Graphical User Interface for Virtualized Mobile Handsets

Janis Danisevskis,
Michael Peter, Jan Nordholz,
Matthias Petschick, Julian Vetter

Security in Telecommunications
Technische Universität Berlin

MoST San José
May 21st, 2015

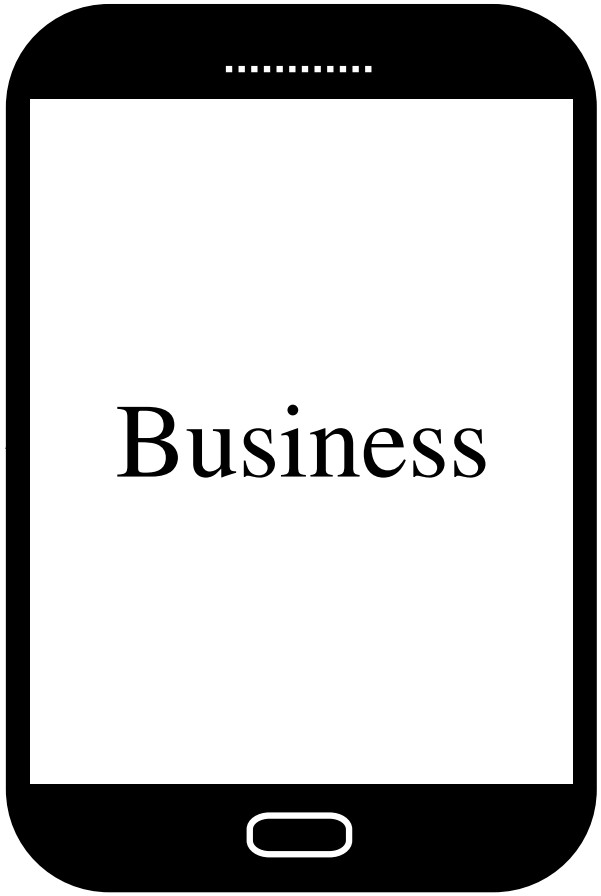
NEMESYS

SECT

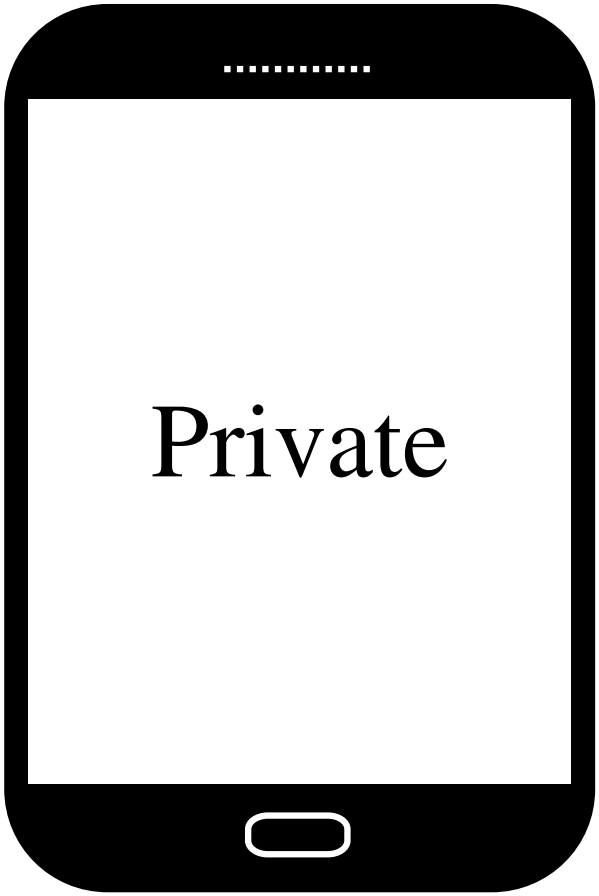
Bring You Own Device

Business Phone Policy (possibly)

- Restricted set of apps
- Restricted internet access (VPN/Firewall)
- Remote provisioning



