CELLS

A virtual Smartphone Architecture

Jeremy Andrus Christoffer Dall Alexander Van't Hof Oren Laadan Jason Nieh

Columbia University
in the city of new york

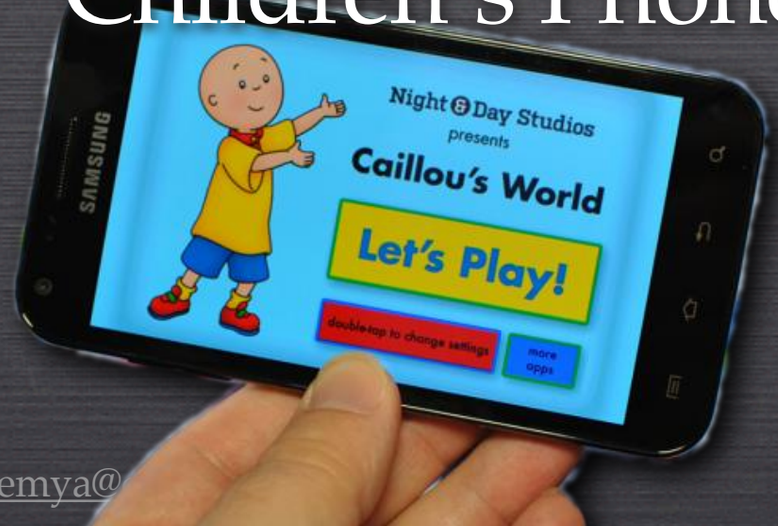
23rd ACM Symposium on Operating Systems Principles, Cascais, Portugal, October 2011

Business Phone

Personal Phone

Developer Phone

Children's Phone



Virtualization

Server Virtualization
Bare-Metal Hypervisor

poor device support / sharing

Desktop Virtualization

Hosted Hypervisor

poor device performance

host user space

Hypervisor / VMM

Host OS Kernel

kernel module

emulated devices

Hardware

Non-Virtualization
User Space SDK

no standard apps
less secure

custom user space API for isolated apps

OS Kernel

Hardware

Key Challenges

- device diversity

		microphone	headset
Power	Touchscreen	Buttons	GPS
Cell Radio	WiFi	GPU	Framebuffer
h.264 accel.	pmem	Binder IPC	Compass
camera(s)	speakers	Accelerometer	RTC / Alarms

- mobile usage model

➡ graphics-accelerated UI

Cells

Cells

Key Observation

large: lots of windows / apps

small:

one app at a time

Cells

Key Observation

screen real-estate is limited, and
mobile phone users are accustomed
to interacting with *one thing* at time

Cells

Usage Model

foreground / *background*

Cells
Complete Virtualization

- multiple, isolated virtual phones (VPs) on a single mobile device
- 100% device support in each VP
  - unique phone numbers - single SIM!
  - accelerated 3D graphics!

Cells
Efficient Virtualization

- less than 2% overhead in runtime tests
- imperceptible switch time among VPs

Cells
Transparent Virtualization

- each VP sees / uses all devices
- user can run any unmodified apps
- foreground VP switches like an app

