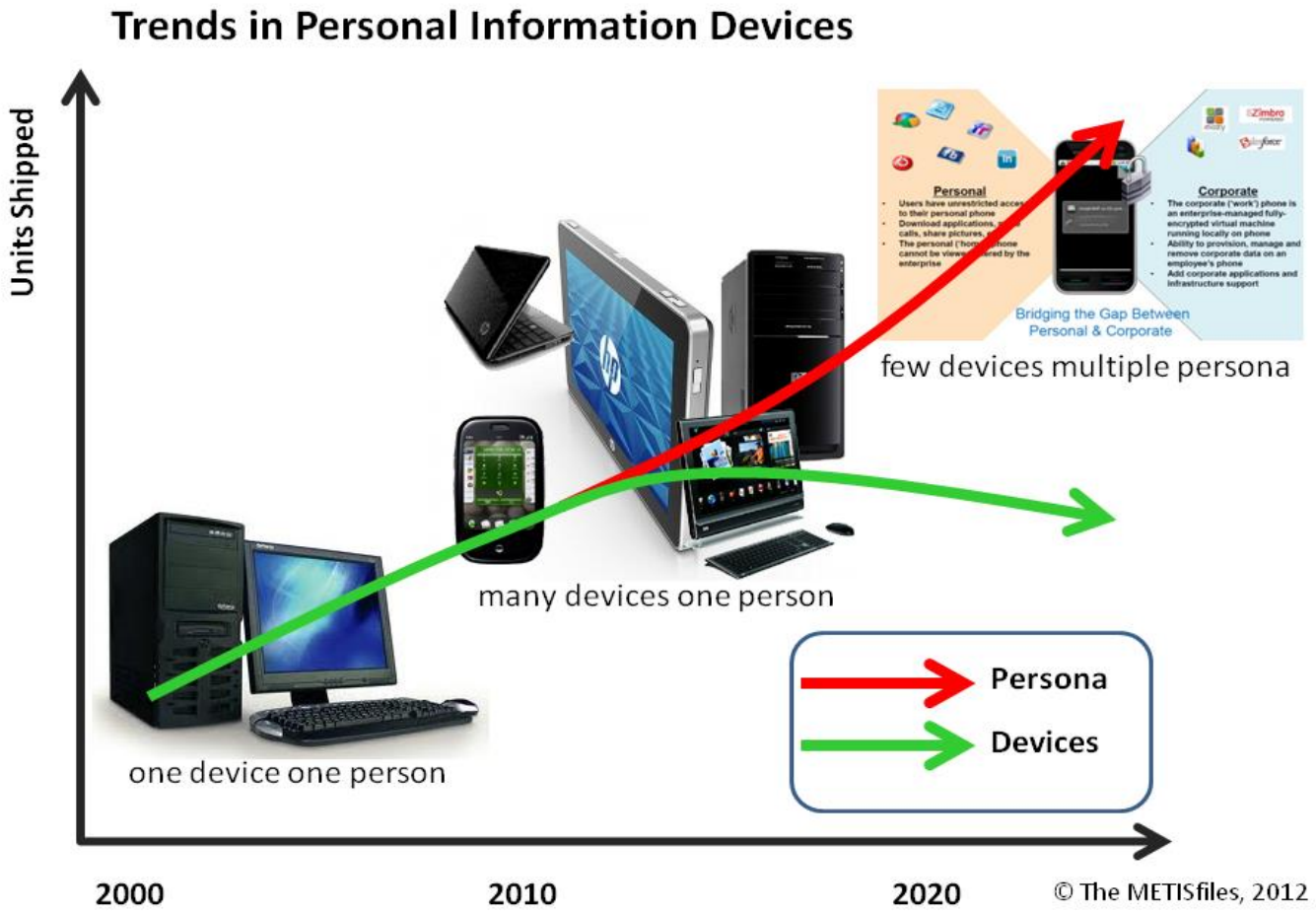
More Use Cases

Personal Phone
Business Phone
Children Phone
Privacy Phone
Secure Phone

Even More Use Cases

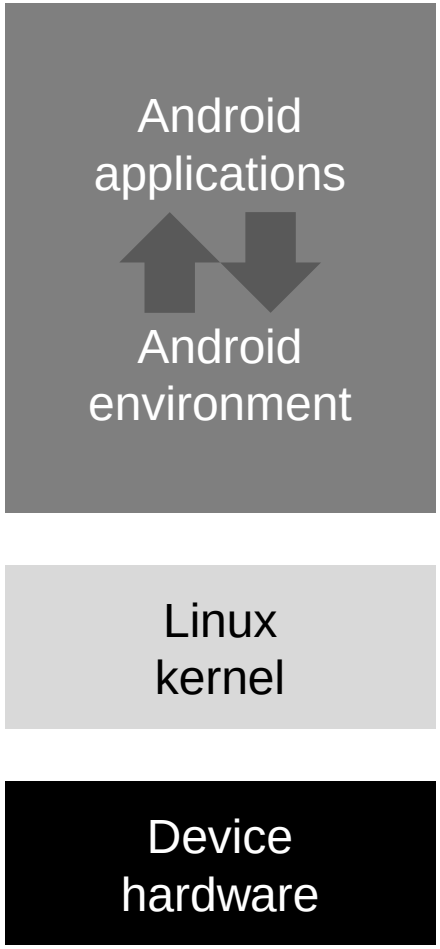
Personal Phone
Business Phone
Children Phone
Privacy Phone
Secure Phone
Social Phone
Guest Phone
Dev Phone

Multi-Persona for Mobile Devices



Mobile Device Virtualization

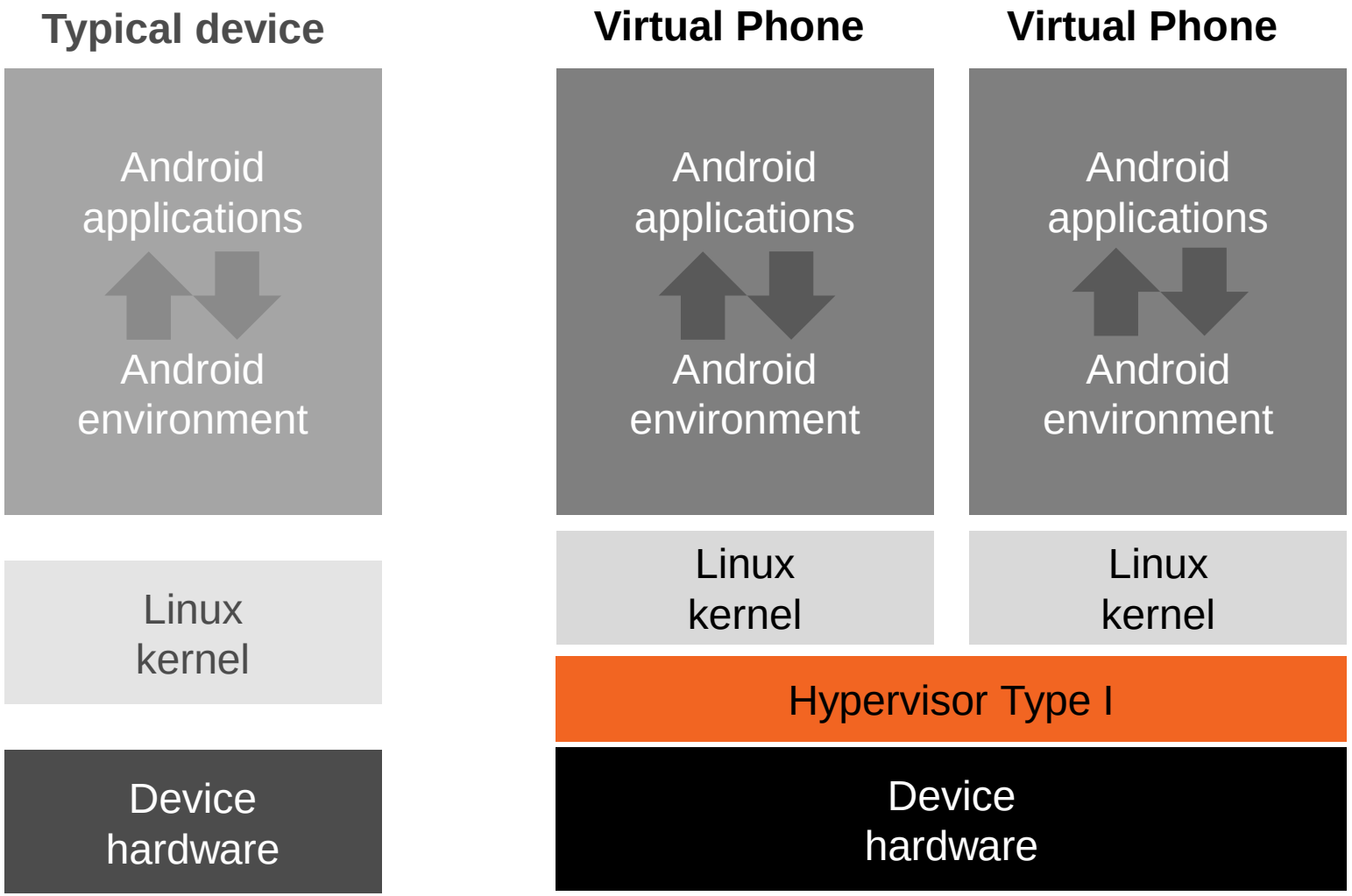
Typical device



Nobody Will Notice?

Performance Transparent
Application Transparent
Platform Transparent
User Transparent

Hardware Virtualization



Hardware Virtualization

Suitable for servers

- standard hardware
- slow server replace rate
- strong security model

Sub-optimal for mobile devices

- burden to support devices
- reduced performance / battery-life
- sub-optimal use of resources

Operating System Virtualization

Namespaces

provide a group of processes with the illusion that they are the only processes on the system.