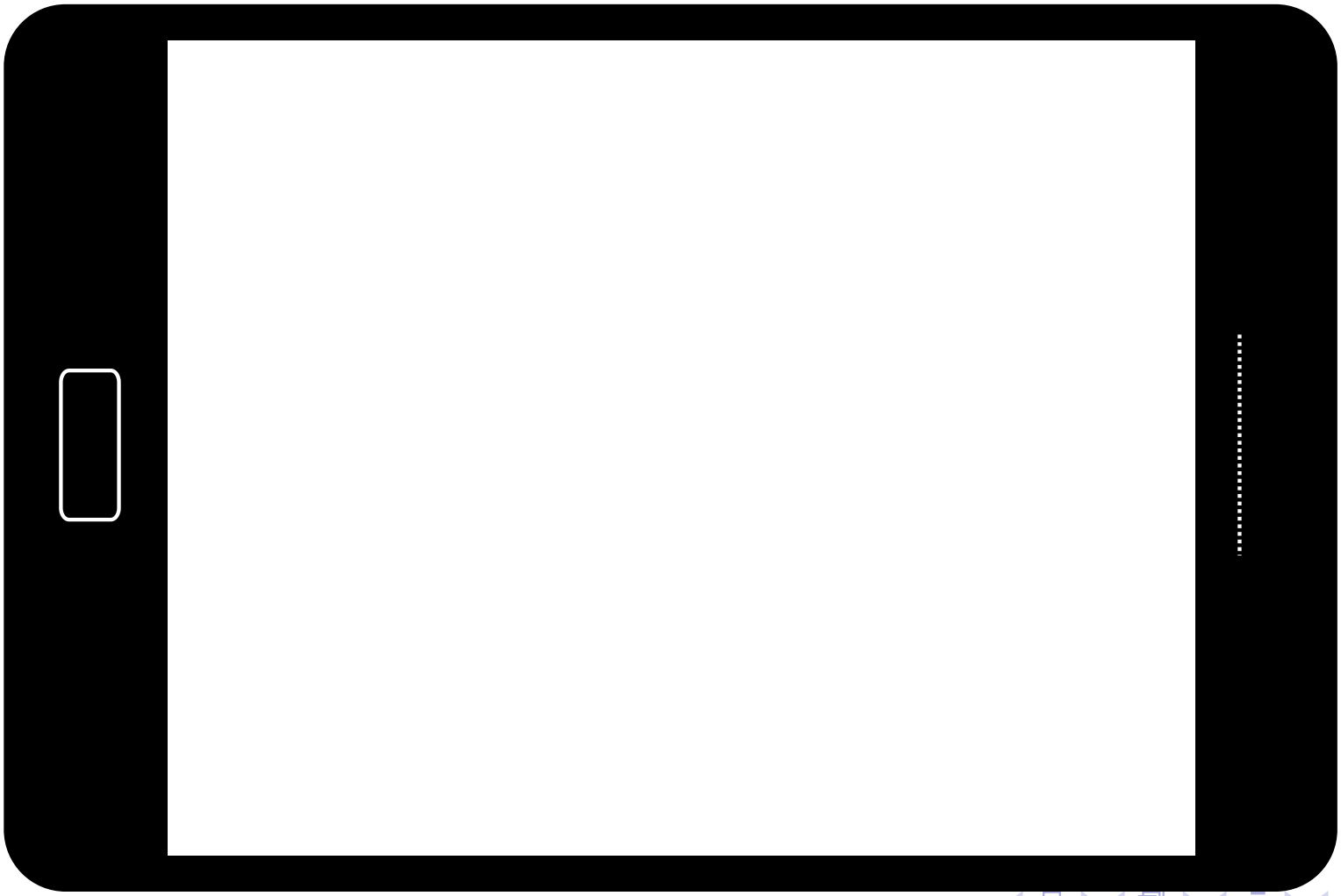
Bring You Own Device



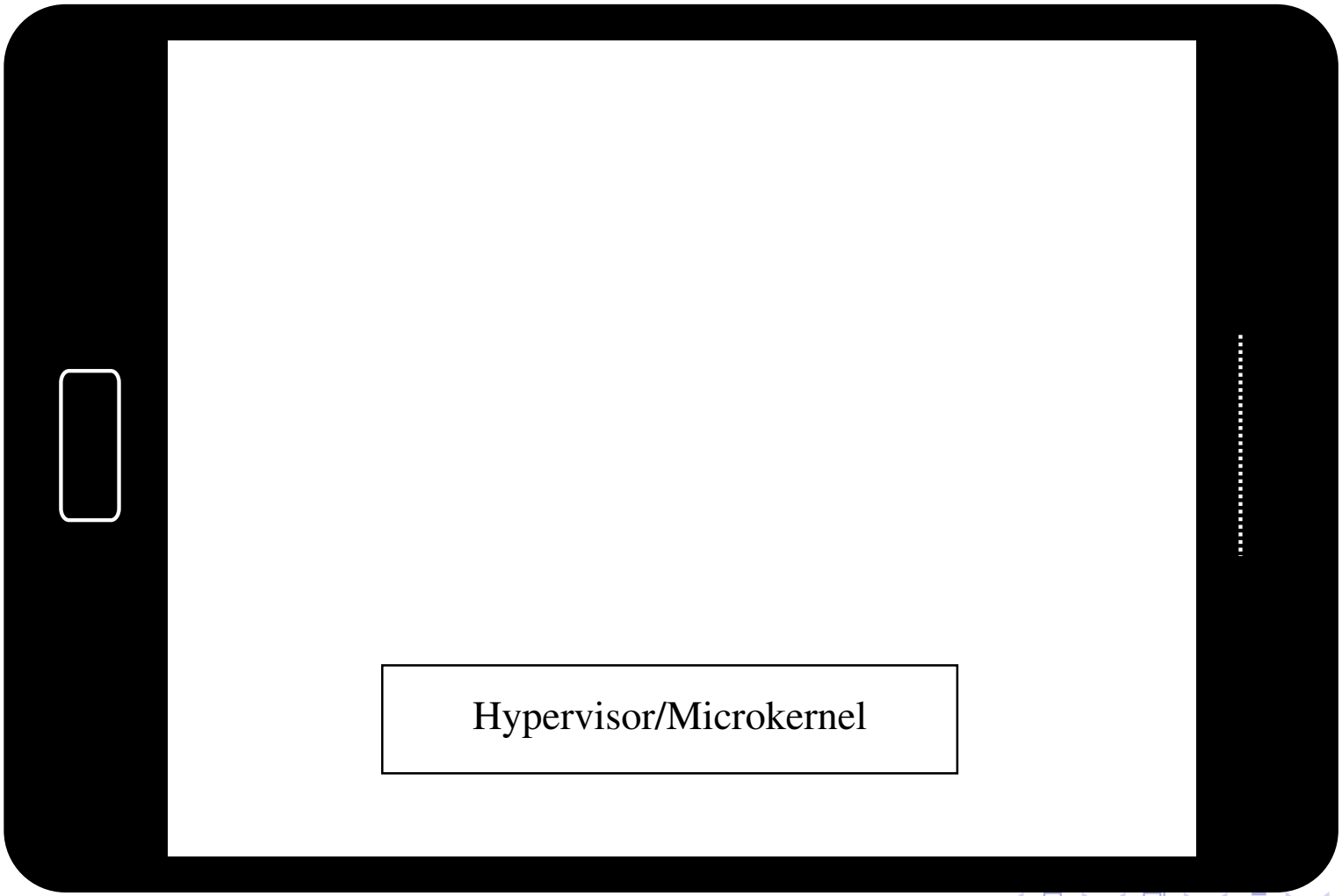
Private Phone Policy (likely)

This is my phone, so I do whatever I want. And, don't meddle with my stuff.

