Cells: A Virtual Mobile Smartphone Architecture

Jeremy Andrus, Christoffer Dall
SOSP'11, October 23-26, 2011

단국대학교
컴퓨터 보안 및 OS 연구실
peonix120@gmail.com
임경환

2015. 06. 09

DKU 단국대학교 DANKOOK UNIVERSITY

Contents

- INTRODUCTION
- OVERALL ARCHITECTURE
- Security Features
- Security policy
- Conclusion
- Reference
- Q & A

INTRODUCTION

- **Platform shifting from computers to smartphones**

- **Need of Virtualizing smartphones**
  - ✓ Existing (system) virtualizations
    - High overhead
    - Hardware devices

- **CELLS: lightweight virtualization architecture**

Overview of Cells

- **Virtual Phones**
  - ✓ Multiple Android instead of multiple OS instances
- **Union FS**
  - ✓ To maximize Sharing common read-only code and data across VPs
  - ✓ minimizing memory consumption, enabling additional VPs to be instantiated without much overhead
- **Small display form factors of smartphones**
  - ✓ Foreground / Background model
- **Using a VoIP service**
  - ✓ To provide individual telephone numbers for each VP without the need for multiple SIM cards.

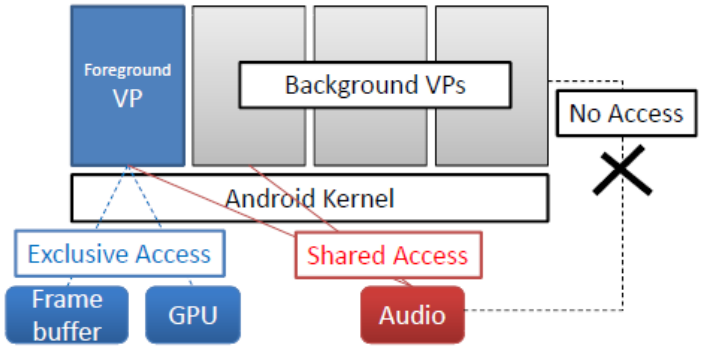
USAGE MODEL

- **Each VP is completely isolated from other VPs**

- **Foreground / Background model**
  - ✓ A user can easily switch among VPs by selecting one of the background VPs to become the foreground one.

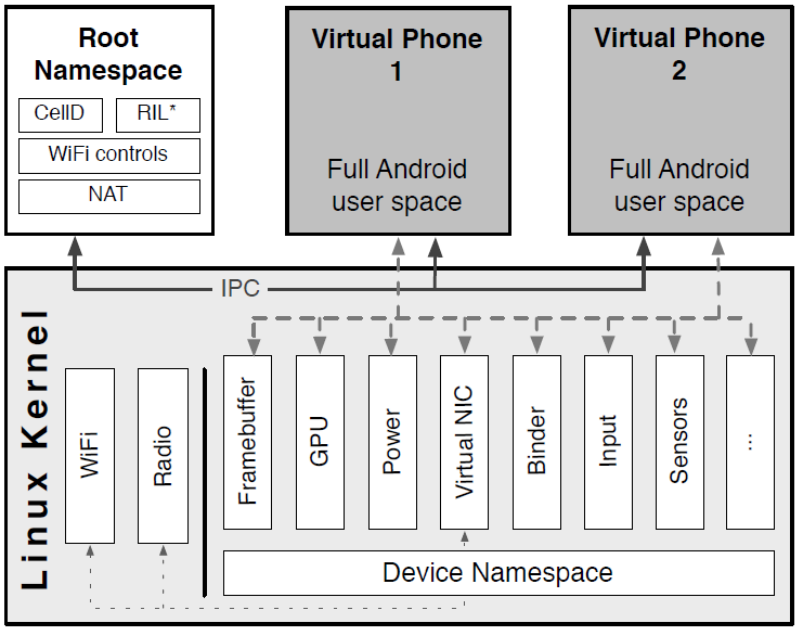
- **VPs are created and configured on a PC and downloaded to a phone via USB.**