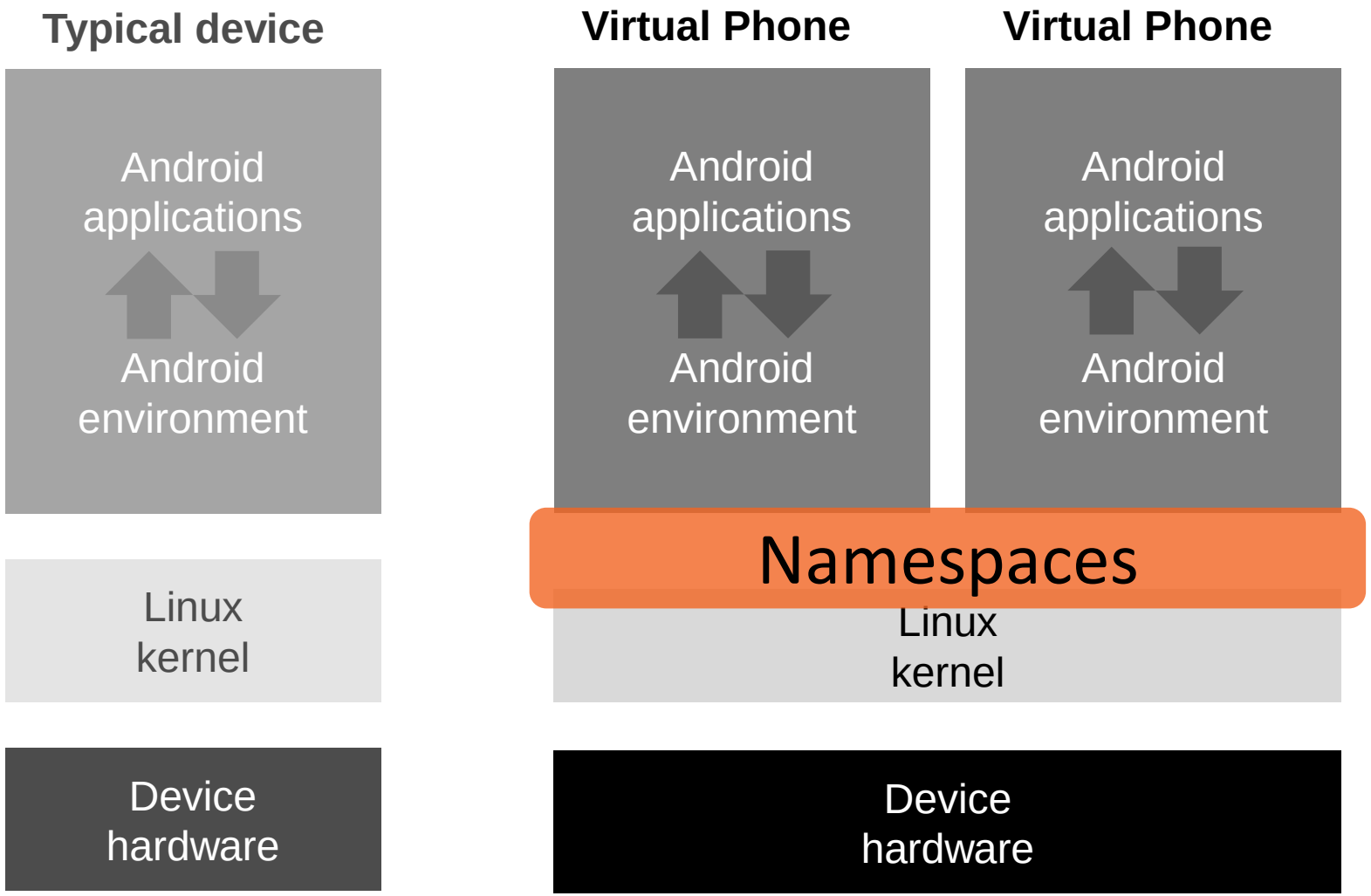
Namespace (r)evolution

Kernel namespaces:

- mount-ns: 2.4.19
- uts-ns: 2.6.19
- ipc-ns: 2.6.19
- pid-ns: 2.6.24
- net-ns: 2.6.24-2.6.29
- user-ns: 2.6.23-3.8

System calls: clone(), unshare(), setns()

Operating System Virtualization



Device Diversity

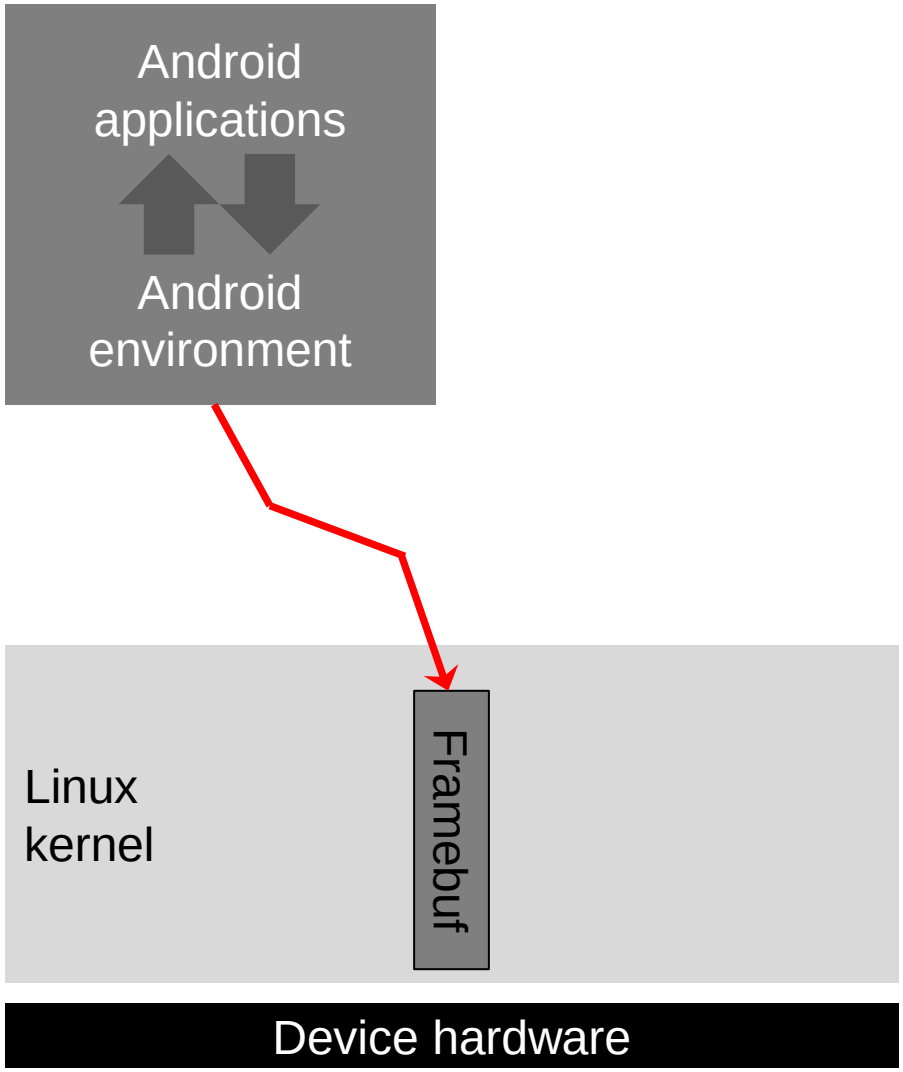
A typical collection of peripherals available on a modern smartphone or tablet:

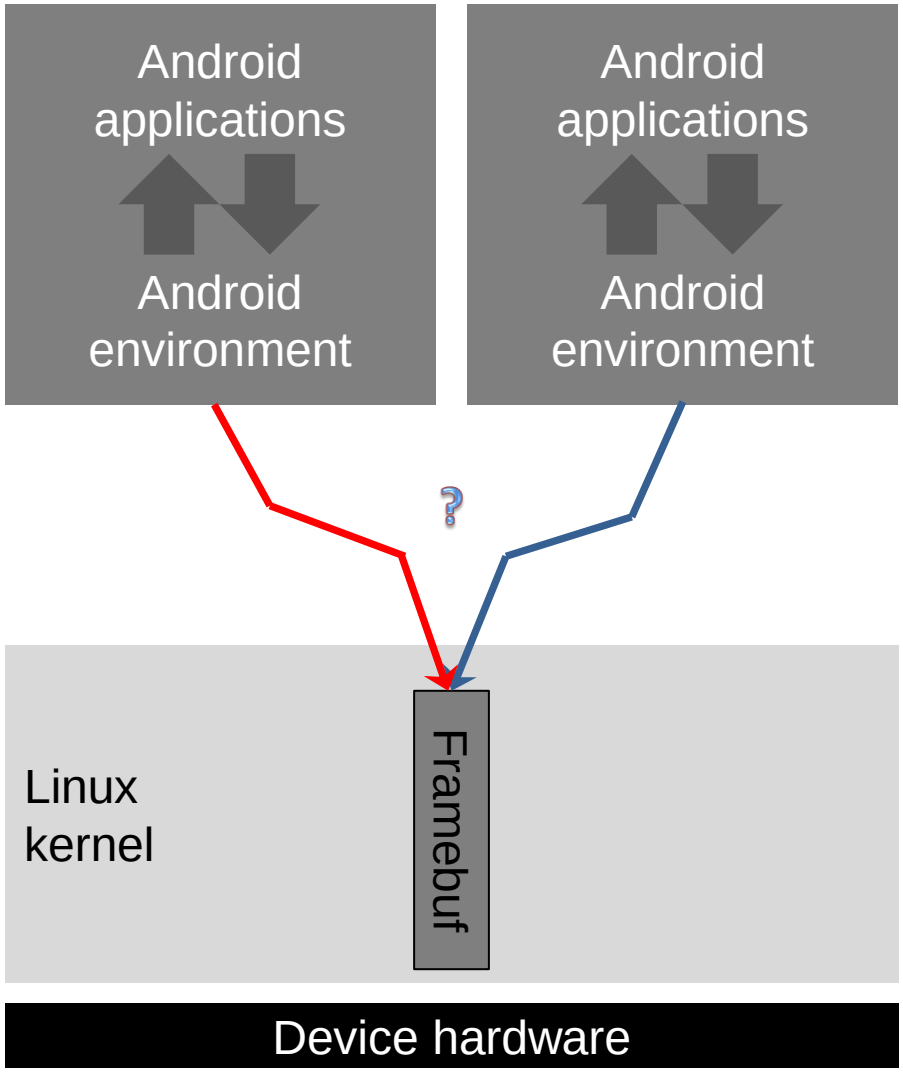
Headset	Microphone	Speakers	(Touch) Screen
Power	Buttons	Telephony	Bluetooth
GPS	WiFi	Framebuffer	GPU
Compass	Camera(s)	Accelerometer	RTC/Alarms