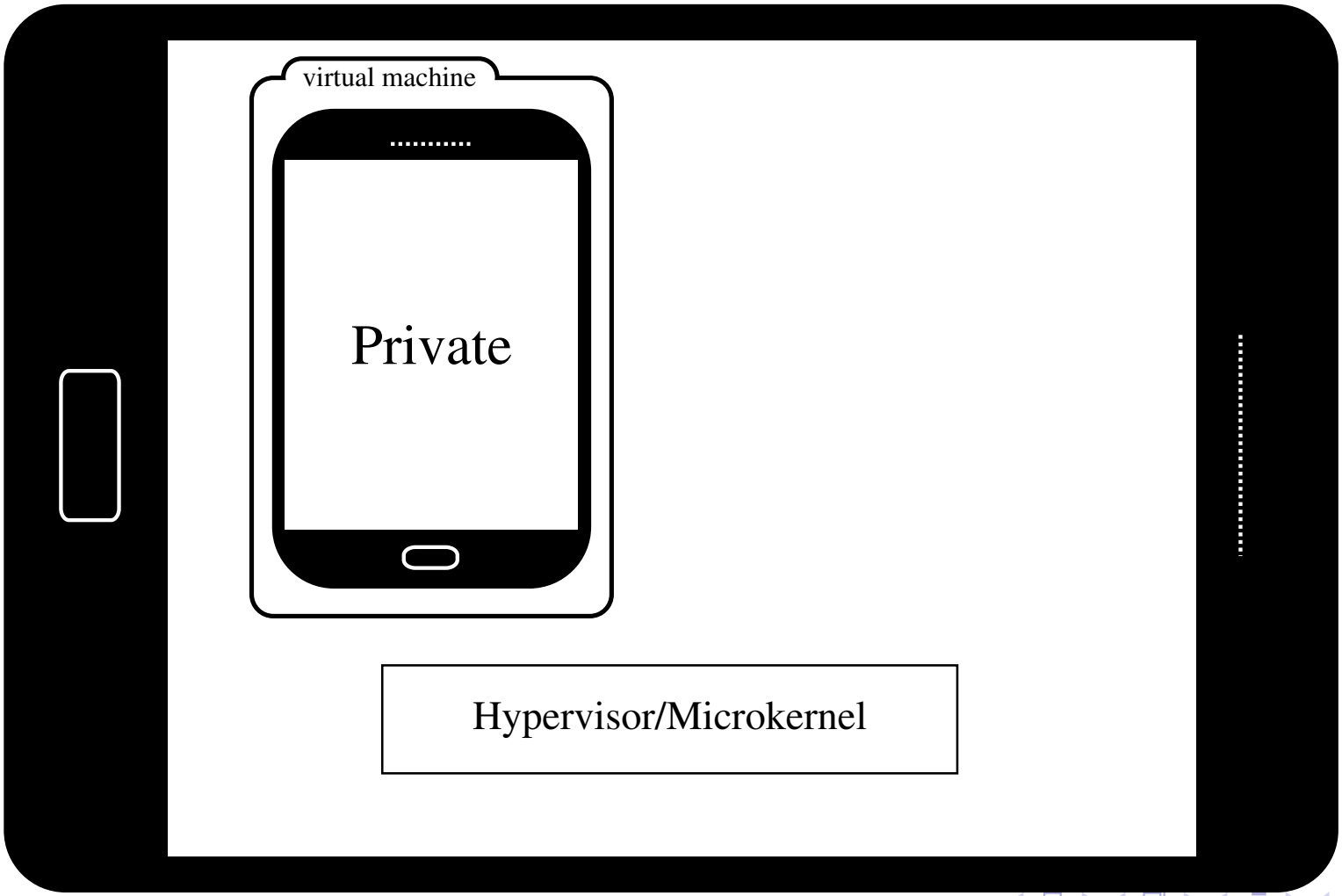
Our approach on BYOD



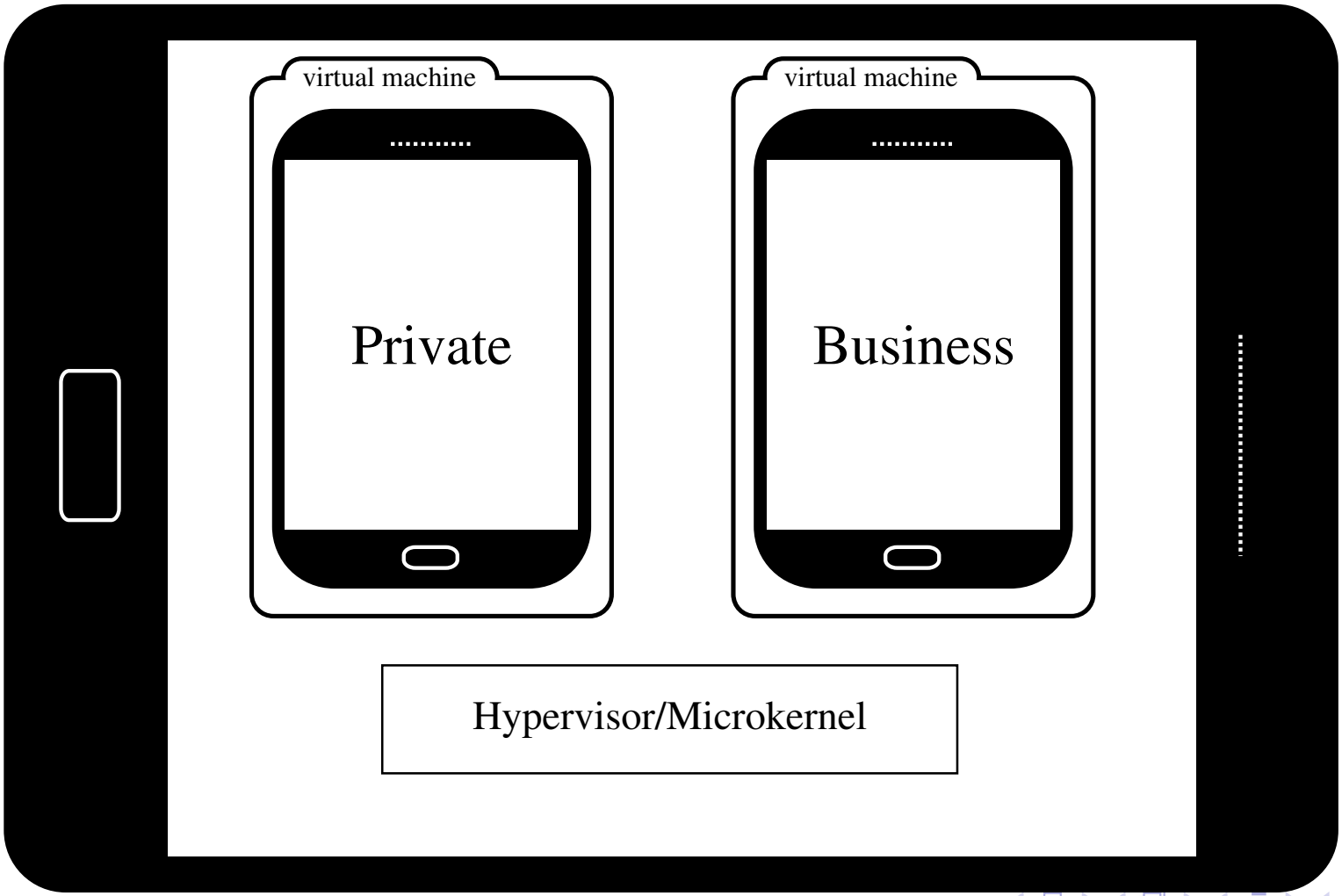
Our approach on BYOD



Our approach on BYOD



Our approach on BYOD

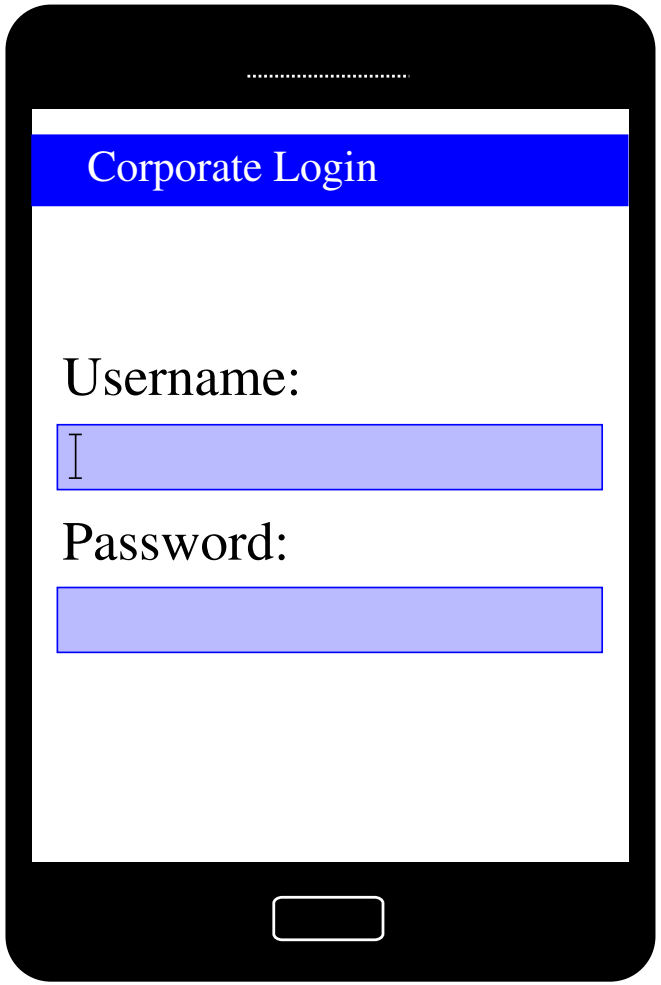


Challenges addressed by this work

Threat Model

Private side is under the control of an attacker

- Impersonation attacks
- Eavesdropping attacks
- Evasion of isolation

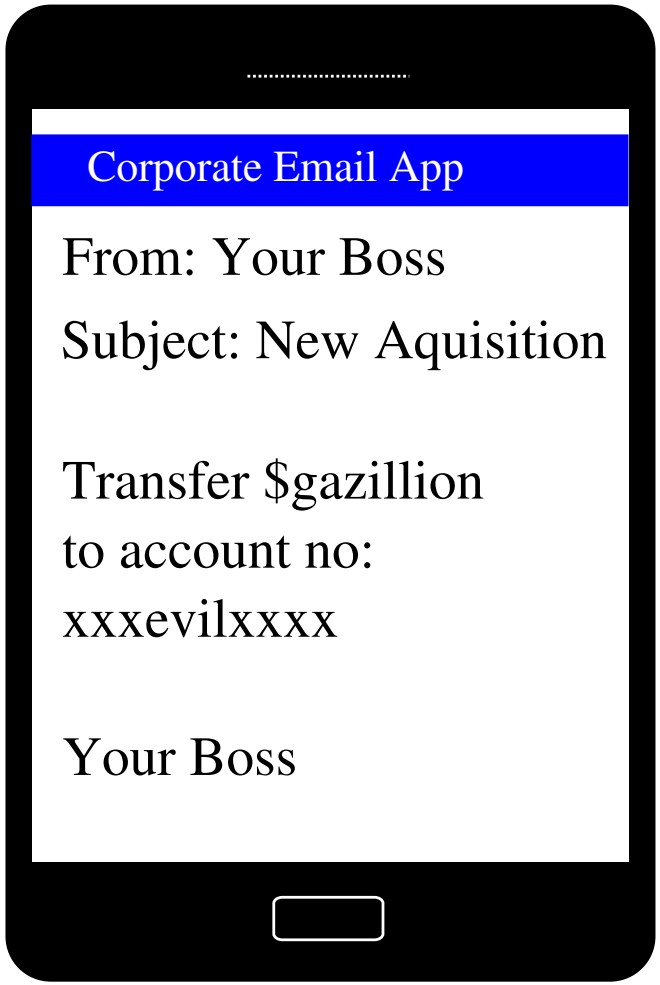


Challenges addressed by this work

Threat Model

Private side is under the control of an attacker

- Impersonation attacks
- Eavesdropping attacks
- Evasion of isolation



Challenges addressed by this work

Threat Model

Private side is under the control of an attacker

- Impersonation attacks
- Eavesdropping attacks
- Evasion of isolation

- Keylogging/
  Logging of touch events
- Spying on screen output

Challenges addressed by this work

Threat Model

Private side is under the control of an attacker

- Impersonation attacks
- Eavesdropping attacks
- Evasion of isolation

DMA devices can threaten isolation

[7] Cloudburst (2009)

[6] Dark Side of the Shader: Mobile GPU-Aided Malware Delivery (2013)

[3, 5, 4] “Fire in the (root) hole!” (2014)