- **Each VP can be configured to have different access rights for different devices.**
  - ✓ No access, shared access, or exclusive access



SYSTEM ARCHITECTURE

- Lightweight OS virtualization for virtual phones isolation
- Single OS Kernel
- Virtualizes identifiers, kernel interfaces and hardware resources
- Transparently remapping OS resource identifiers to virtual ones



*RIL: Vendor Radio Interface Layer library is loaded by CellID

Kernel-Level Device Virtualization

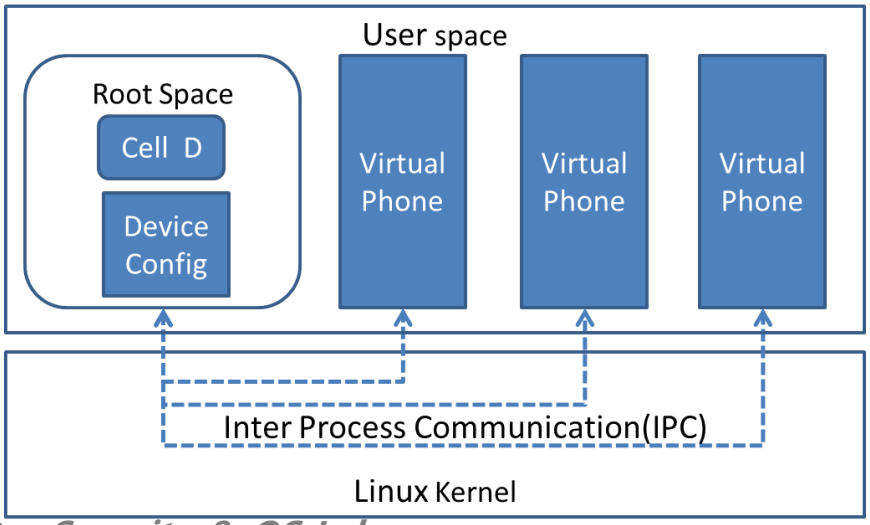
- **Device namespaces**
  - ✓ Hardware resource multiplexing and isolation

- **Callback functions**
  - ✓ These are called when a device namespace changes state.

- **Cells virtualizes existing kernel interfaces based on three methods of implementing device namespace functionality.**
  - ✓ The first method is to create a device driver wrapper using a new device driver for a virtual device.
  - ✓ The second method is to modify a device subsystem to be aware of device namespaces.
  - ✓ The third method is to modify a device driver to be aware of device namespaces.

User-Level Device Virtualization

- Name space proxy mechanisms
- Virtualize device configuration
- Kernel device namespaces export an interface to the root namespace through the /proc file system that is used to switch the foreground VP and set access permissions for devices.
- CellD also coordinates user space virtualization mechanisms such as the configuration of telephony and wireless networking