Single Kernel: Multiple VPs

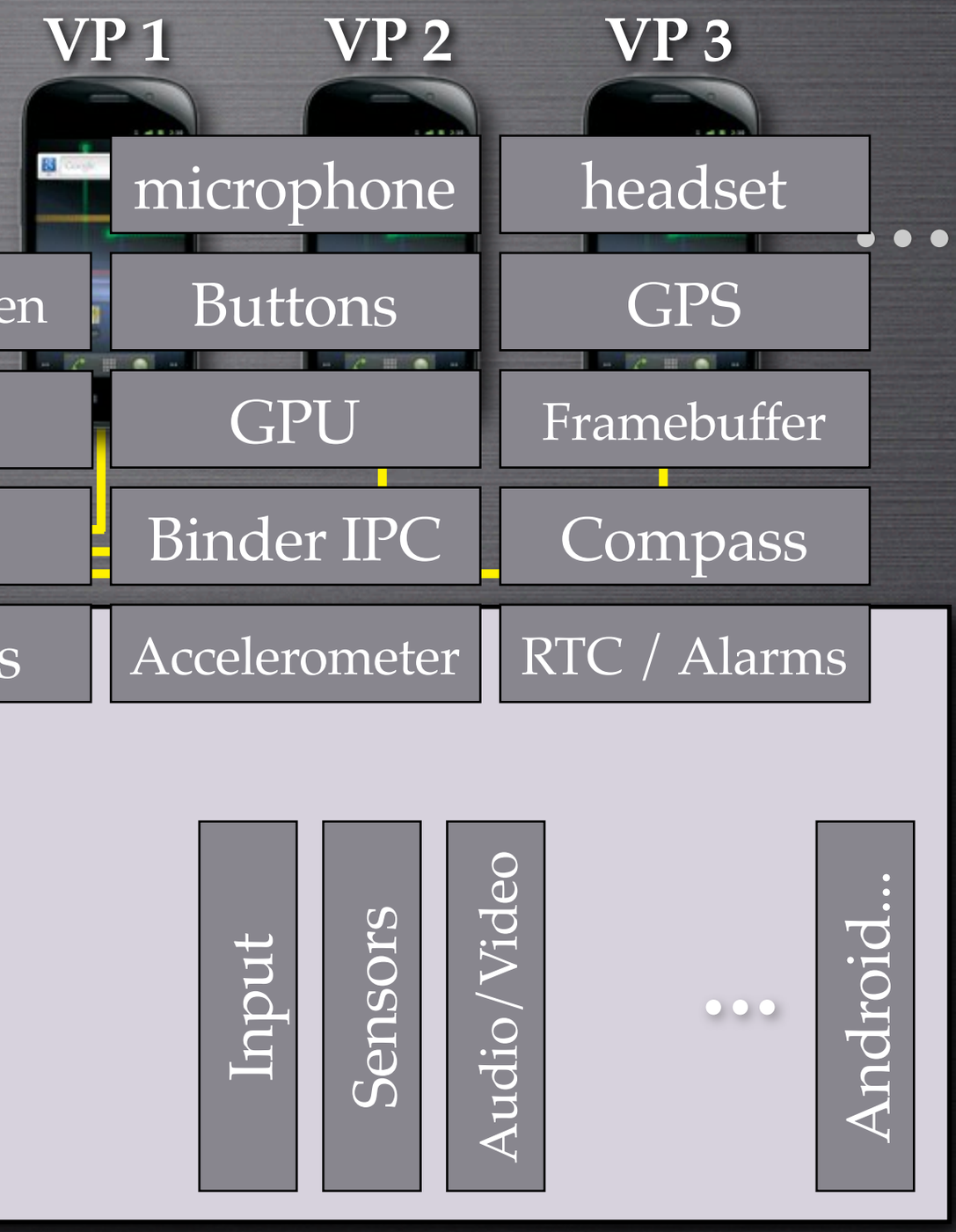
isolated collection of processes

virtualize at OS interface

Linux Kernel

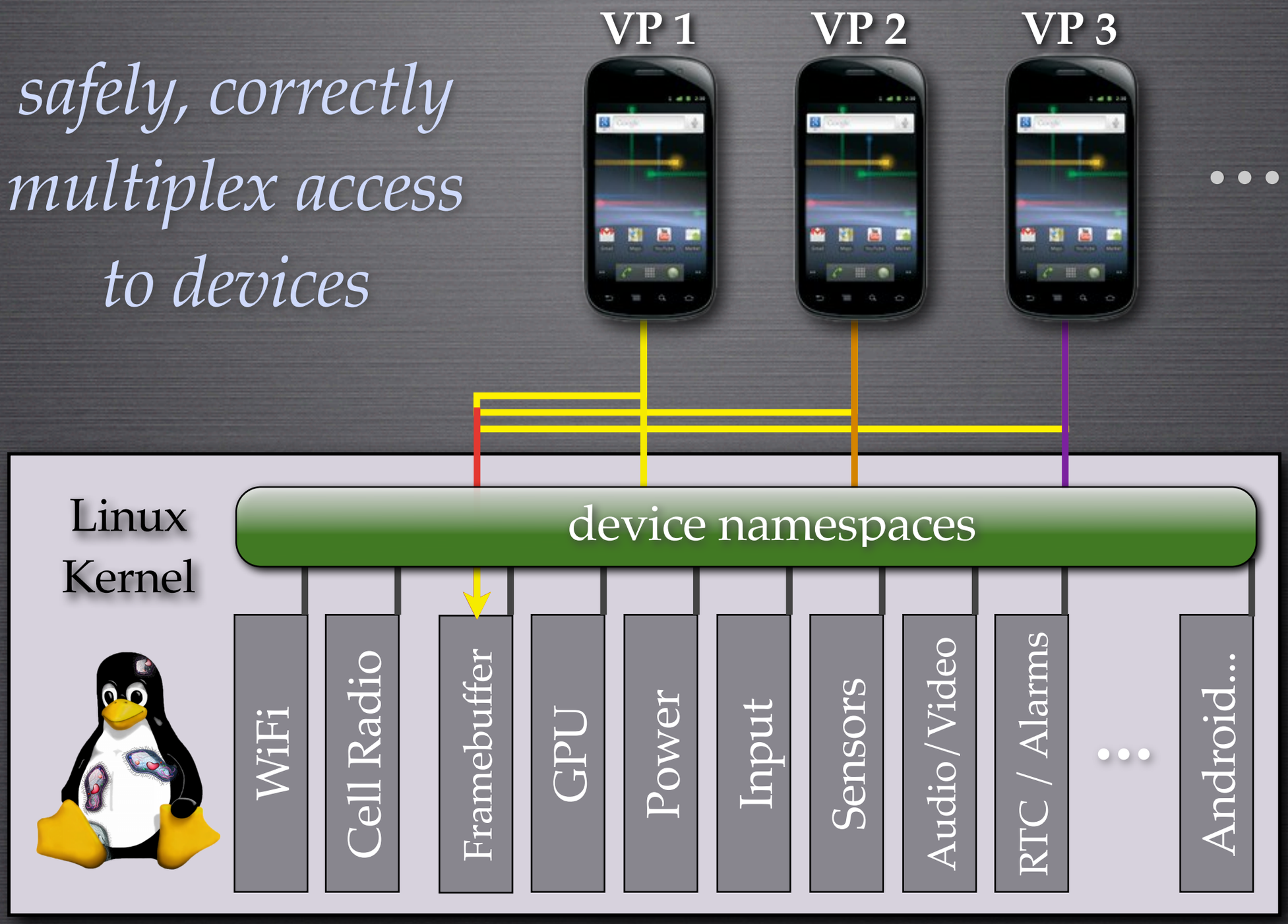
Single Kernel: Device Support