Challenges addressed by this work

Threat Model

Private side is under the control of an attacker

- Impersonation attacks
- Eavesdropping attacks
- Evasion of isolation

Design Goals

- High graphics performance
- Low impact on CPU load
- Low impact on the TCB

Challenges addressed by this work

Threat Model

Private side is under the control of an attacker

- Impersonation attacks
- Eavesdropping attacks
- Evasion of isolation

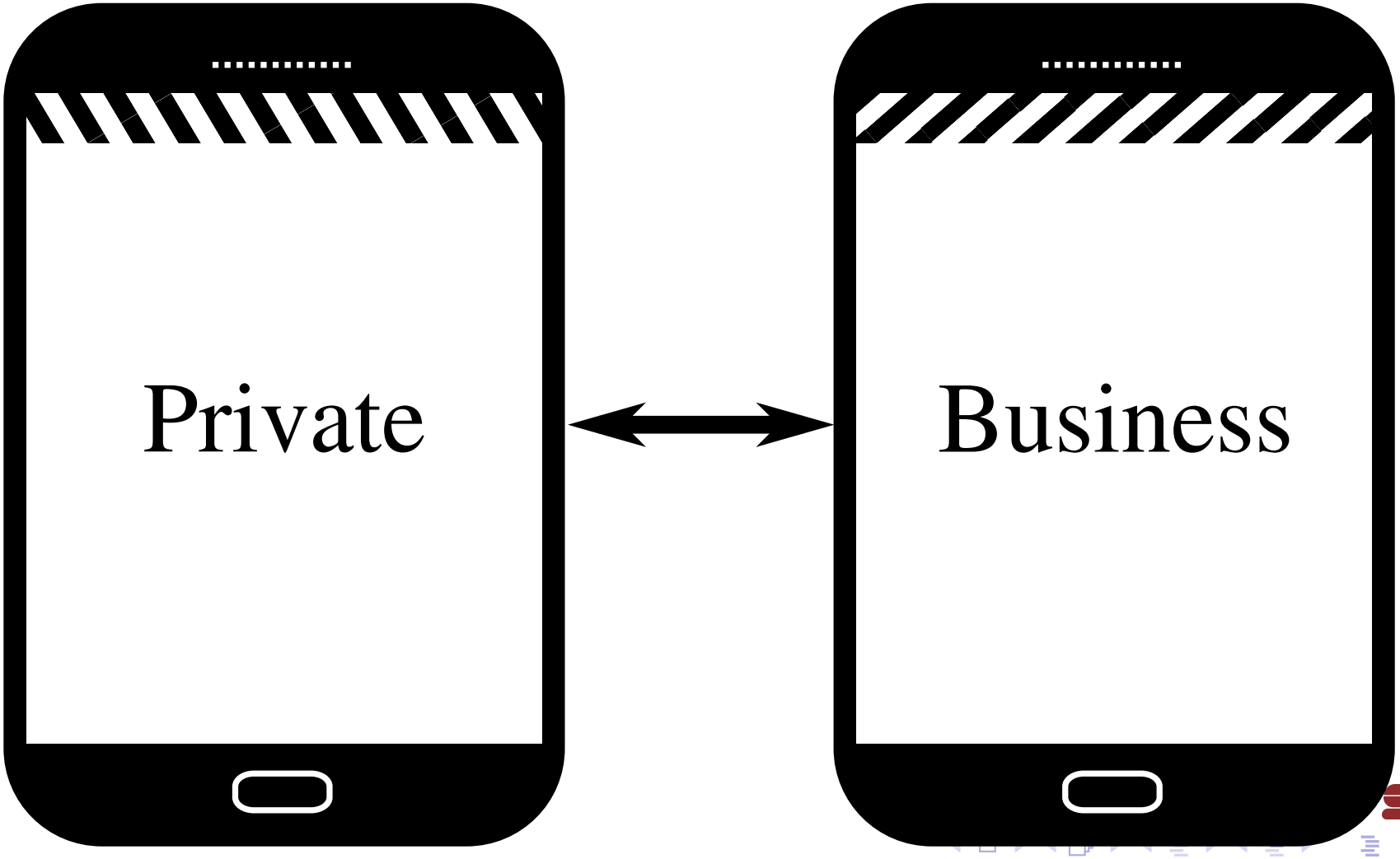
Design Goals

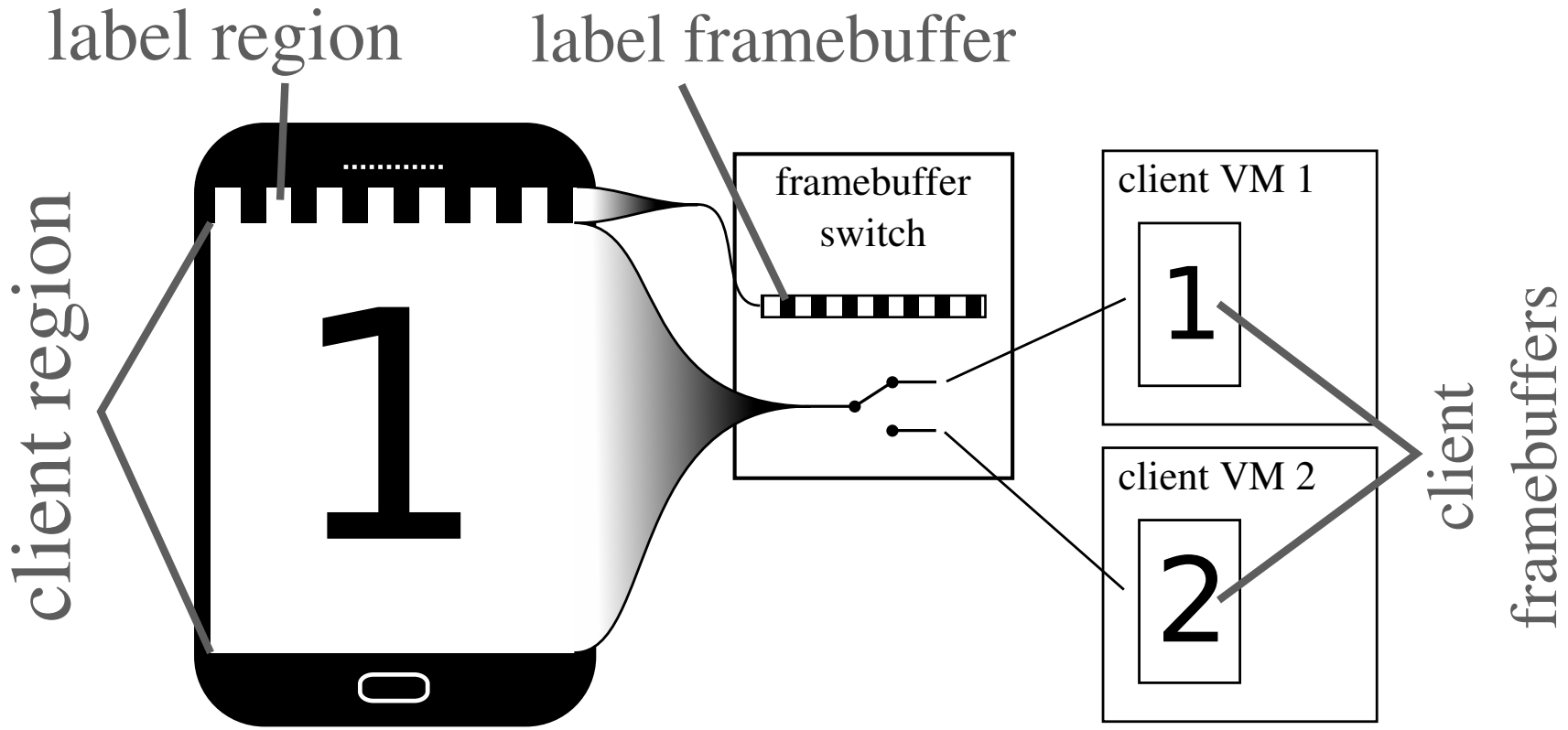
- High graphics performance
- Low impact on CPU load
- Low impact on the TCB