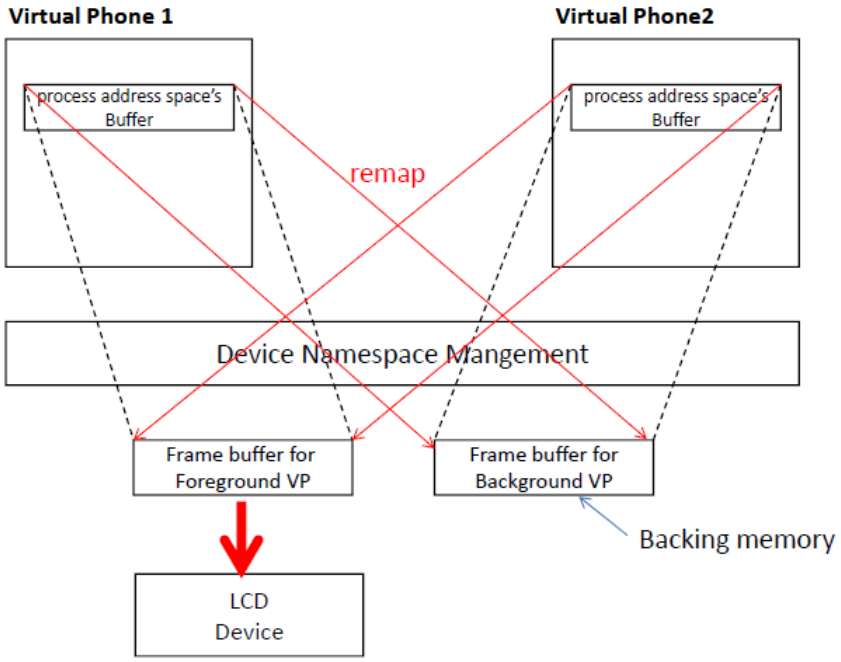


GRAPHICS

❖ Framebuffer

- ✓ To virtualize FB access in multiple VPs, Cells leverages the kernel-level device namespace and its foreground-background usage model in a new multiplexing FB device driver, **mux_fb**.

- ✓ **The foreground VP is given exclusive access to the screen memory and display hardware**



POWER MANAGEMENT

- **Early suspend subsystem**
  - ✓ The subsystem is an ordered callback interface

- **Frame buffer early suspend:**
  - ✓ Fbearlysuspend driver exports display device suspend and resume state into user space.
  - ✓ Block all processes using display while display powered off/ redraws screen when powered on.

- **Wake locks: Two states**
  - ✓ Active-locked
  - ✓ Inactive-unlocked

- **Wake locks can be created statically at compile time or dynamically by kernel drivers or user space.**

TELEPHONY

- **Radio stack virtualization**

- **RIL proxy**
  - ✓ CELLS own proxy RIL library by RILD
  - ✓ RIL library + CELLD=RIL proxy

- **Multiple phone numbers**
  - ✓ Pairing cells with VOIP service
  - ✓ Single digit scheme
  - ✓ Asterisk server