all VPs access the same device



Device Namespaces

safely, correctly multiplex access to devices



Cells

VP 1 VP 2 VP 3

device namespaces

\+

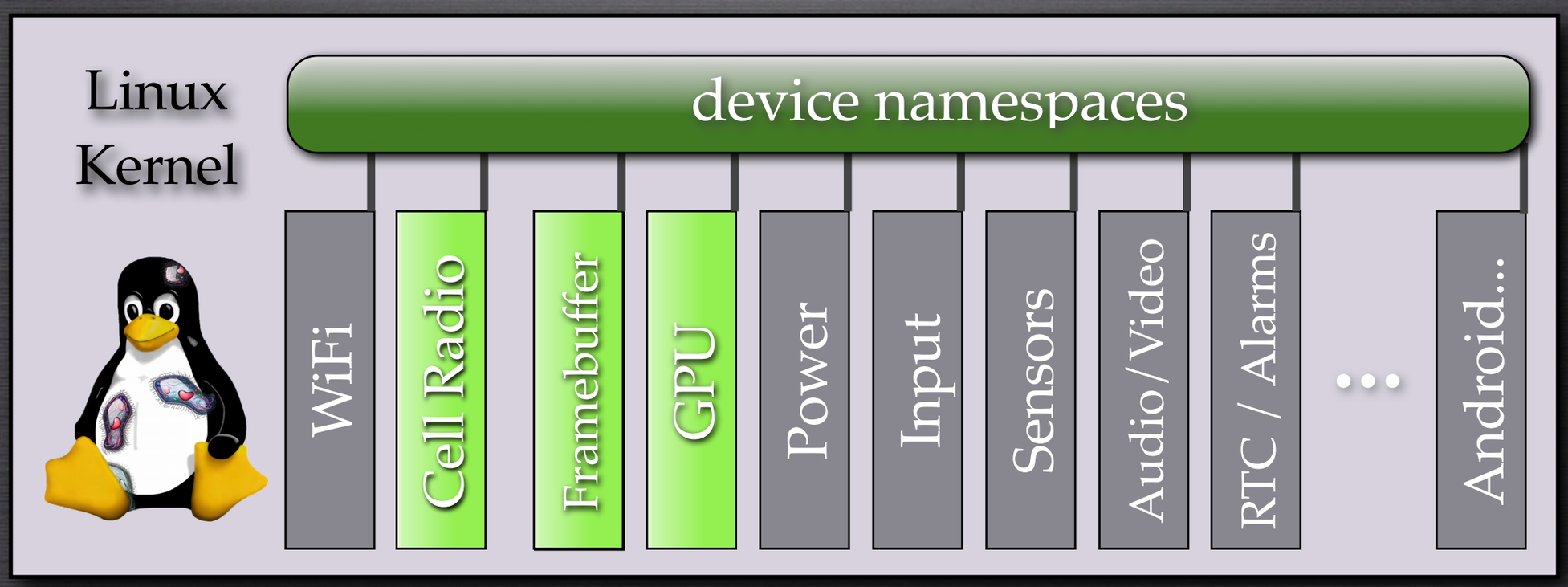
foreground / *background*

...