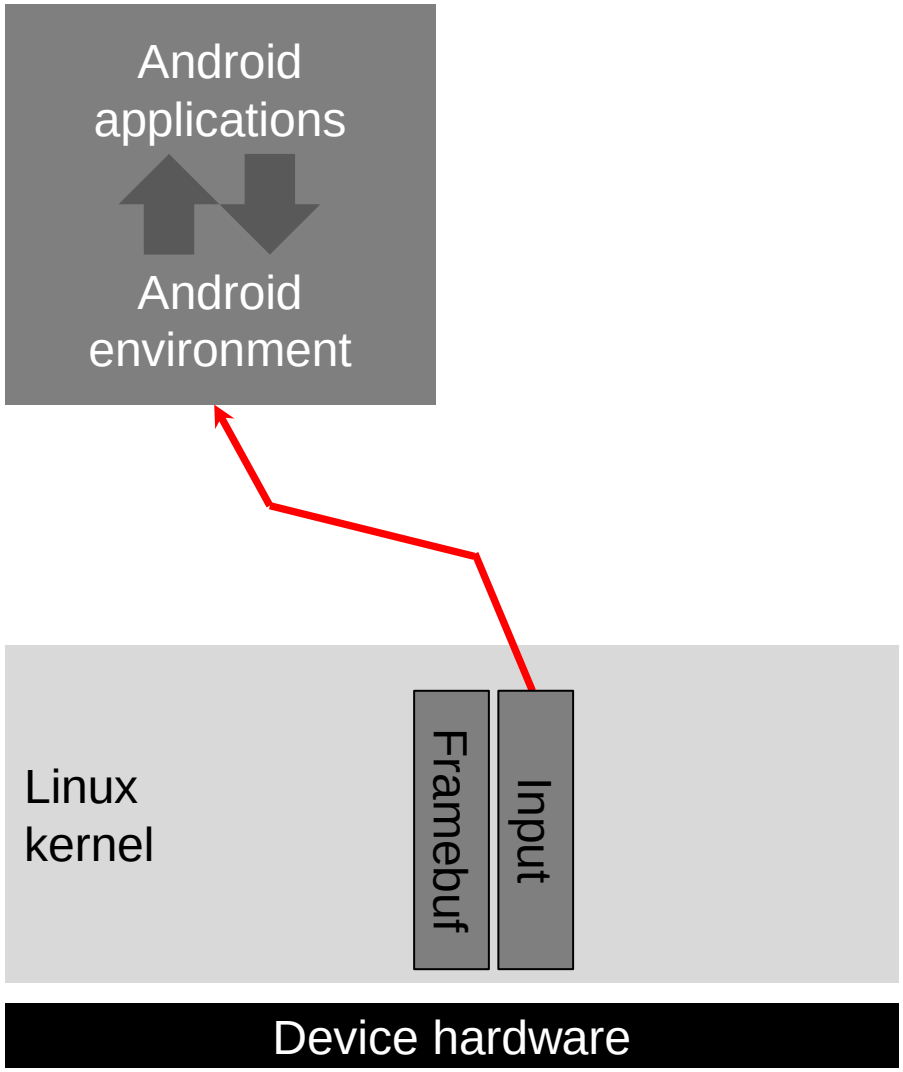
Device Interactivity

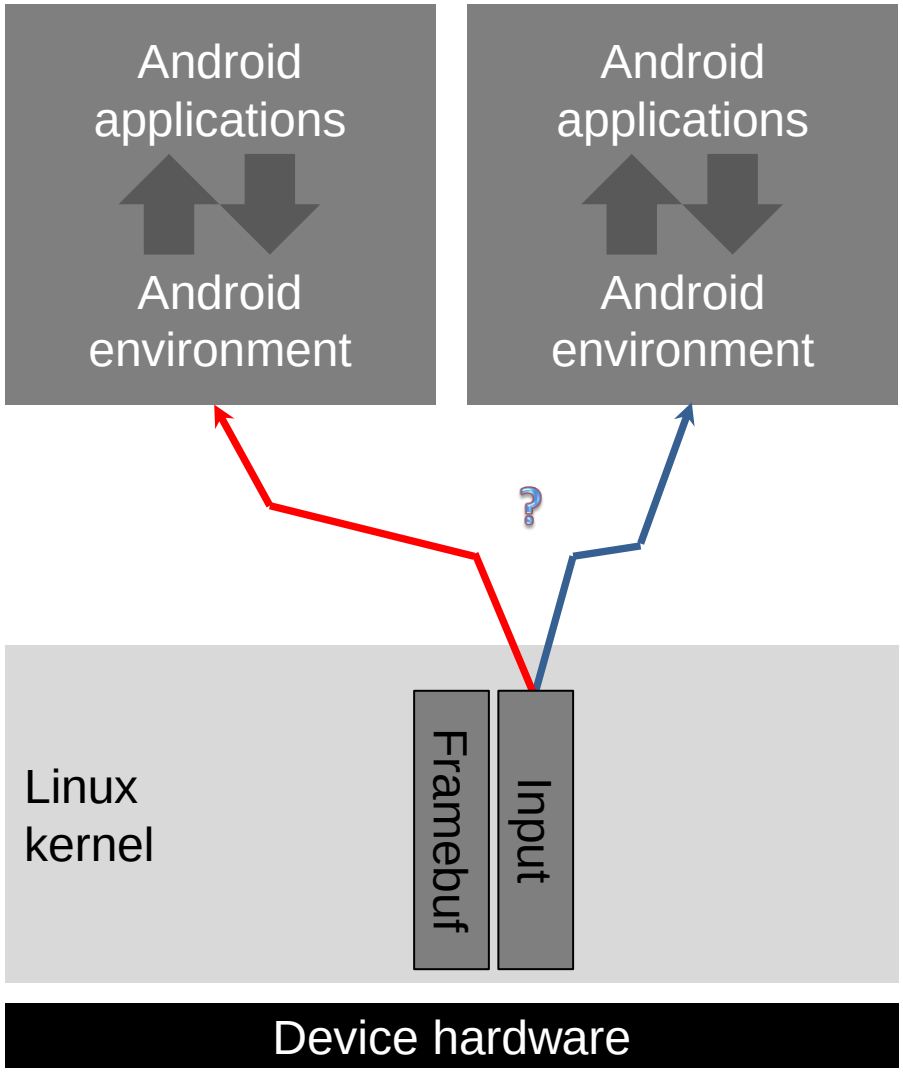
Users interact with a device one application at a time, expect consistent user experience:

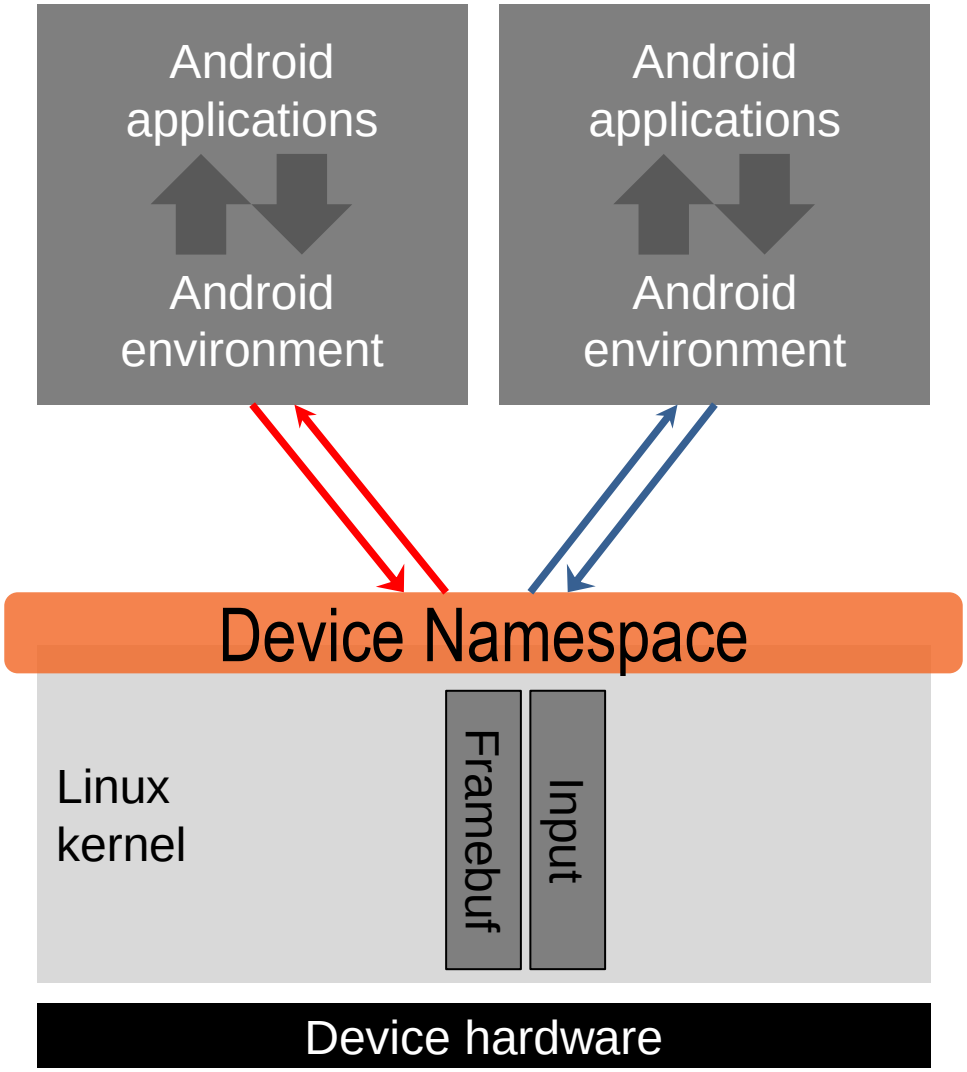
Split the “attention” of resources between the multiple persona, depending on context.