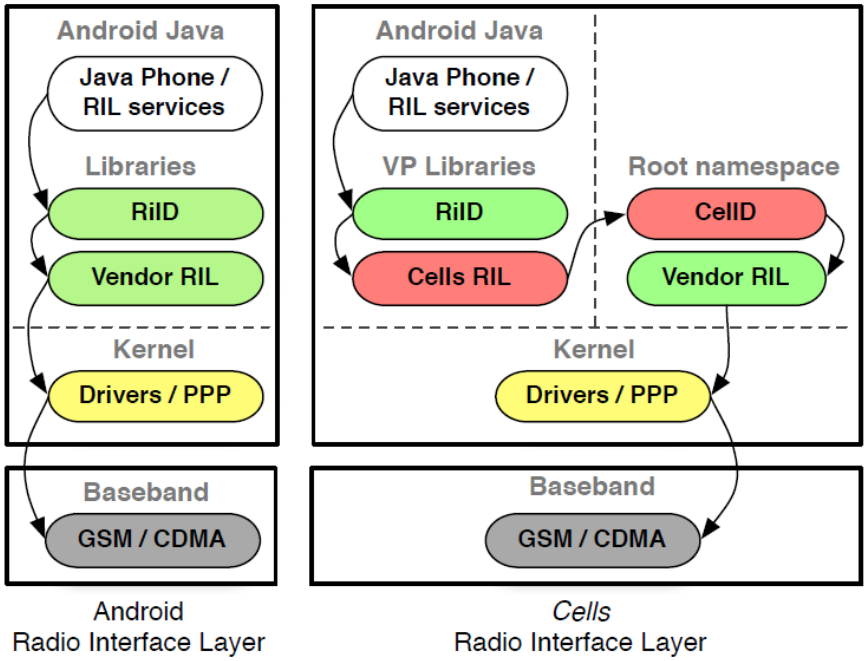
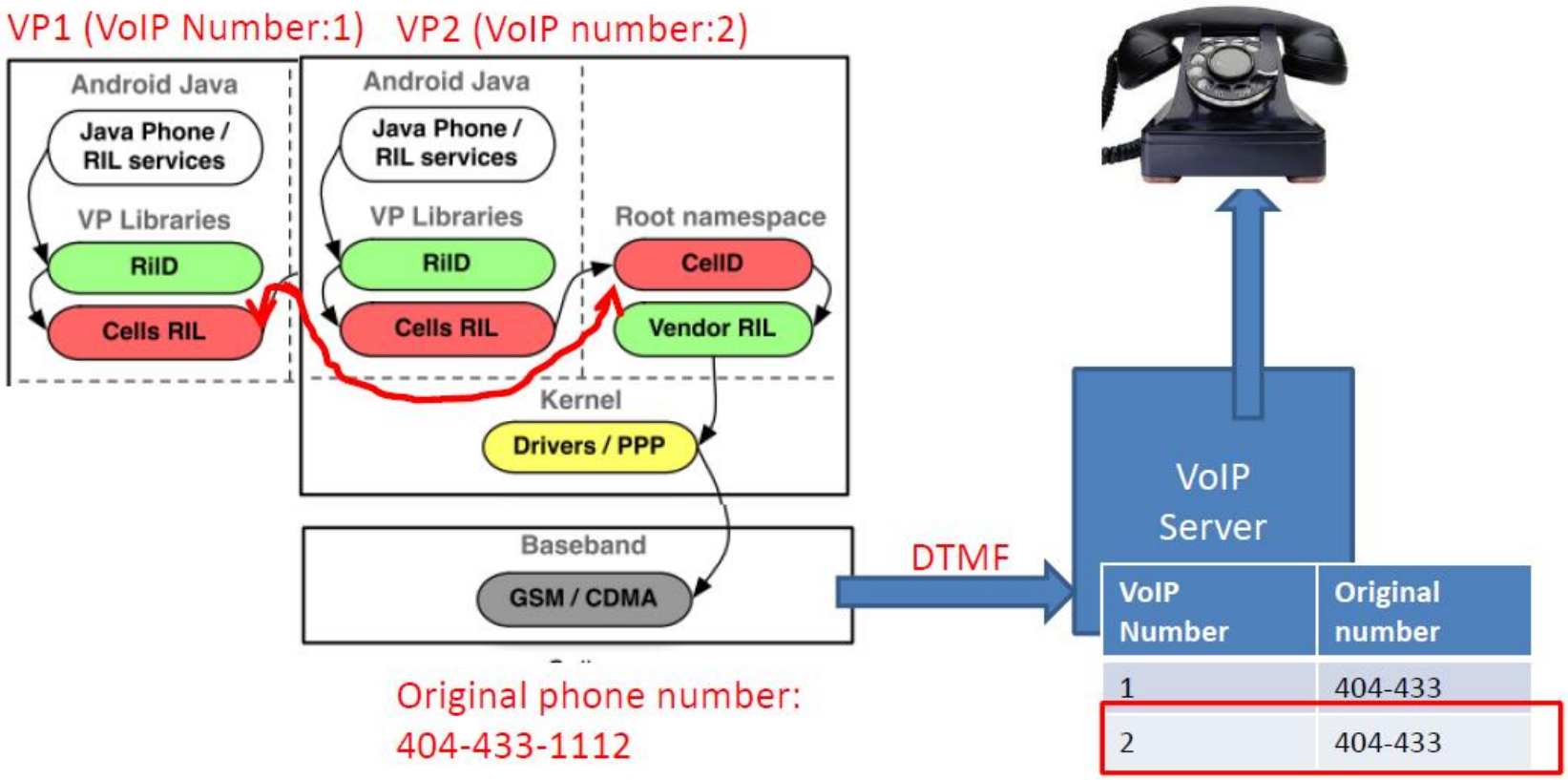