efficient basic graphics virtualization

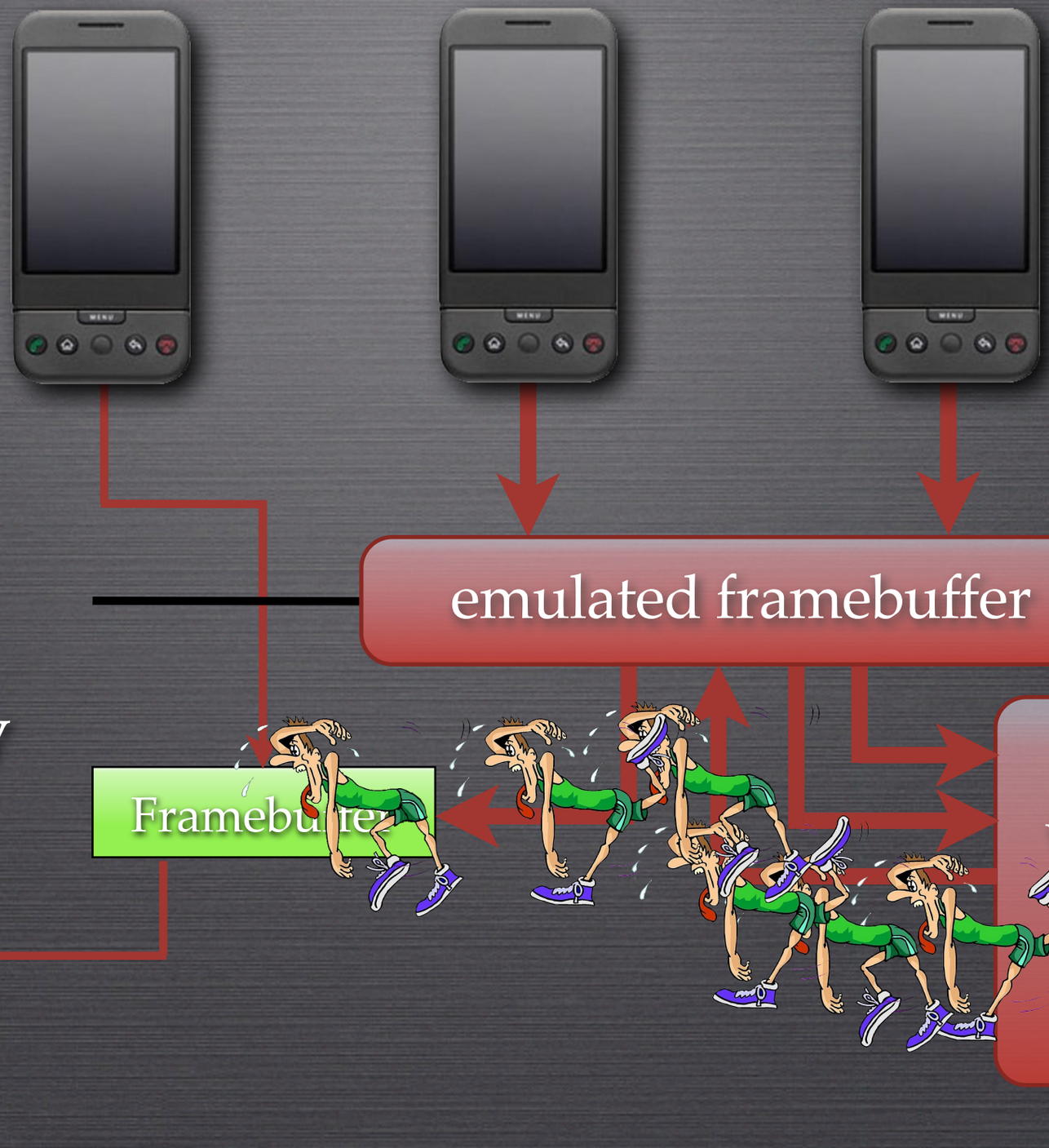
hardware accelerated graphics

proprietary / closed interface



Approach 1: Single Assignment

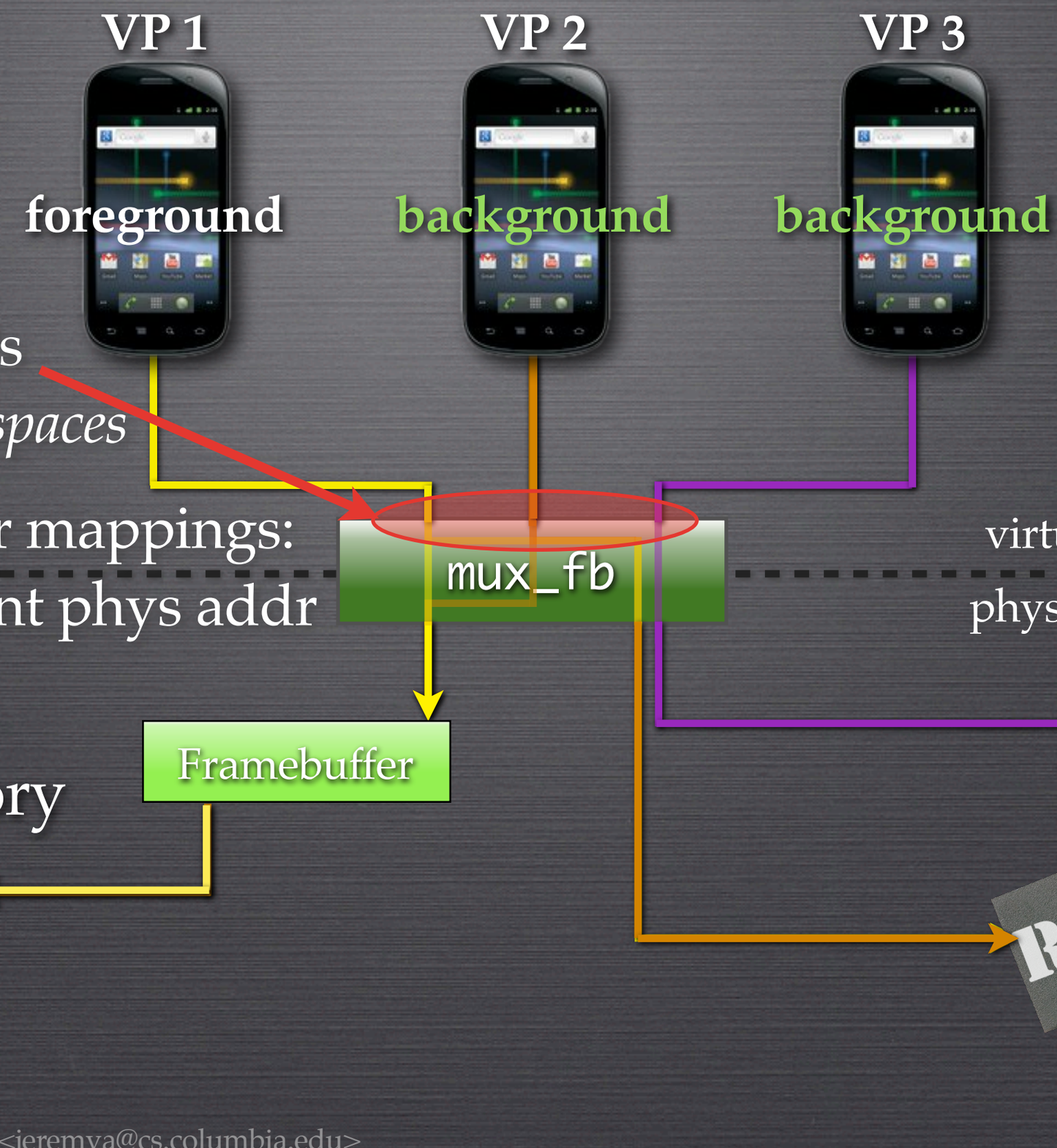
Approach 2: Emulated Hardware



Cells: Device Namespaces

`mux_fb` presents identical device interface to all VPs using *device namespaces*

swap virt addr mappings: point to different phys addr



Accelerated Graphics

VP: just a set of processes!