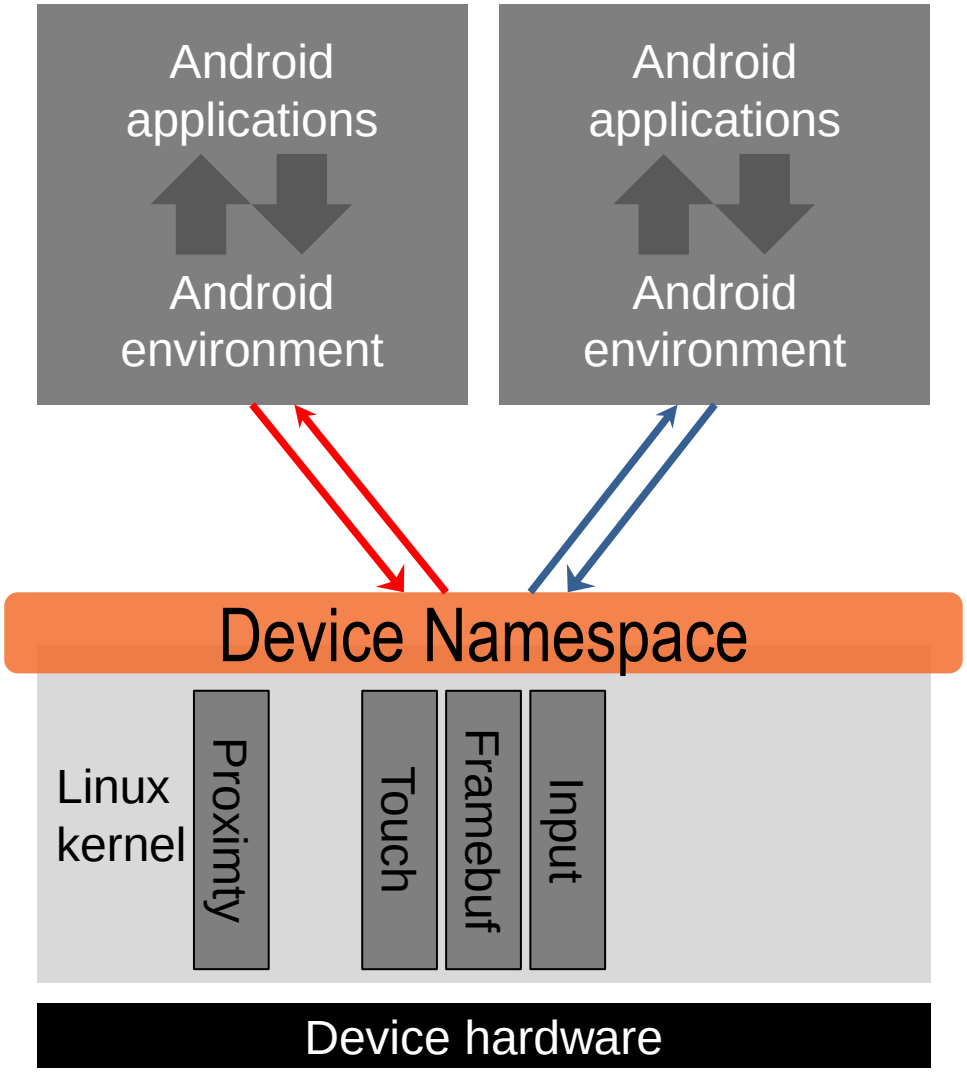

Android
applications
Android
environment
Linux
kernel
Framebuf
Device hardware

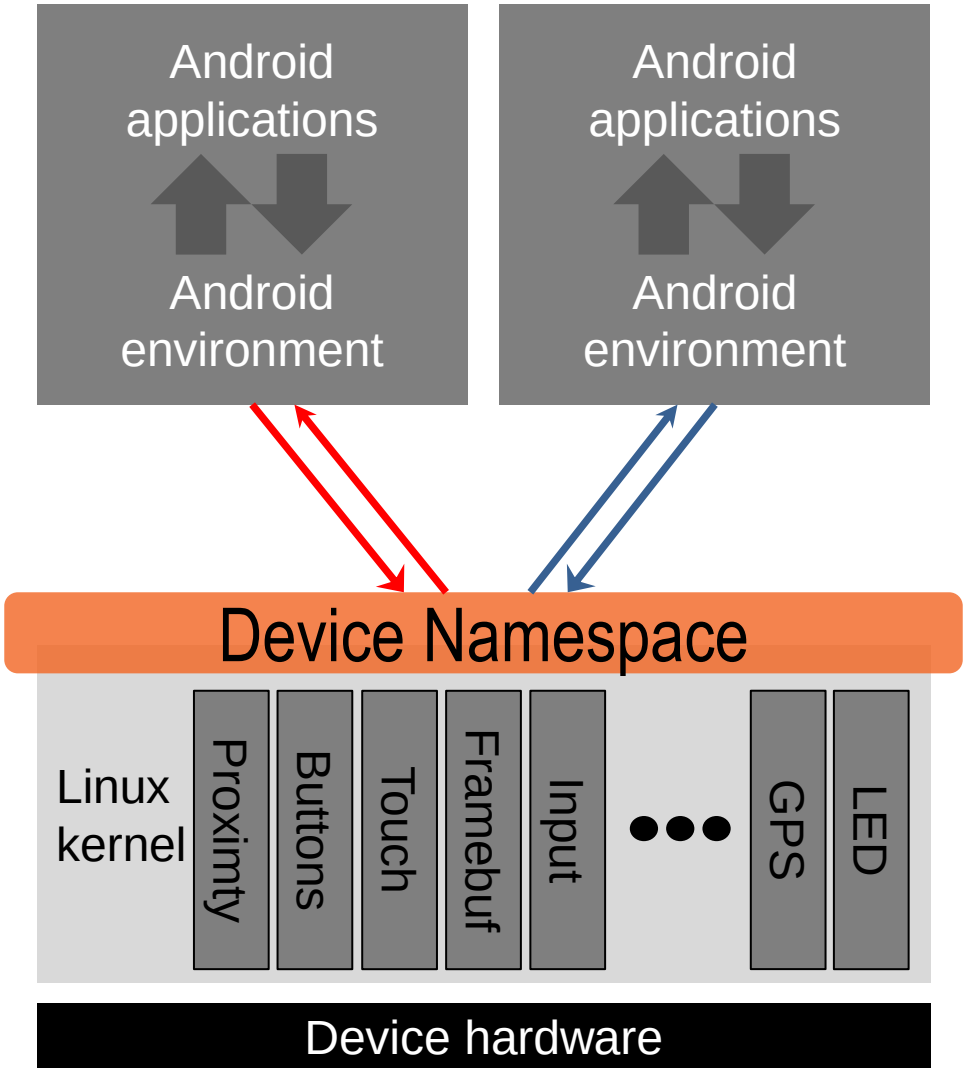

Android applications
Android environment
Android applications
Android environment
?
Linux kernel
Framebuf
Device hardware


Android
applications
Android
environment
Linux
kernel
Framebuf
Input
Device hardware


Android applications
Android environment
Android applications
Android environment
?
Linux kernel
Framebuf
Input
Device hardware


Android applications
Android environment
Android applications
Android environment
Device Namespace
Linux kernel
Framebuf
Input
Device hardware


Android applications
Android environment
Android applications
Android environment
Device Namespace
Linux kernel
Proximty
Touch
Framebuf
Input
Device hardware


Android applications
Android environment
Android applications
Android environment
Device Namespace
Linux kernel
Proximty
Buttons
Touch
Framebuf
Input
GPS
LED
Device hardware

Mobile Virtualization Challenges

Challenge 1: device diversity

- plethora of peripherals not virtualized
- key logical devices not virtualized

⇨ **virtualize physical & logical devices**

Mobile Virtualization Challenges

Challenge 1: device diversity

- plethora of peripherals not virtualized
- key logical devices not virtualized

⇨ **virtualize physical & logical devices**

Challenge 2: interactive usage

- users interact with one app at a time
- foreground vs. background apps

⇨ **multiplex access based on context**

Device Namespaces

Device diversity: traditional virtualization

- create the illusion that processes interact exclusively with a set of devices
- hide the fact that other processes interact with the same set of devices
- Device major/minor (e.g. loop, dm), and device setup and internal state

“Traditional” virtualization

Examples:

- alarm-dev
- binder
- logger
- wakelocks
- …

“Traditional” virtualization

Typical driver:

- global driver state
- per open fd state

- open() is special

- read/write/ioctl etc
  use per open fd state
  (and global state)

Virtualized driver?

“Traditional” virtualization

Typical driver:

- global driver state
- per open fd state

- open() is special

- read/write/ioctl etc
  use per open fd state
  (and global state)

⇨

Virtualized driver:

- per-devns state

“Traditional” virtualization

Typical driver:

- global driver state
- per open fd state

- open() is special

- read/write/ioctl etc
  use per open fd state
  (and global state)

Virtualized driver:

⇨ - per-devns state

⇨ - per open fd state points
  to per-devns state

“Traditional” virtualization

Typical driver:

- global driver state
- per open fd state
- open() is special
- read/write/ioctl etc use per open fd state (and global state)

Virtualized driver:

⇨ - per-devns state

⇨ - per open fd state points to per-devns state

⇨ - obtain per-devns state and perform in context

“Traditional” virtualization

Typical driver:		Virtualized driver:
- global driver state	⇨	- per-devns state
- per open fd state	⇨	- per open fd state points to per-devns state
- open() is special	⇨	- obtain per-devns state and perform in context
- read/write/ioctl etc use per open fd state (and global state)	⇨	- read/write/ioctl etc use per open fd state and per-devns state (and global state)

“Traditional” virtualization

A peek at the code:

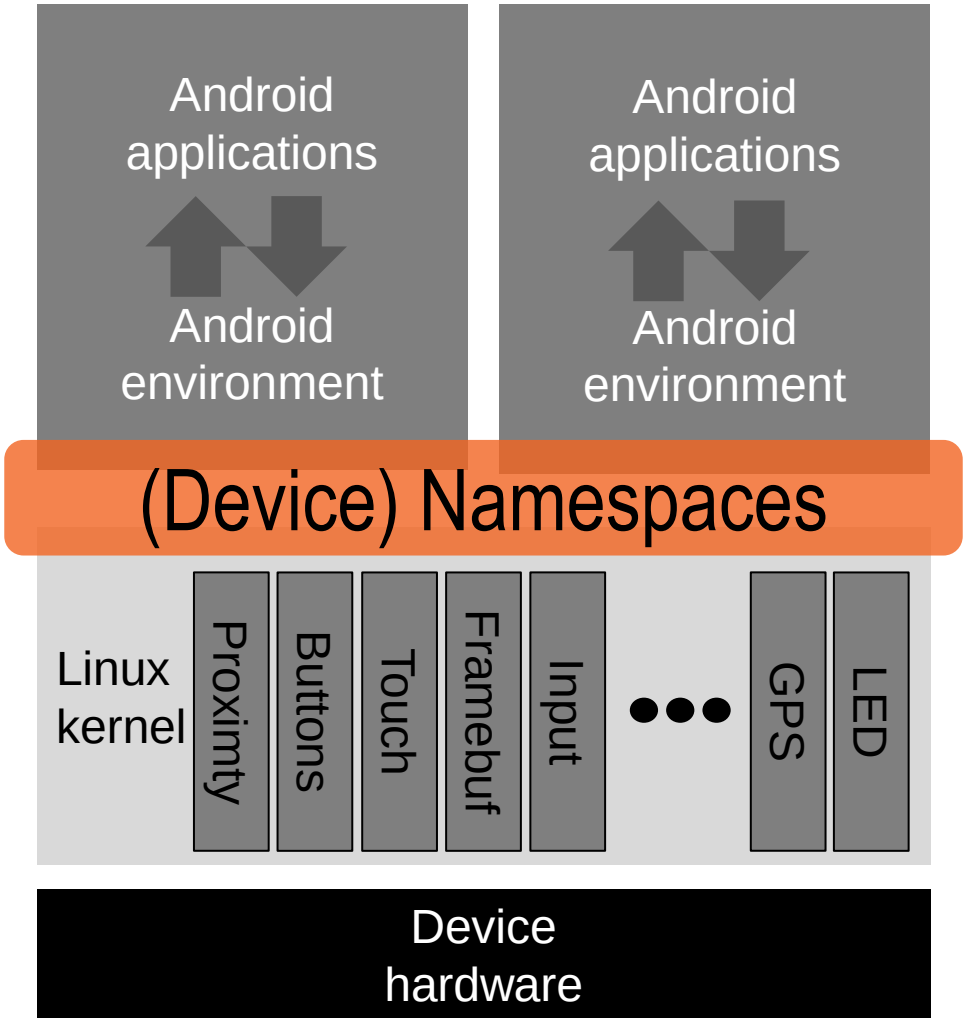
- alarm-dev
- binder
- …

Device Namespaces

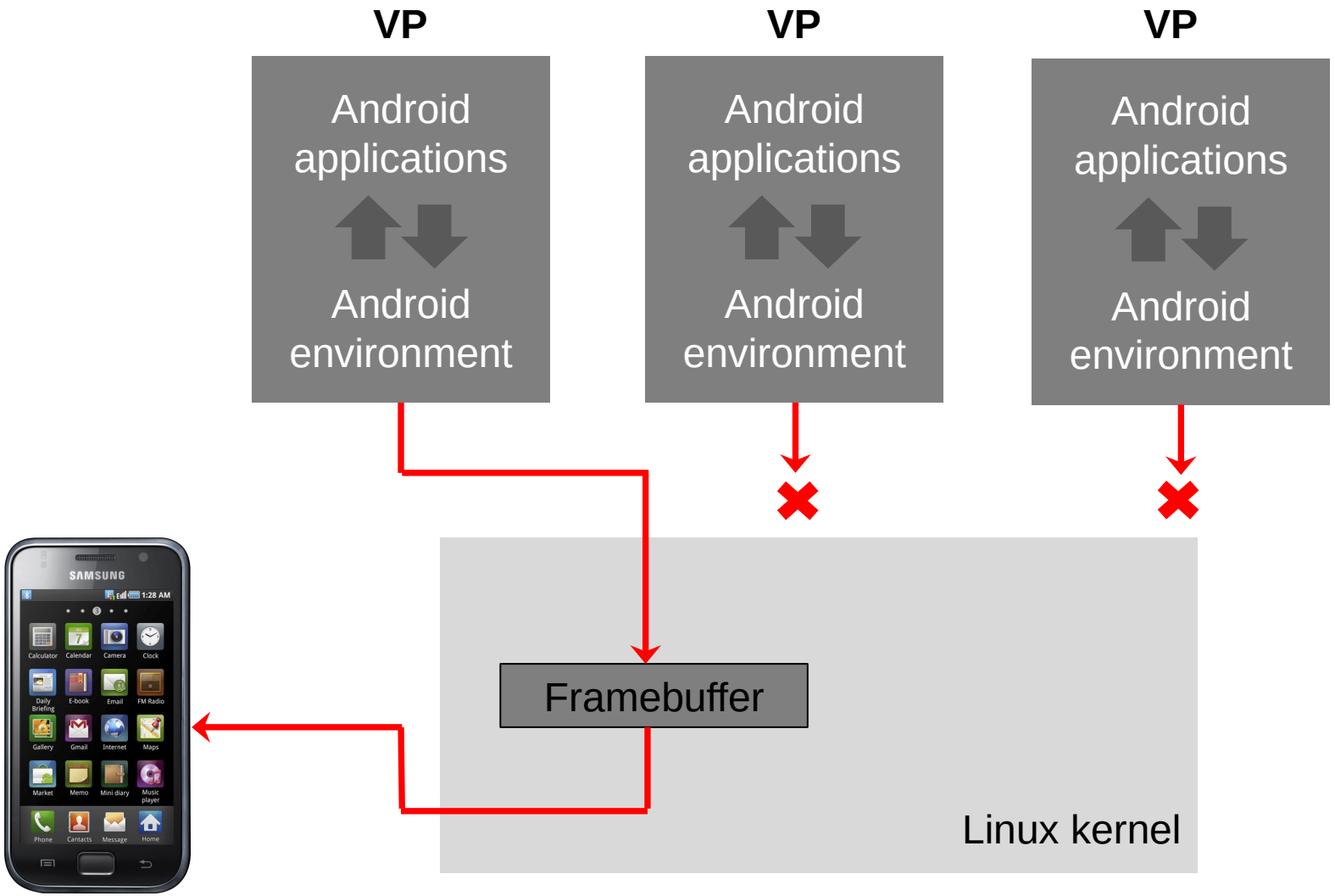
Interactivity: context-aware virtualization

- concept of an active namespace, with which the user actually interacts
- ability to switch namespaces, to allow interacting with multi-namespaces
- users really interact with one namespace at a time

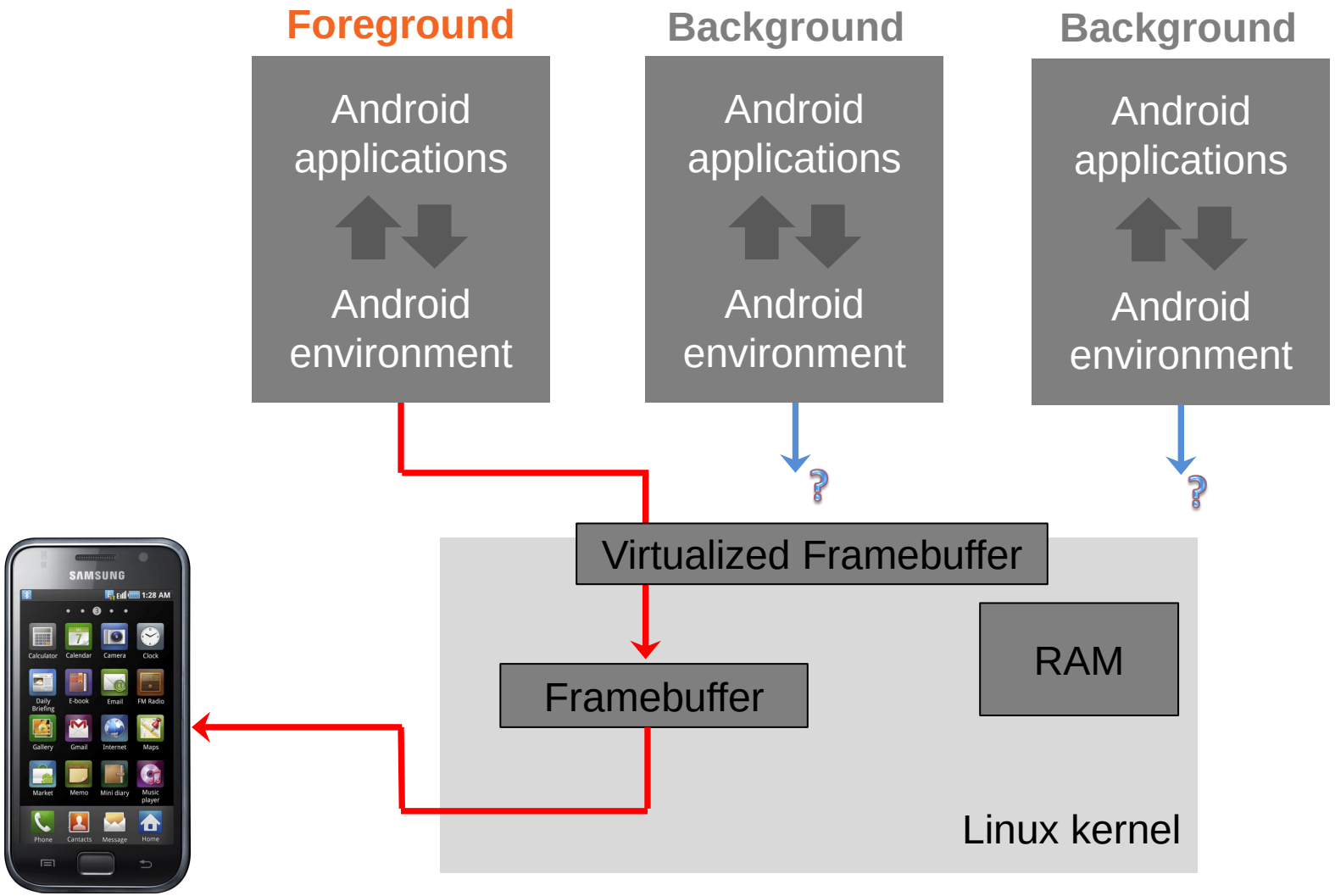
Device Namespaces



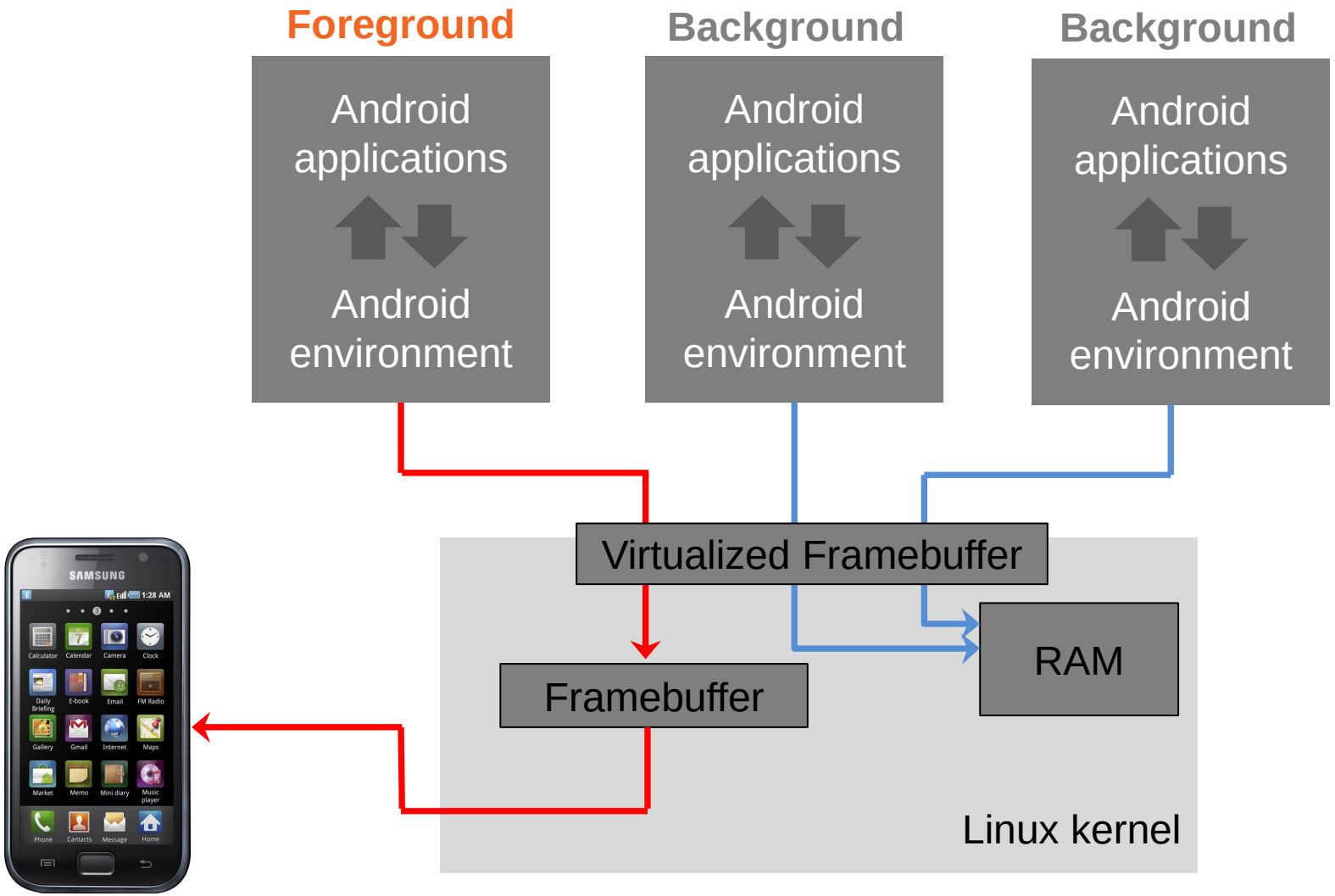
Framebuffer ?