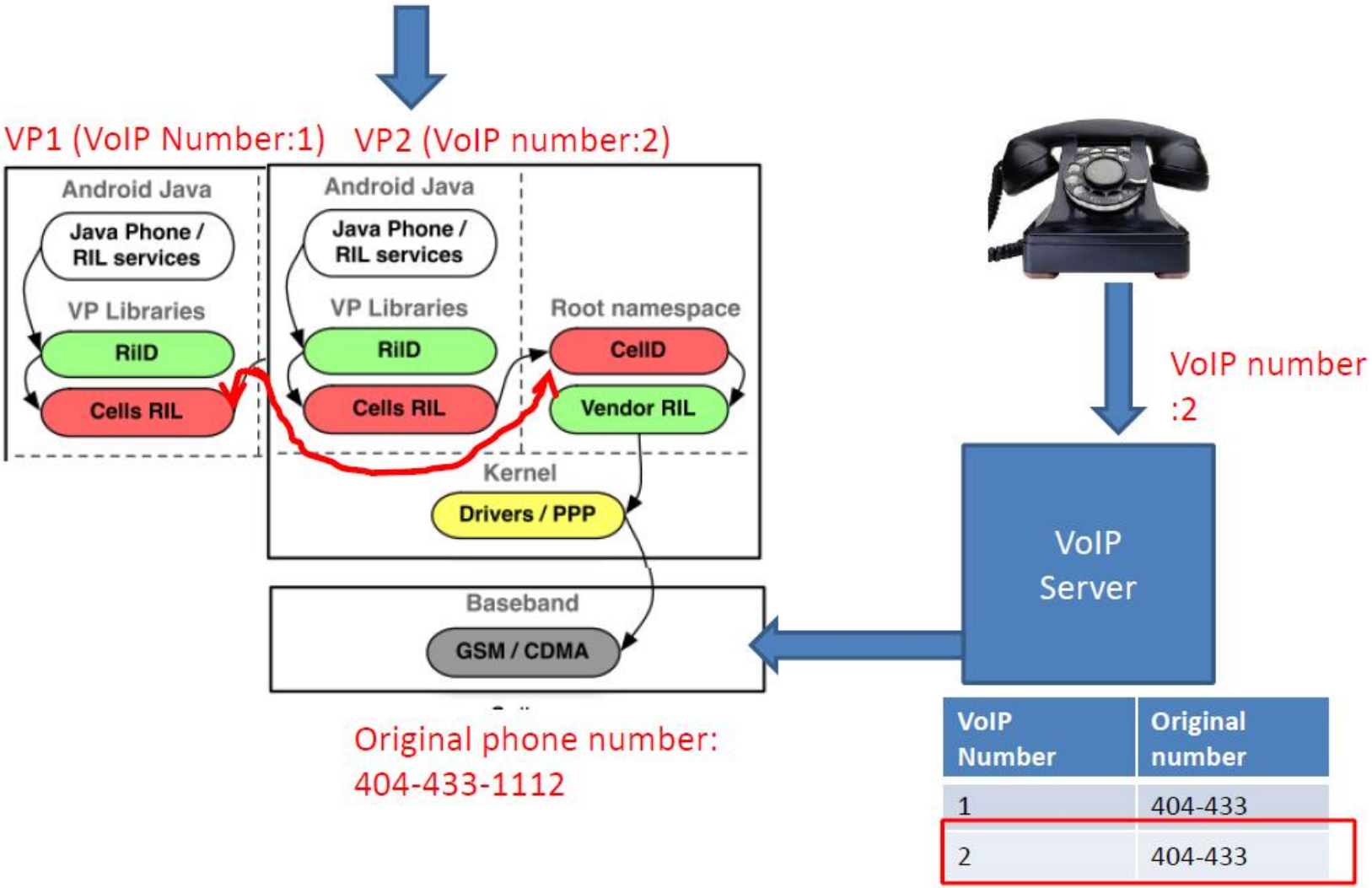