Design and Implementation

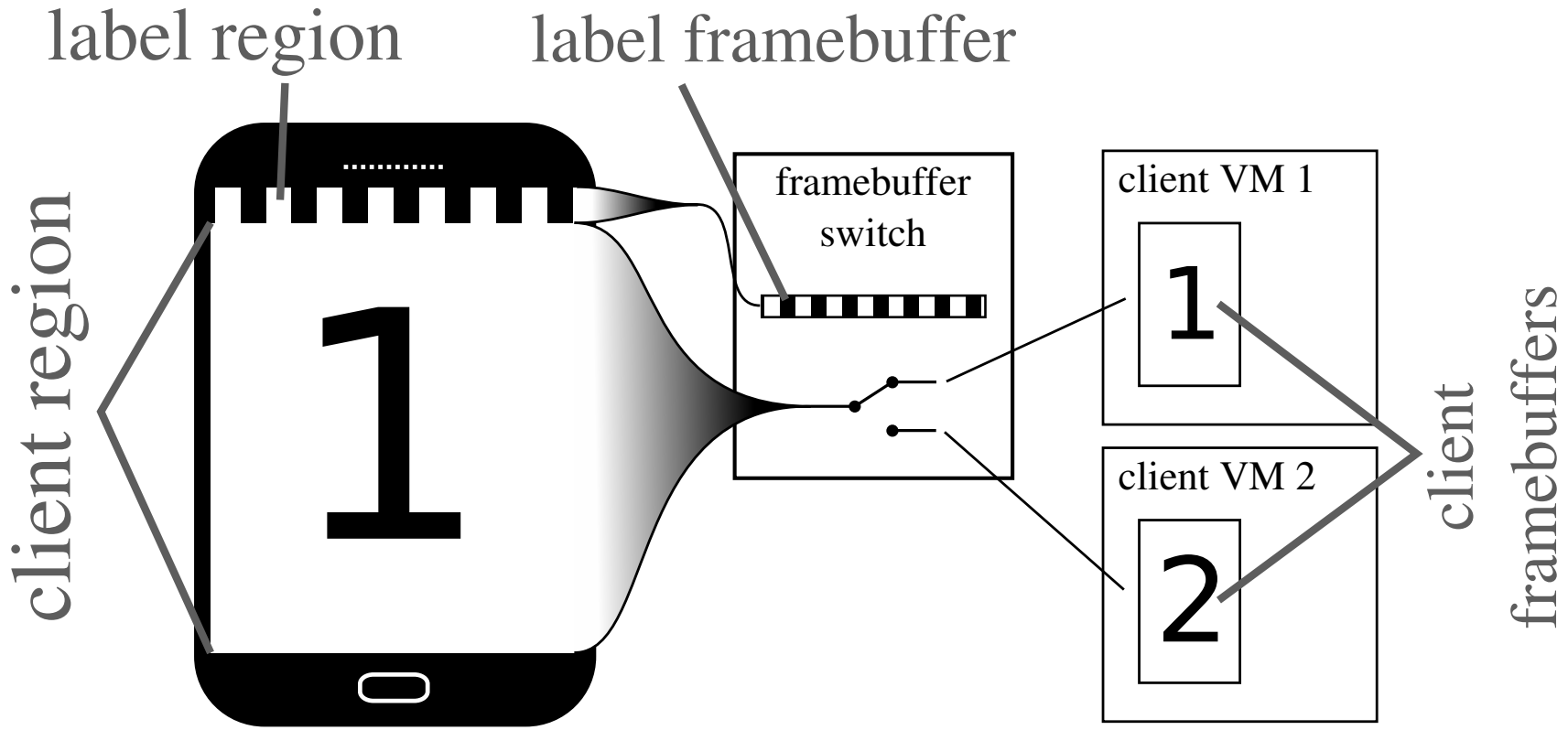
- Secure GUI (Trusted path)
- Secure Mobile GPU Virtualization