process isolation

Framebuffer

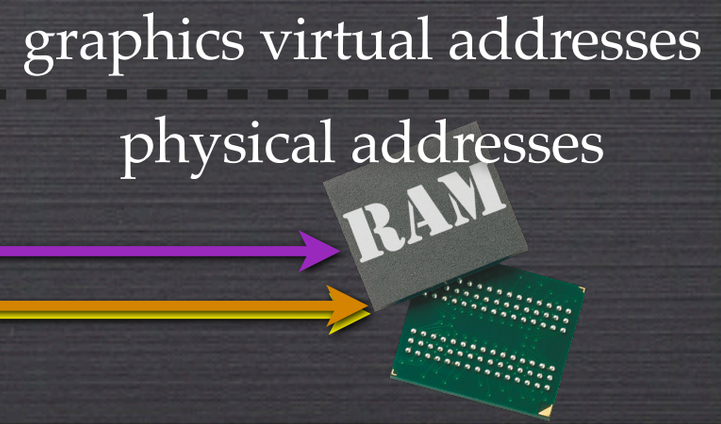
MMU

GPU

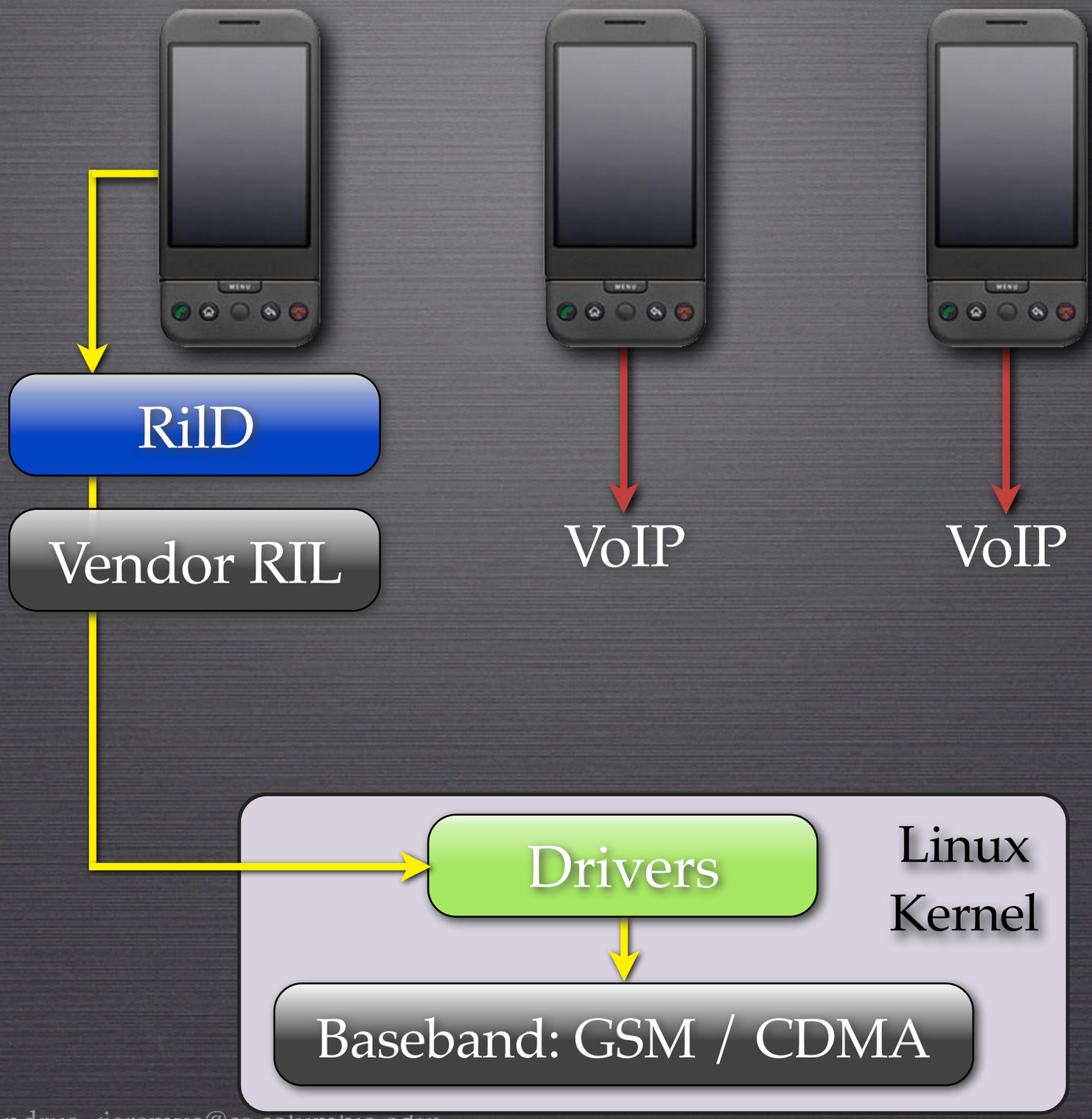
graphics virtual addresses