Figure 2: *Cells* Radio Interface Layer

Telephony/Outgoing Calls



Telephony/Incoming Calls



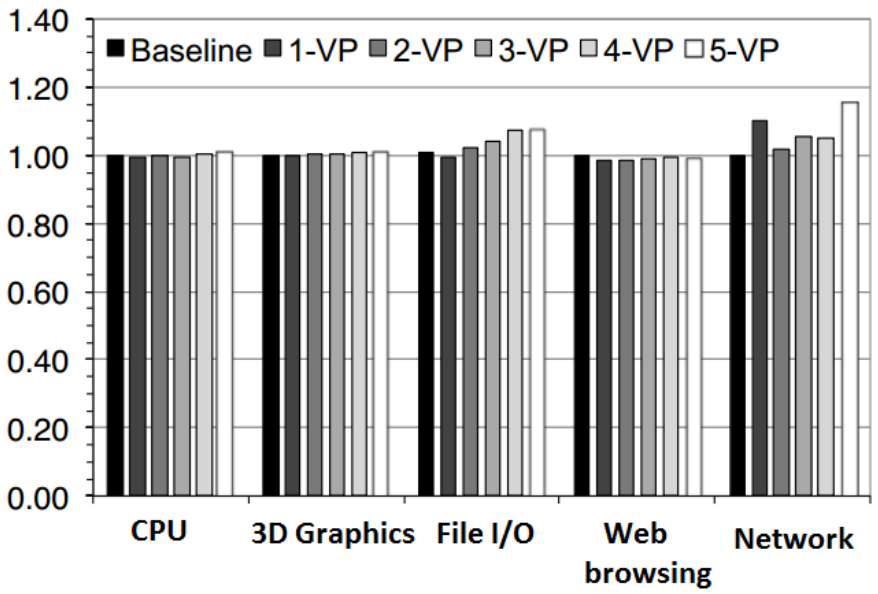
VoIP Number	Original number
1	404-433
2	404-433

EXPERIMENTAL RESULTS

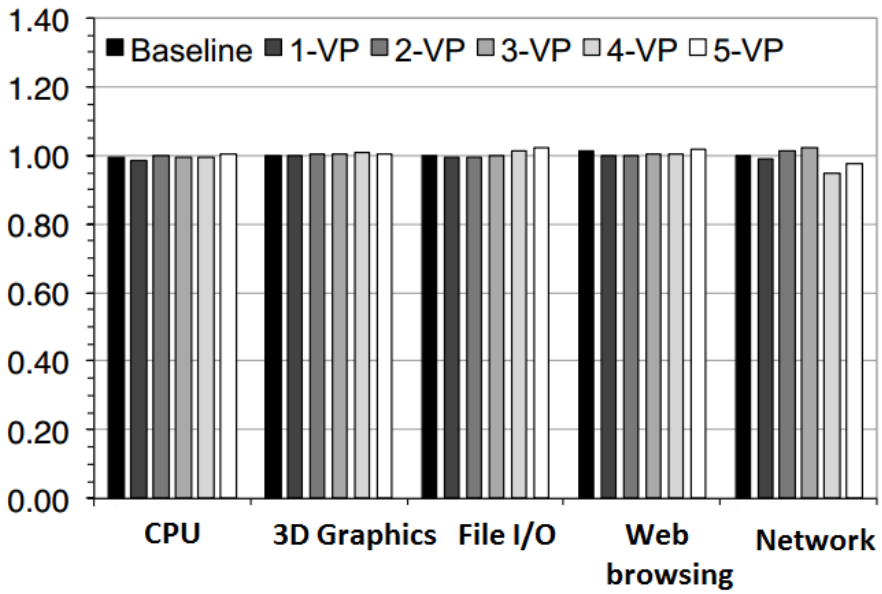
- **Experiment aim to measure:**
  - ✓ runtime overhead;
  - ✓ power consumption;
  - ✓ memory usage.

- **Testing configuration:**
  - ✓ Nexus1, Nexus S
  - ✓ Up to 5 Virtual phones;
  - ✓ With App and NoApp modes;
  - ✓ Network test downloading 400 Mb file using wi-fi;
  - ✓ Power consumption test 4 and 12 hours Active and Idle modes

Runtime overhead(Baseline)