Output label

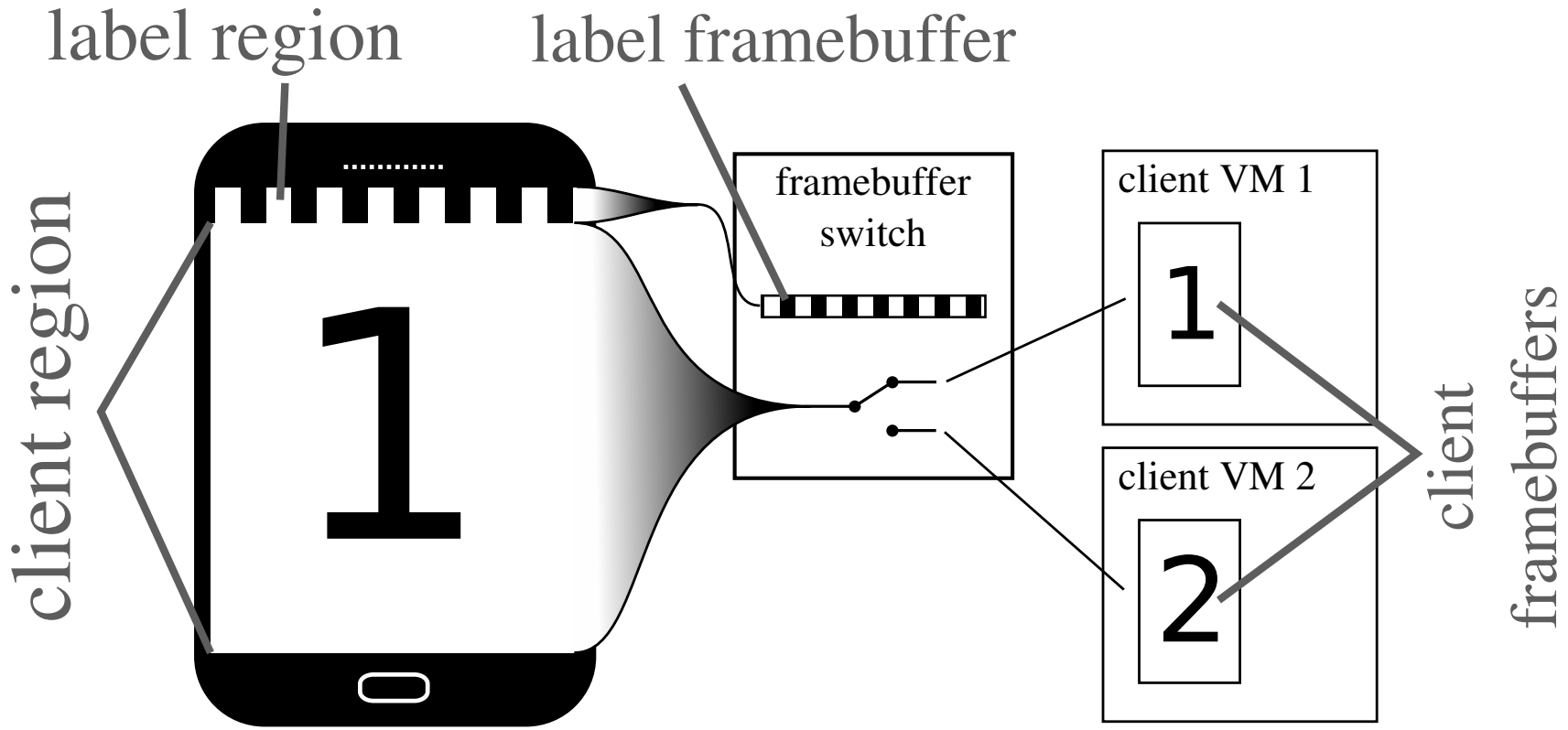




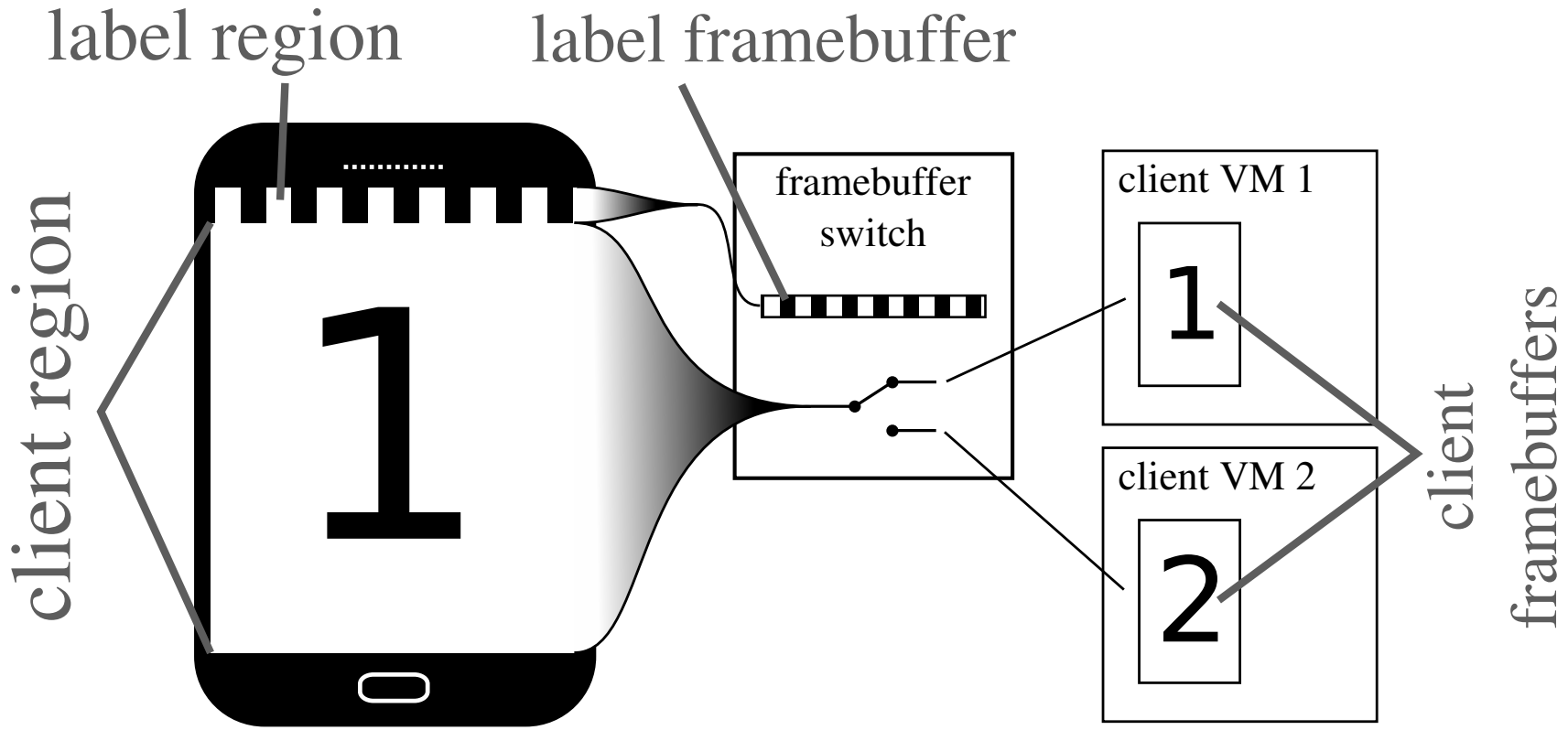
Screen is split into label region and client region