physical addresses

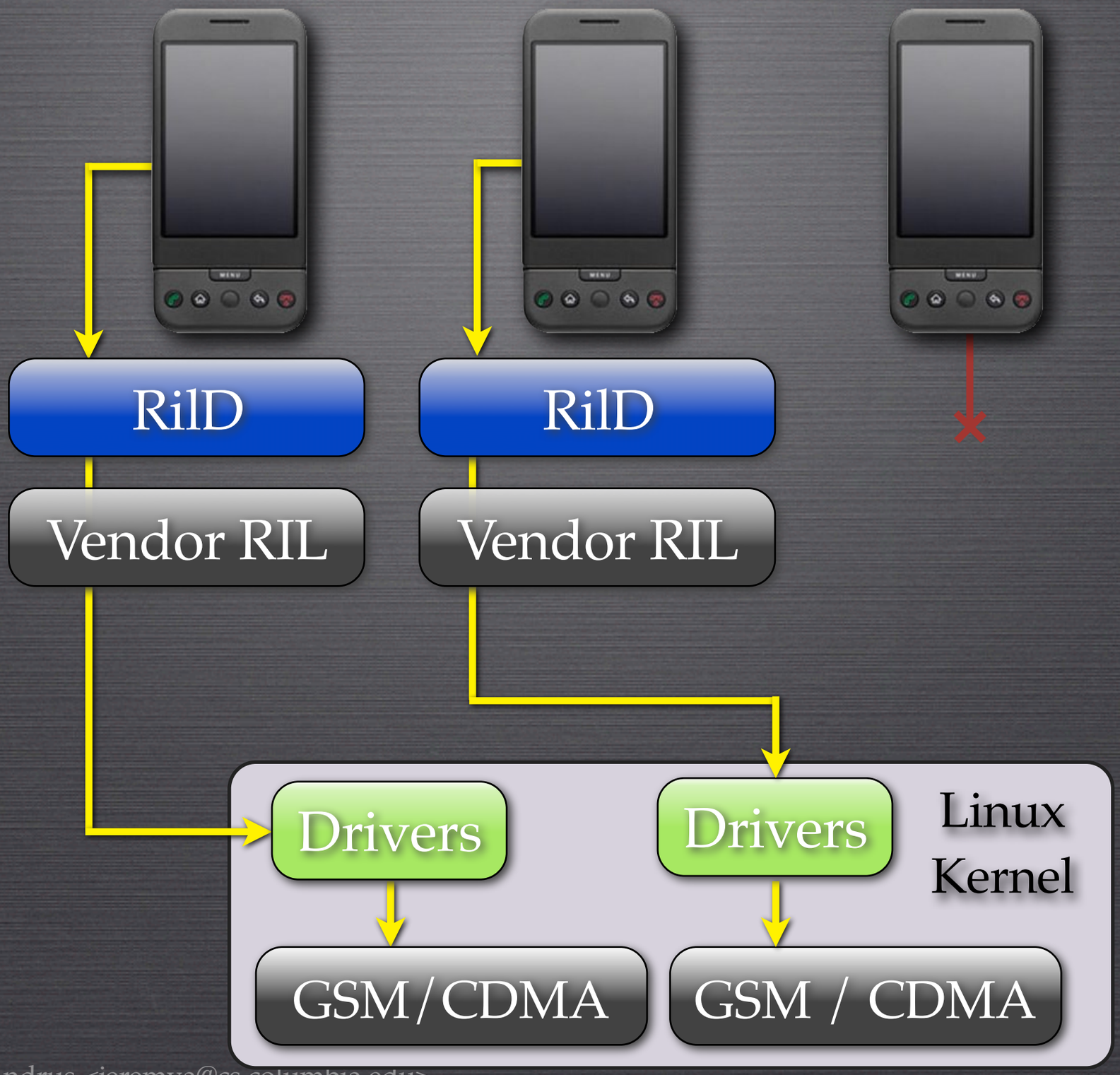
Device Namespace + Graphics Context



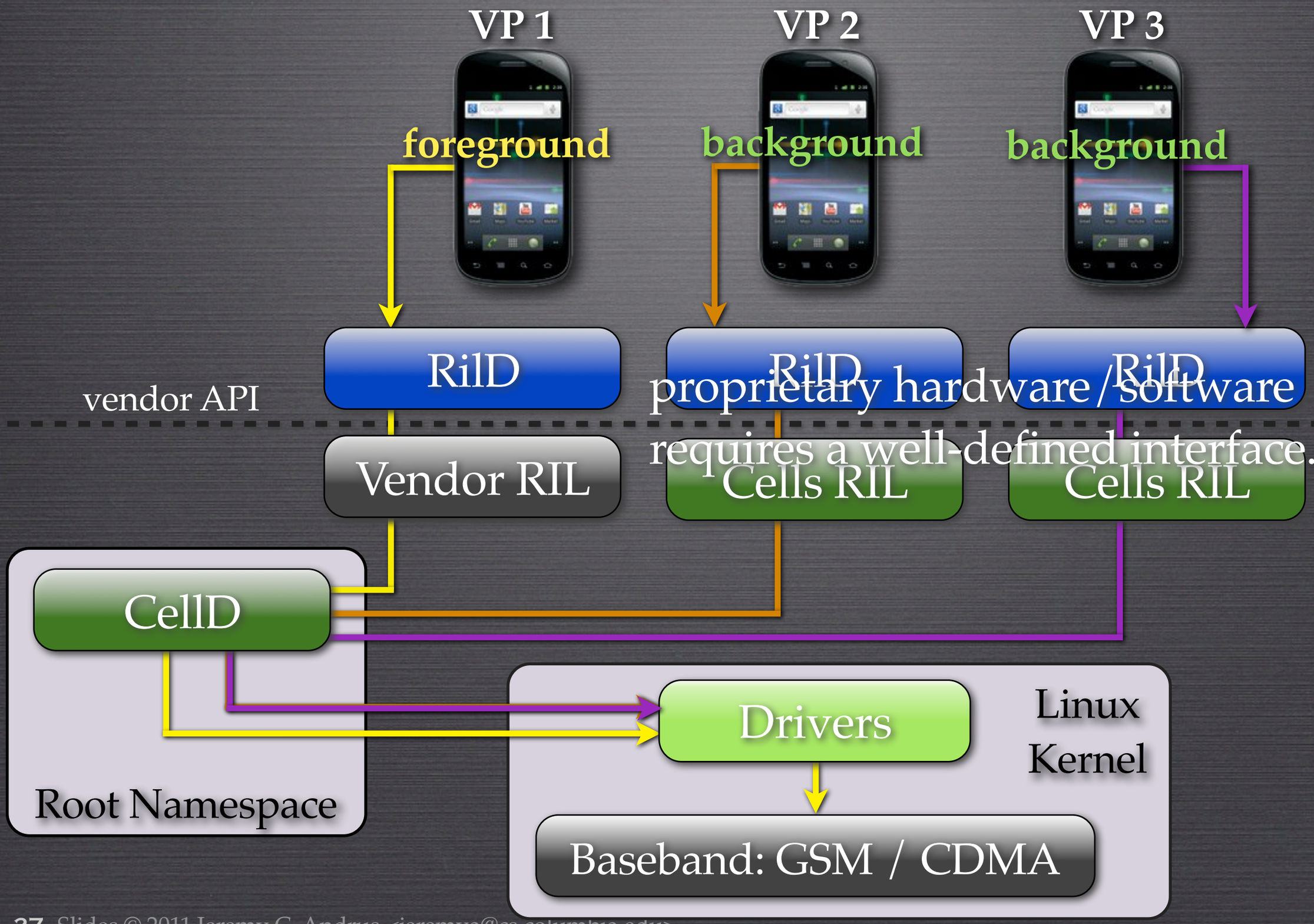
VoIP?