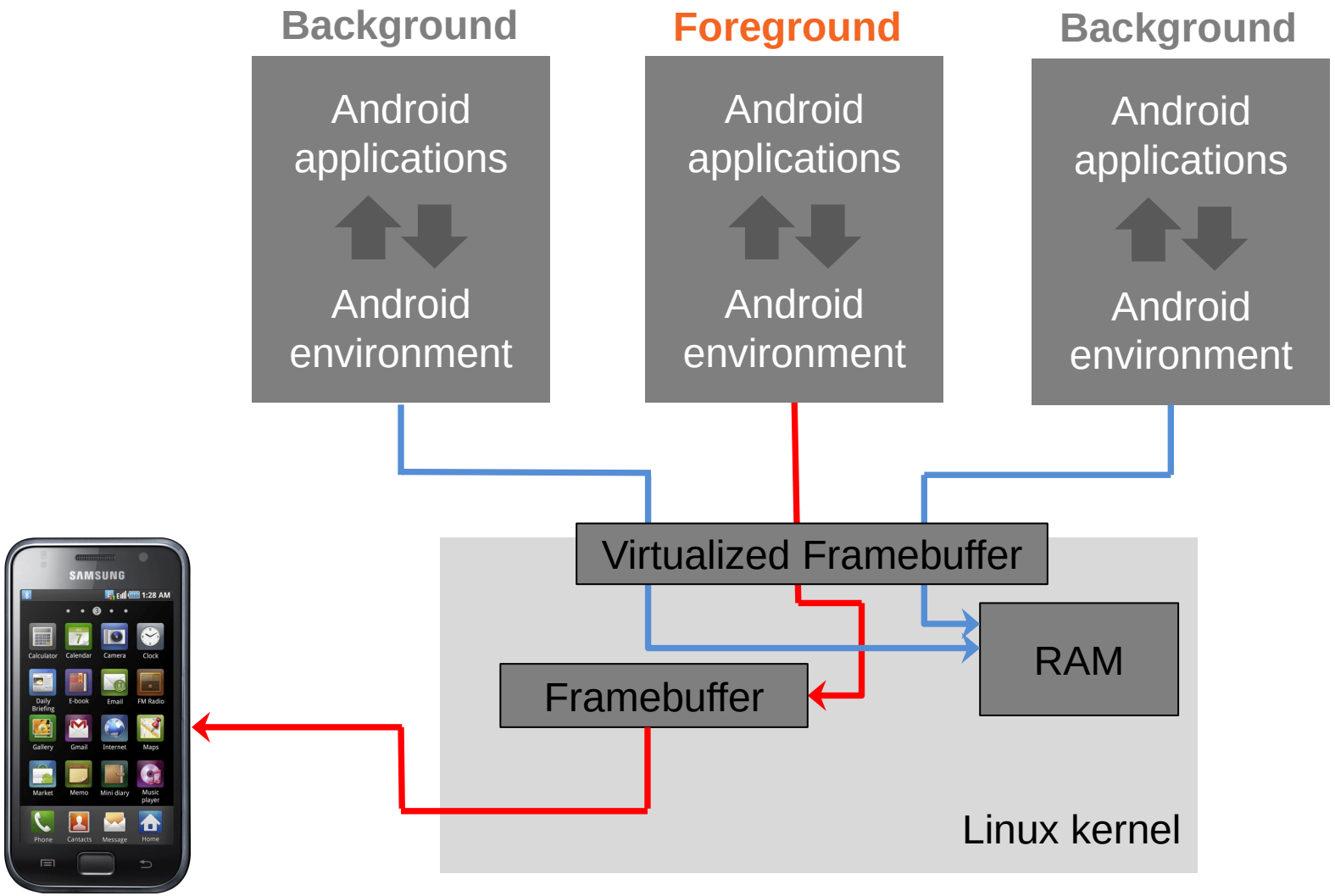
Framebuffer: device namespaces



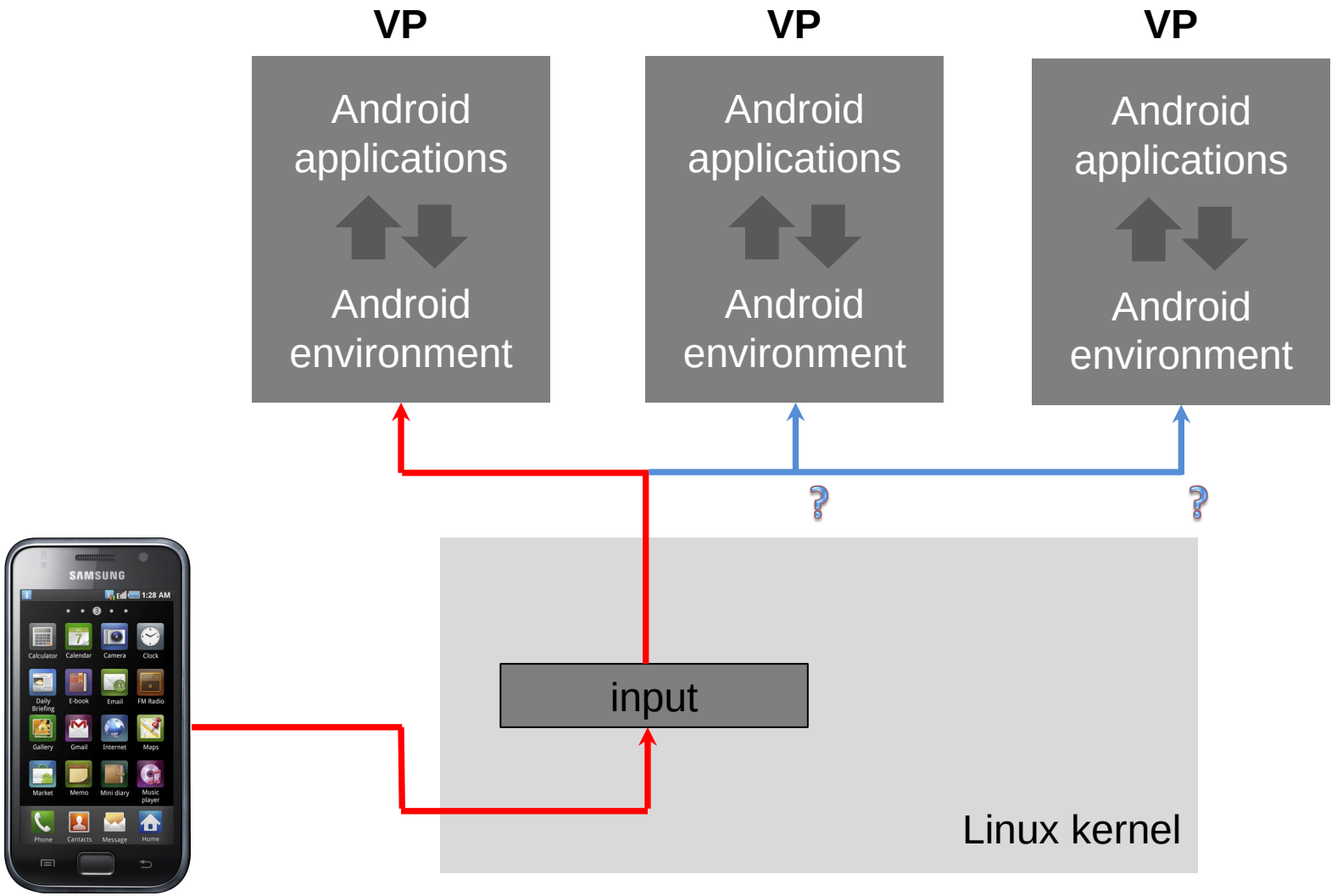
Framebuffer: device namespaces



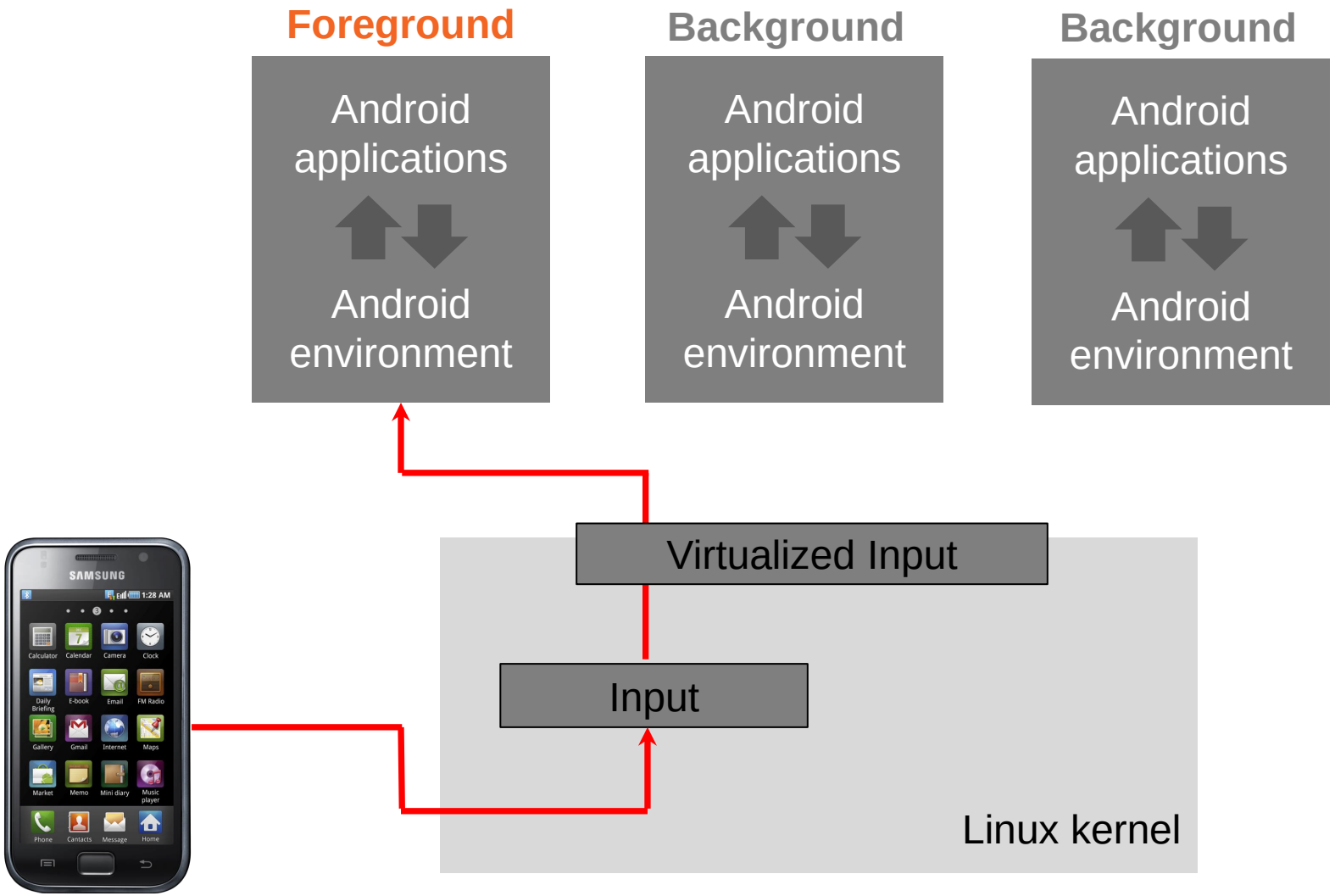
Framebuffer: device namespaces



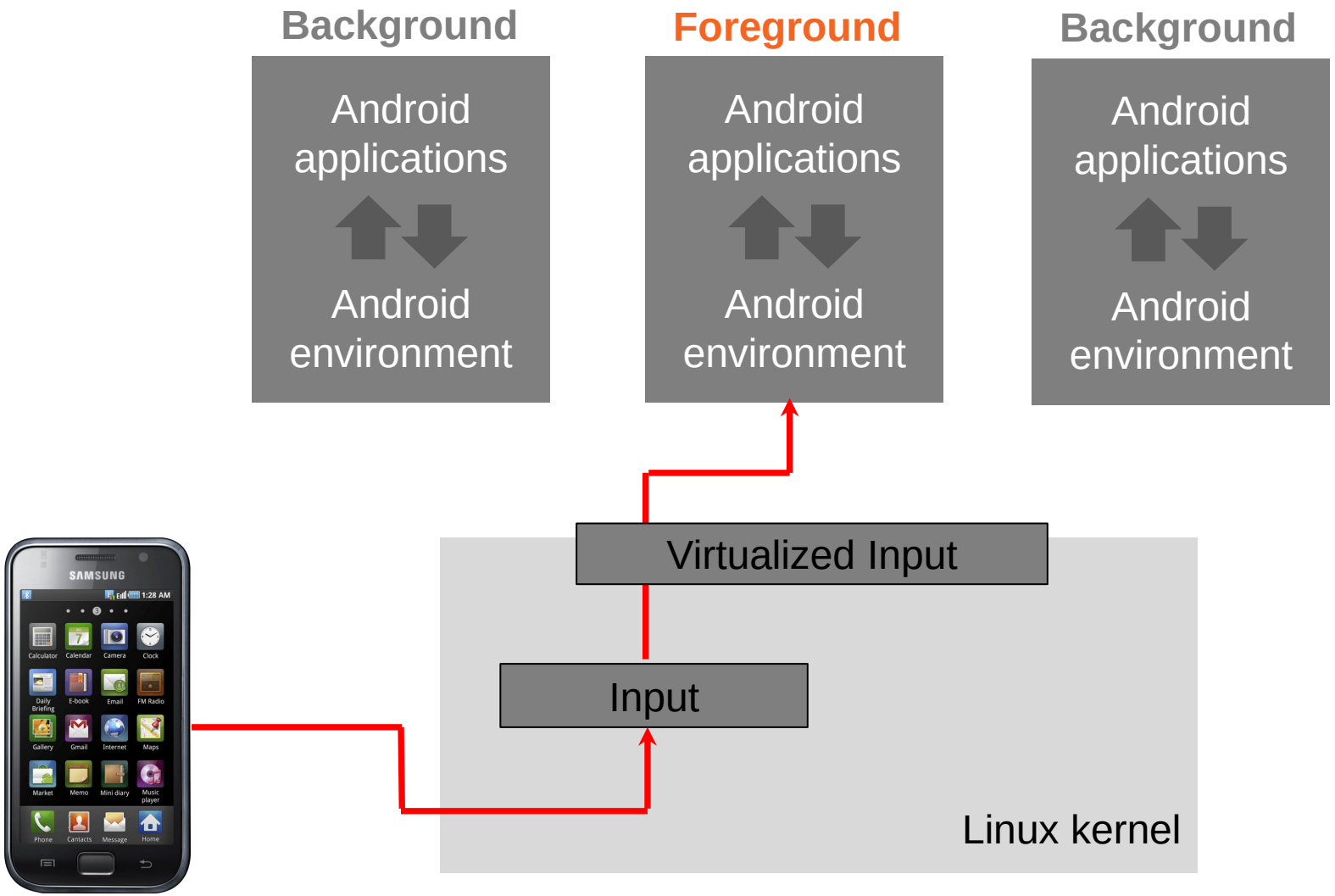
Input ?


VP
VP
VP
Android applications
Android environment
Android applications
Android environment
Android applications
Android environment
?
?
input
Linux kernel
SAMSUNG

Input: device namespaces



Input: device namespaces

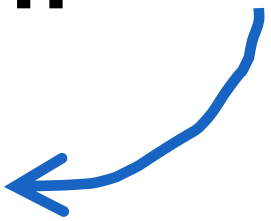


“Context-aware” virtualization

Typical driver:		Virtualized driver:
- global driver state	⇨	- **per-devns state**
- per open fd state	⇨	- per open fd state points to per-devns state
- open() is special	⇨	- obtain per-devns state and perform in context