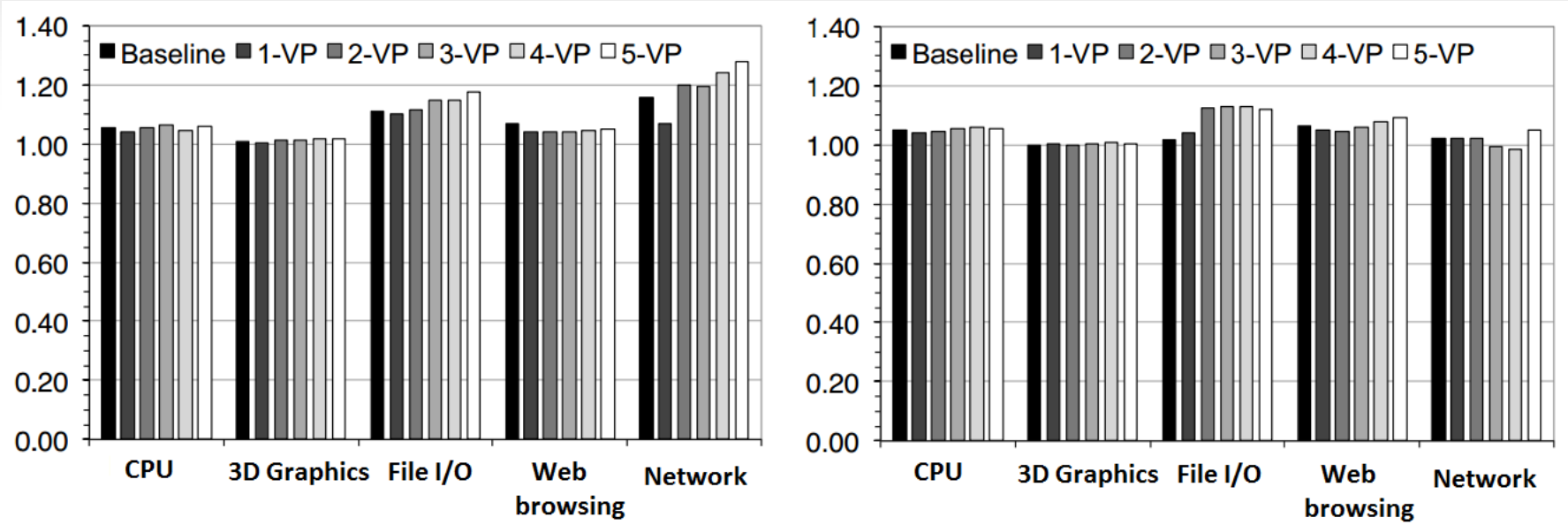
(a) Normalized Nexus 1 results



(b) Normalized Nexus S results

- **No App mode, Nexus 1 perform worse than Nexus S on Quadrant(File I/O) and Network tests(might be SD card usage). Overall performance overhead of Cells was within 7% in comparison to Baseline(original).**

Runtime overhead(With App)