Dual-SIM?



Cells: User-Level Namespace Proxy



Experimental Results

Setup

- Nexus S
- *five* virtual phones
- overhead vs. stock ⟨*Android 2.3*⟩

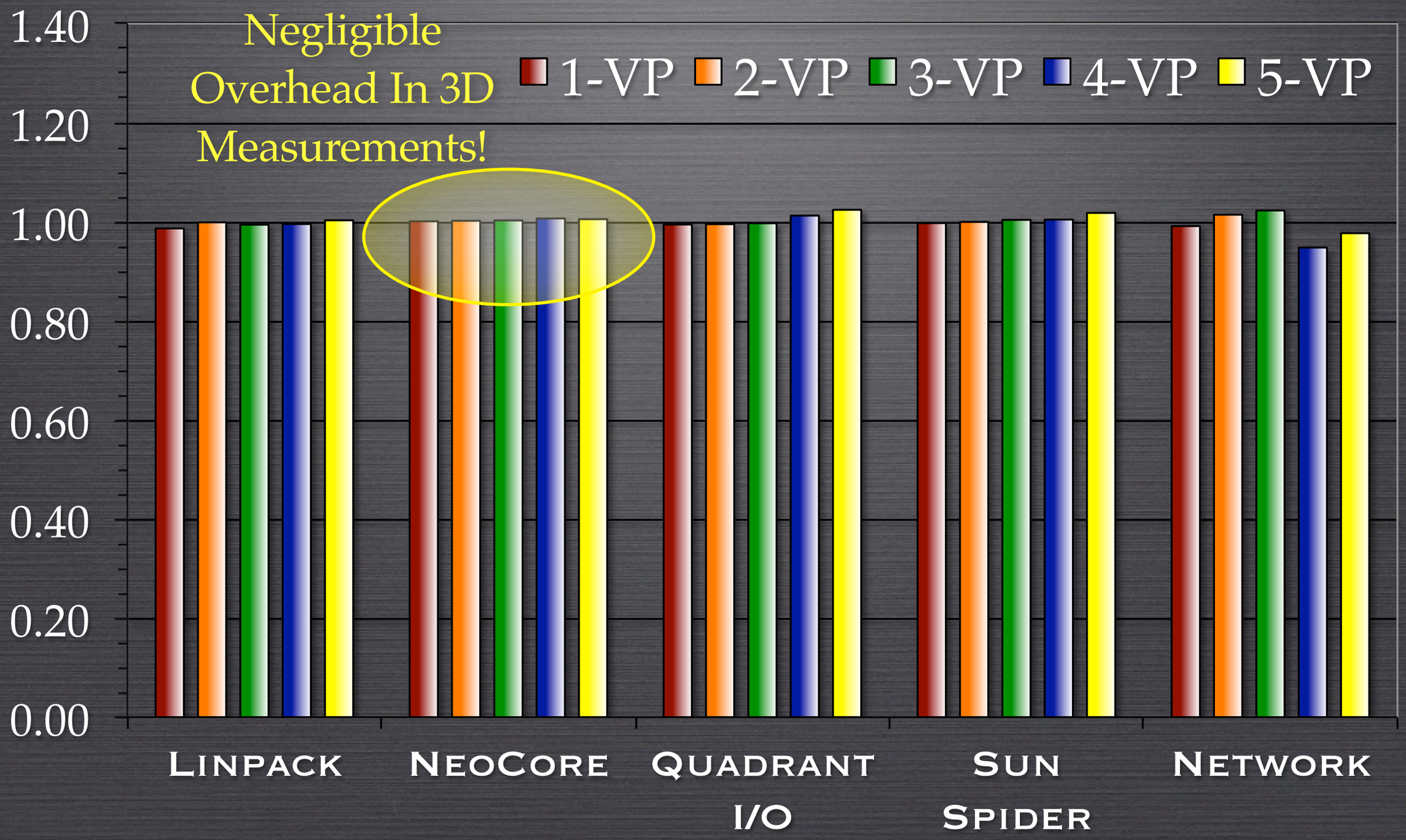
Experimental Results

Setup

- CPU ⟨*Linpack*⟩
- graphics ⟨*Neocore*⟩
- storage ⟨*Quadrant*⟩
- web browsing ⟨*Sun Spider*⟩
- networking ⟨*Custom WiFi Test*⟩

Experimental Results

Runtime Overhead



Demo

Cells

Complete, Efficient, Transparent Mobile Virtualization

- device namespaces
  - safely and efficiently share devices
- foreground / background
  - designed specifically for mobile devices
- implemented on Android
- less than 2% overhead on Nexus S

More Info

Software Systems Laboratory

cells.cs.columbia.edu

CELLROX

cellrox.com