- read/write/ioctl etc use per open fd state (and global state)	⇨	- read/write/ioctl etc use per open fd state and per-devns state (and global state)

“Context-aware” virtualization

Typical driver:		Virtualized driver:
- global driver state	⇨	- **per-devns state**
- per open fd state	⇨	- per open fd state points to per-devns state
- open() is special	⇨	- obtain per-devns state and perform in context
- read/write/ioctl etc use per open fd state (and global state)	⇨	- read/write/ioctl etc use per open fd state and per-devns state (and global state)



per devns state:

- active flag (foreground/background)
- callbacks (create, destroy, switch)

“Context-aware” virtualization

A peek at the code:

- input layer
- backlight
- LED
- …

Device namespaces in action

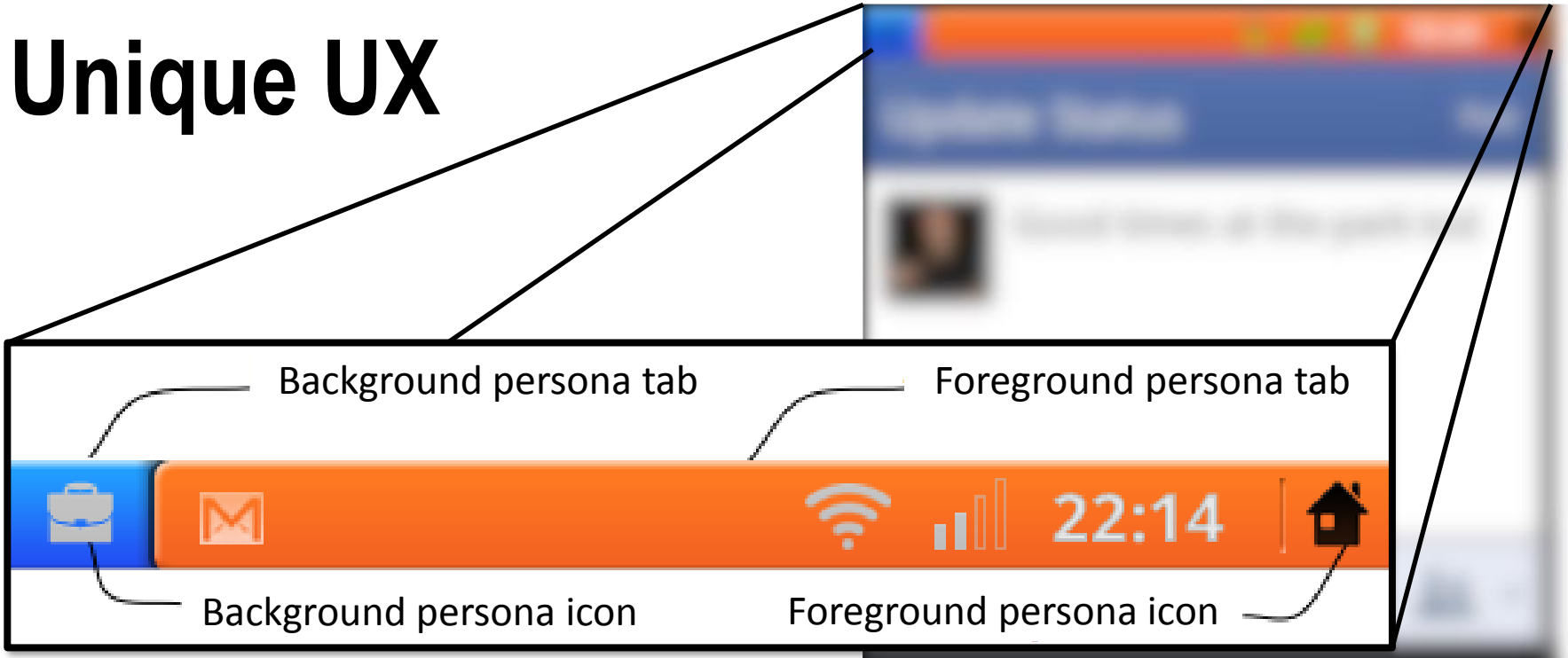
A quick hands on with the Android emulator

User-experience ?