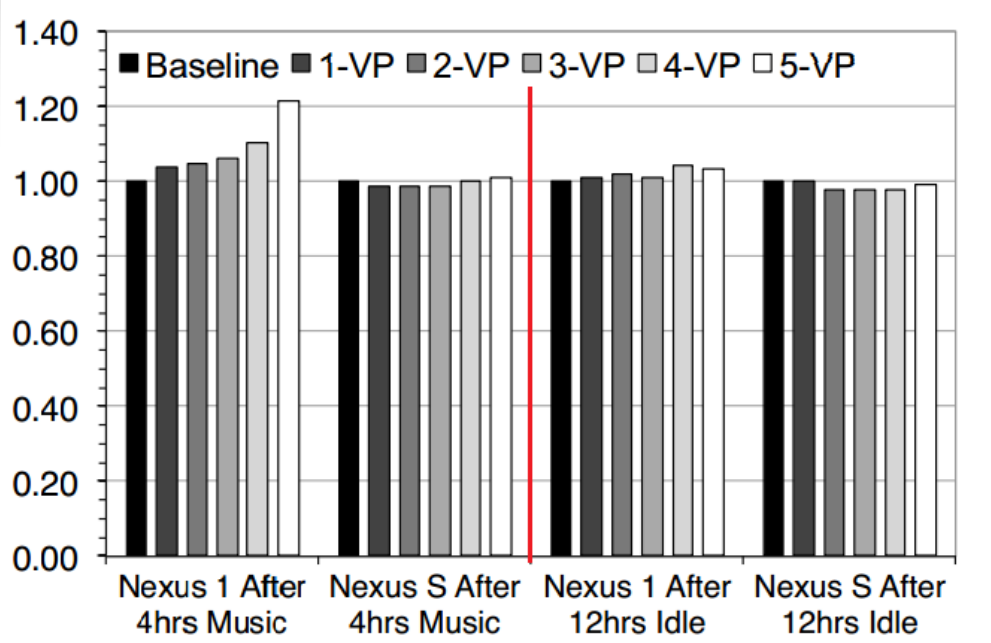


(c) Normalized Nexus 1 + music results

(d) Normalized Nexus S + music results

- In "With App" mode background music was continuously played during the test. Performance of Nexus S shown less than 10% overhead.

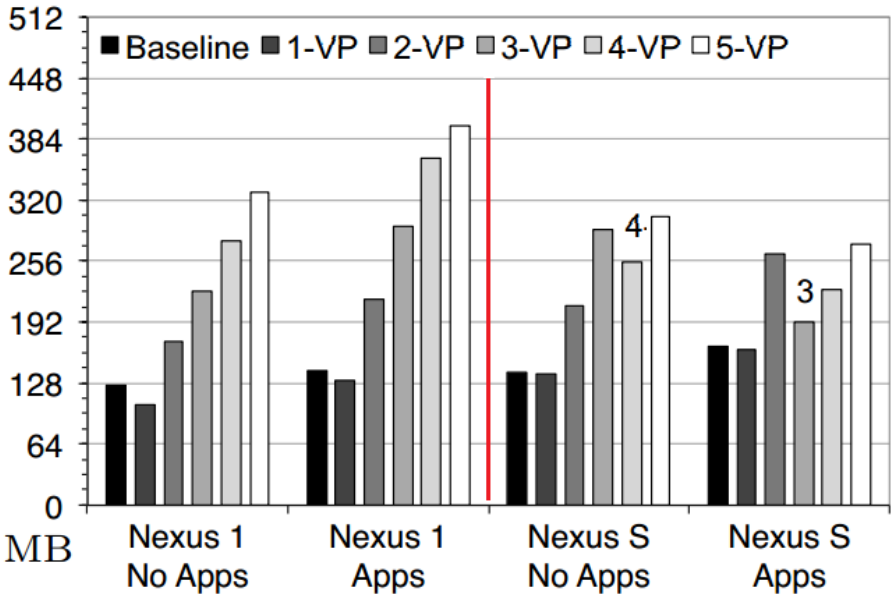
Power consumption



(e) Normalized battery capacity

- Power consumption in active mode 4 hours and 12 passive mode. Nexus 1 during playing the music shows gradual increase up to 20% overhead because of scheduling more processes. While on Nexus S power overhead was within 2%.

Memory usage



(f) Memory usage

- Memory test in "Apps mode" running applications such as Web browser, mail client, Calendar on all VPs. Nexus 1 show 20% better result because of Kernel Same page Merging algorithm(KSM).
- Nexus S use Low memory killer that free the memory during the execution 4th VP in "No Apps" mode and after 3rd VP in "Apps" mode.

Conclusion

- First complete OS virtualization for mobile devices.
  - ✓ Device namespaces + Foreground/Background = Complete virtualization
- Less overhead
- No visible performance variation for benchmark configurations

Reference

- S. Bhattiprolu, E. W. Biederman, S. Hallyn, andD. Lezcano. Virtual Servers and Checkpoint/Restart in Mainstream Linux. ACM SIGOPS Operating Systems Review, 42:104{113, July 2008.

- S. Osman, D. Subhraveti, G. Su, and J. Nieh. TheDesign and Implementation of Zap: a System for Migrating Computing Environments. In Proceedings of the 5th Symposium on Operating Systems Design and Implementation, Boston, MA, Dec. 2002.

감사합니다.