User-experience

Identity
Awareness
Switching
Sharing

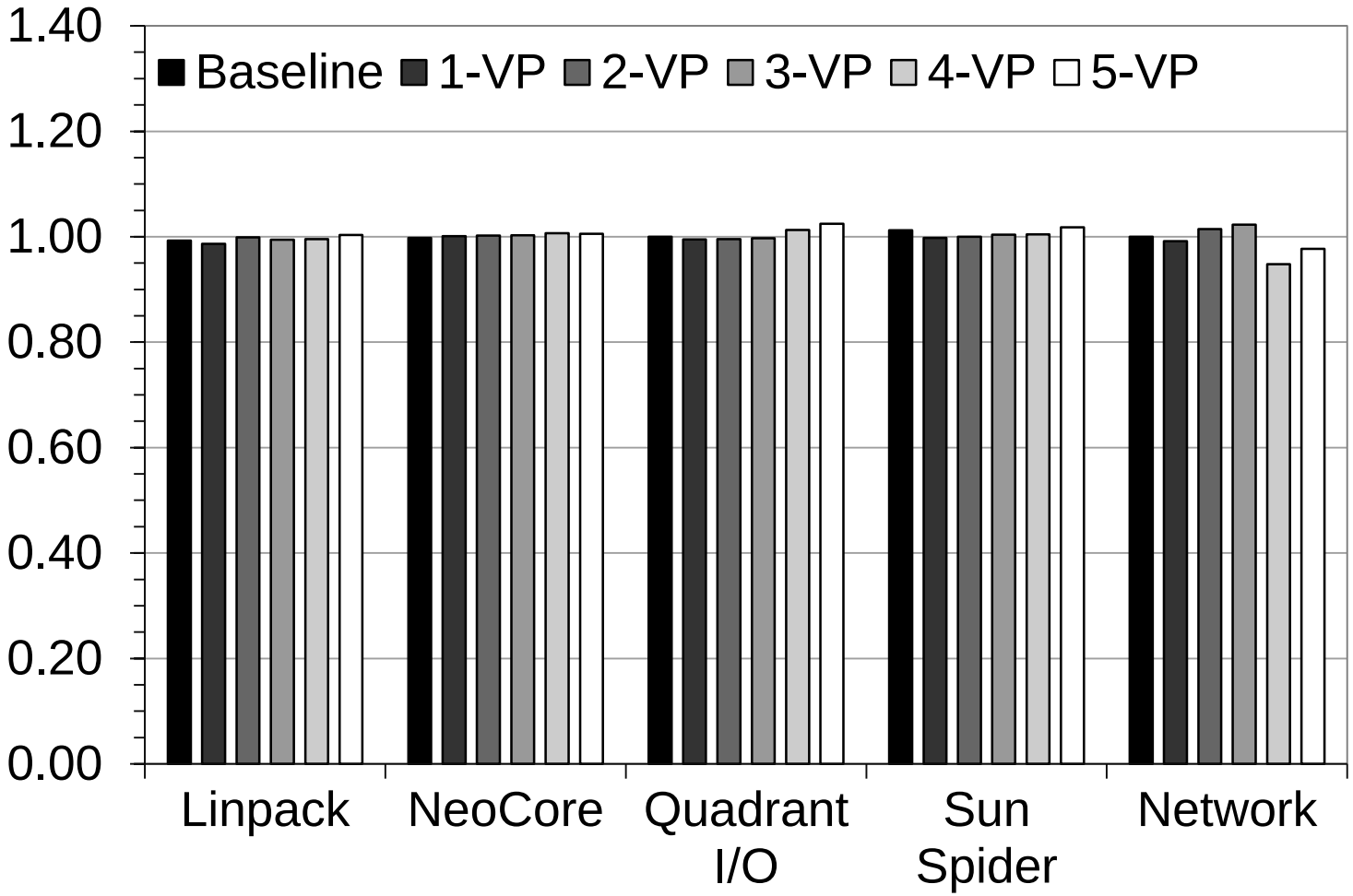
Unique UX


Background persona tab
Foreground persona tab
22:14
Background persona icon
Foreground persona icon

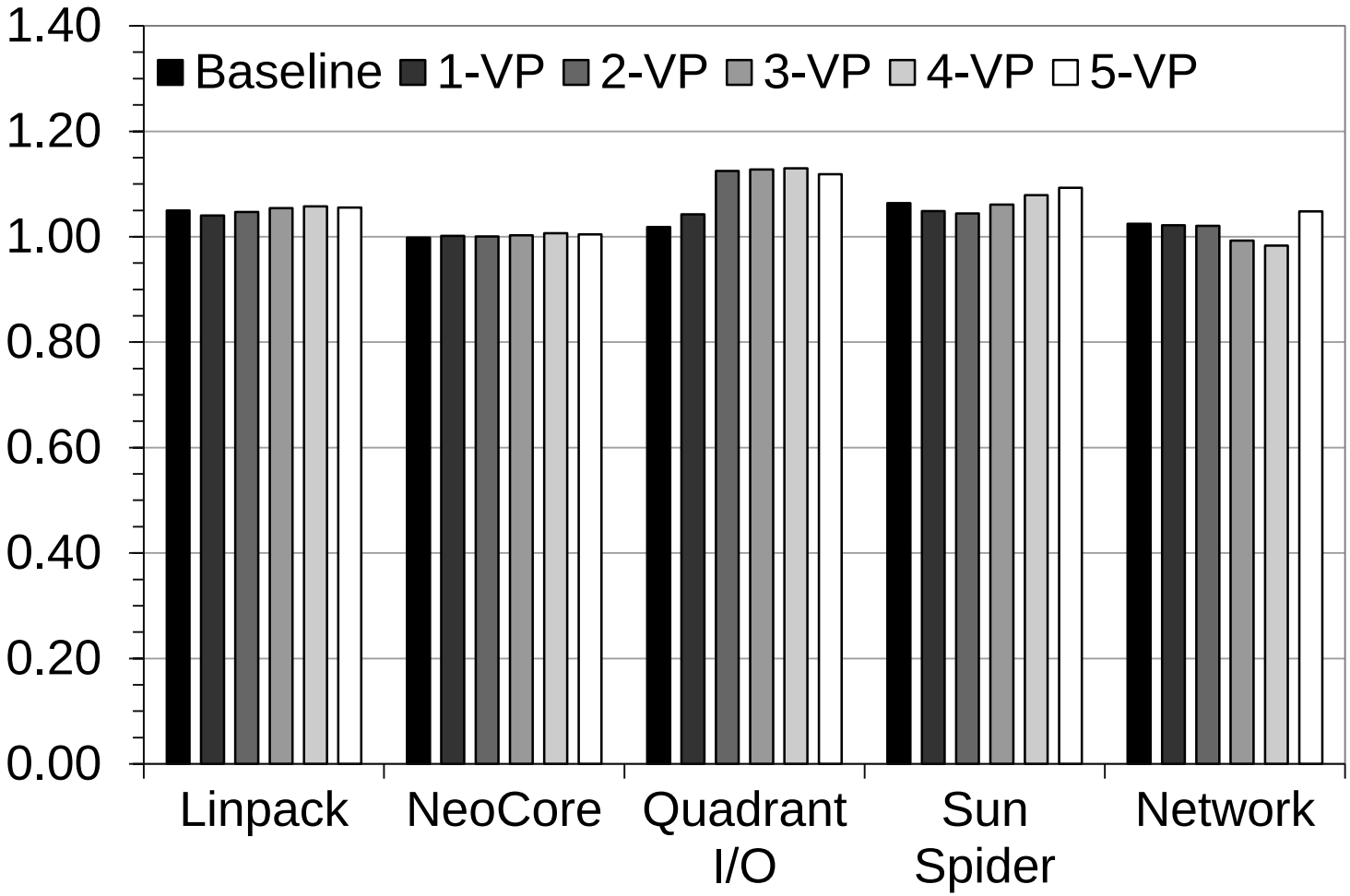
Experimental Benchmarks

- CPU (Linpack)
- Graphics (Neocore)
- Storage (Quadrant)
- Web browsing (SunSpider)
- Networking (custom)

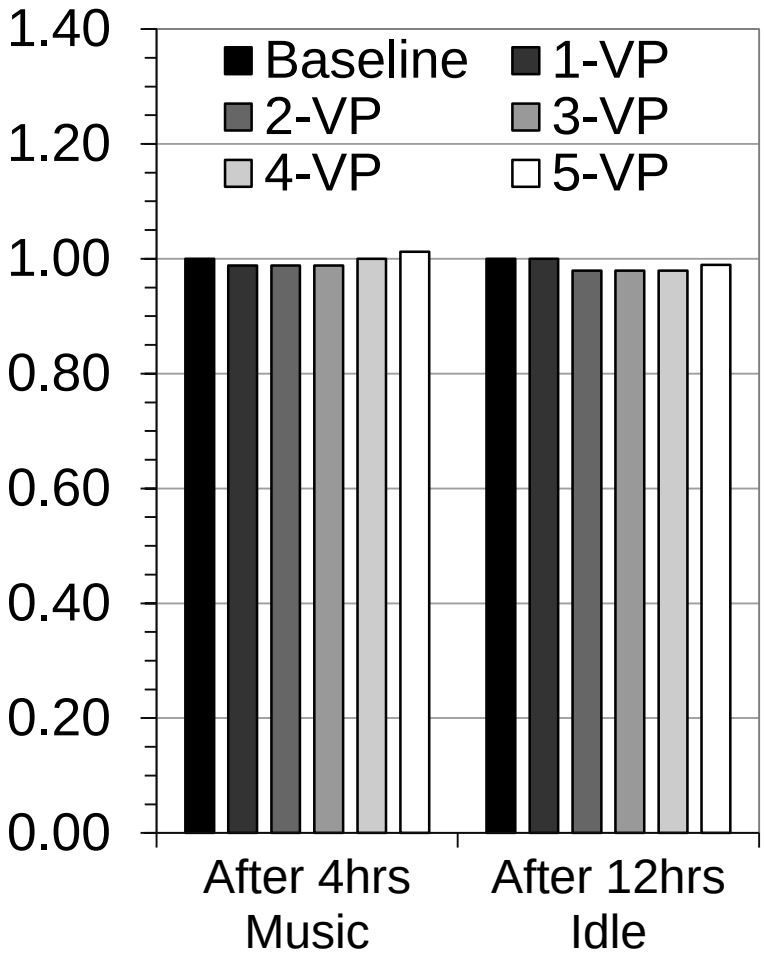
Runtime Overhead (Idle)



Runtime Overhead (load)



Power Consumption Overhead



Summary

- Multi-persona Android
- Device namespaces (?!)

More info:

https://github.com/Cellrox/devns-patches/wiki

orenl@cellrox.com