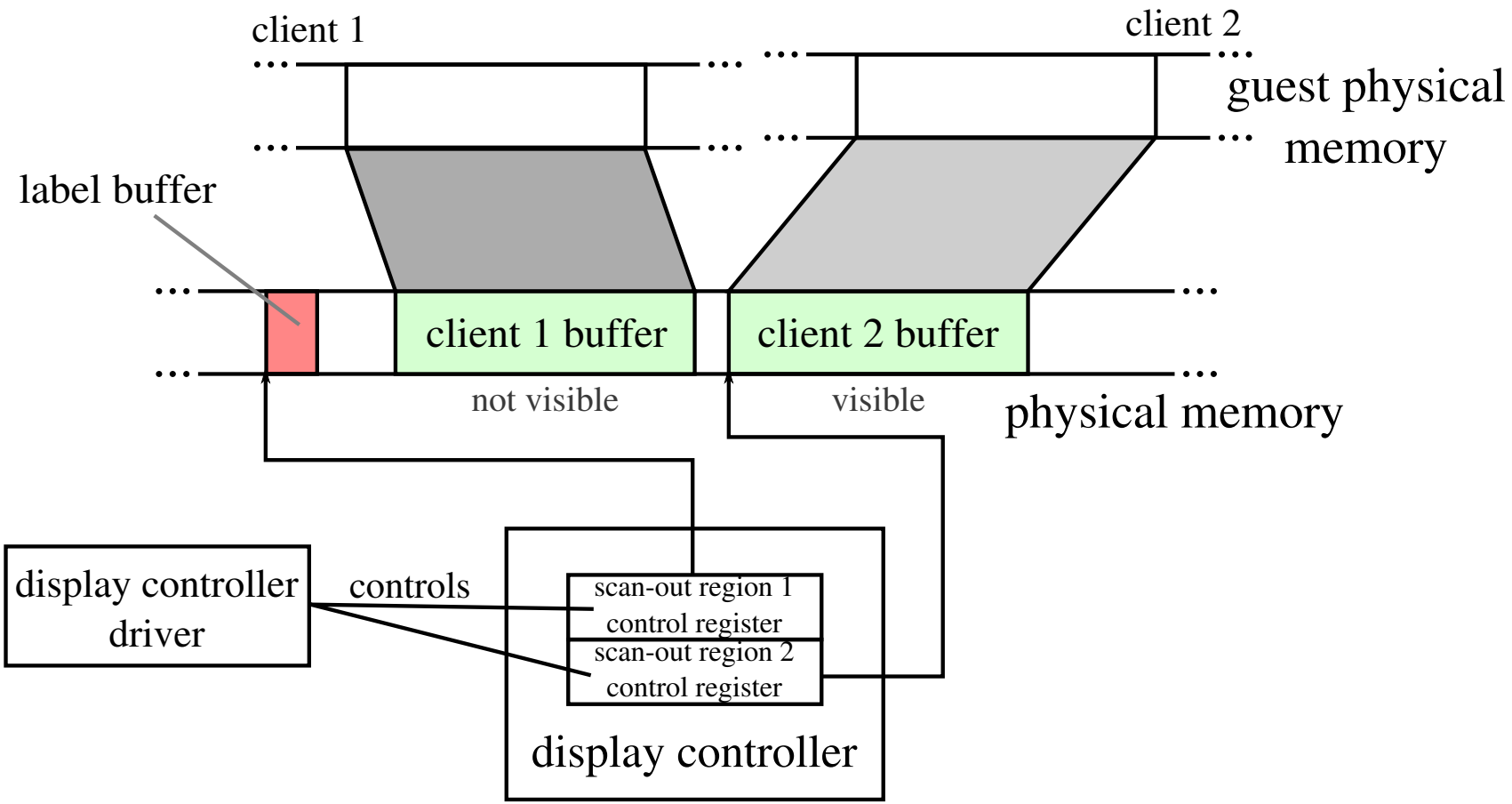
Client VMs have private framebuffers

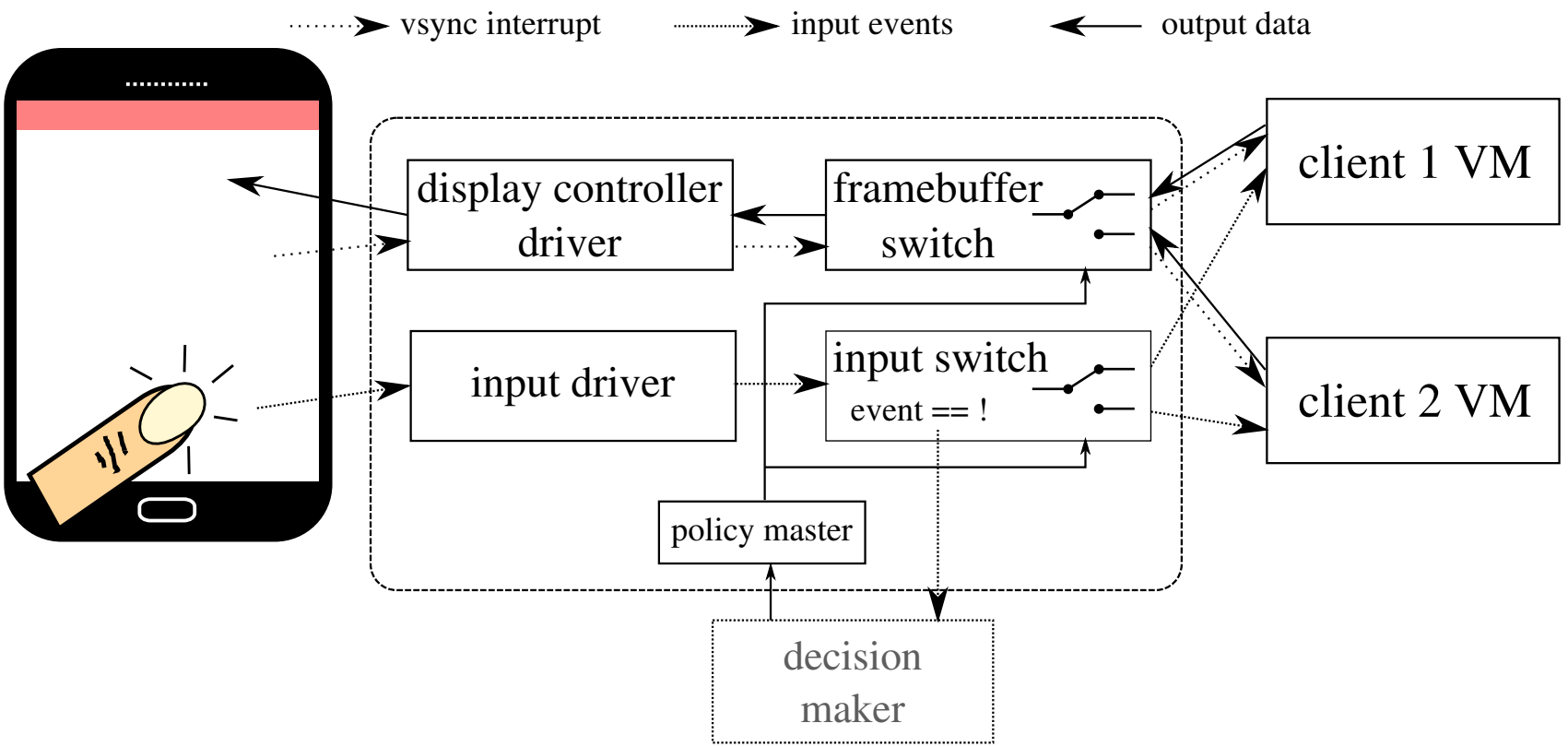


Label controlled by the switcher indicates output routing



Zero copy and composition in hardware

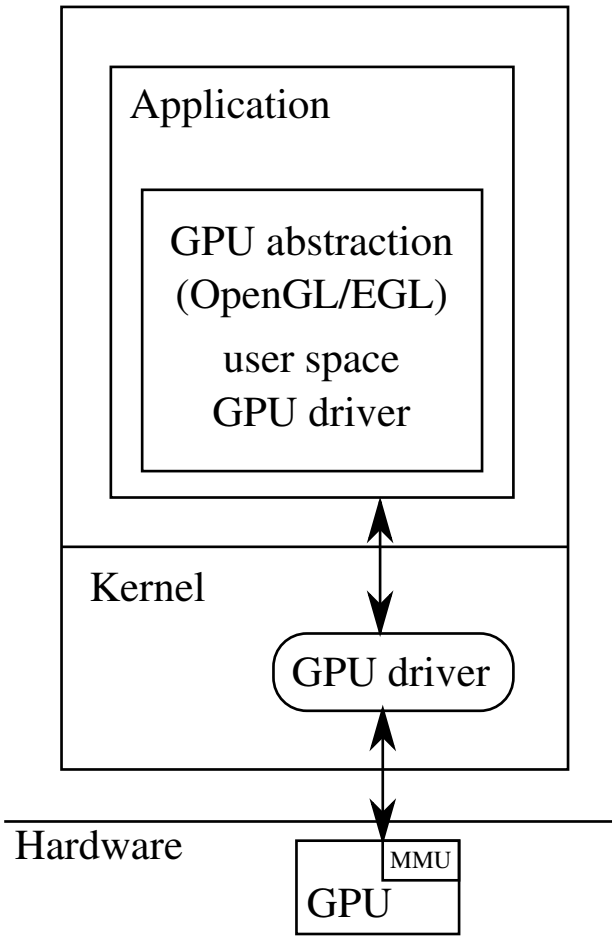

client 1
client 2
guest physical memory
label buffer
client 1 buffer
client 2 buffer
not visible
visible
physical memory
display controller driver
controls
scan-out region 1 control register
scan-out region 2 control register
display controller


vsync interrupt
input events
output data
display controller driver
framebuffer switch
client 1 VM
input driver
input switch
event == !
client 2 VM
policy master
decision maker

Summary: Secure GUI

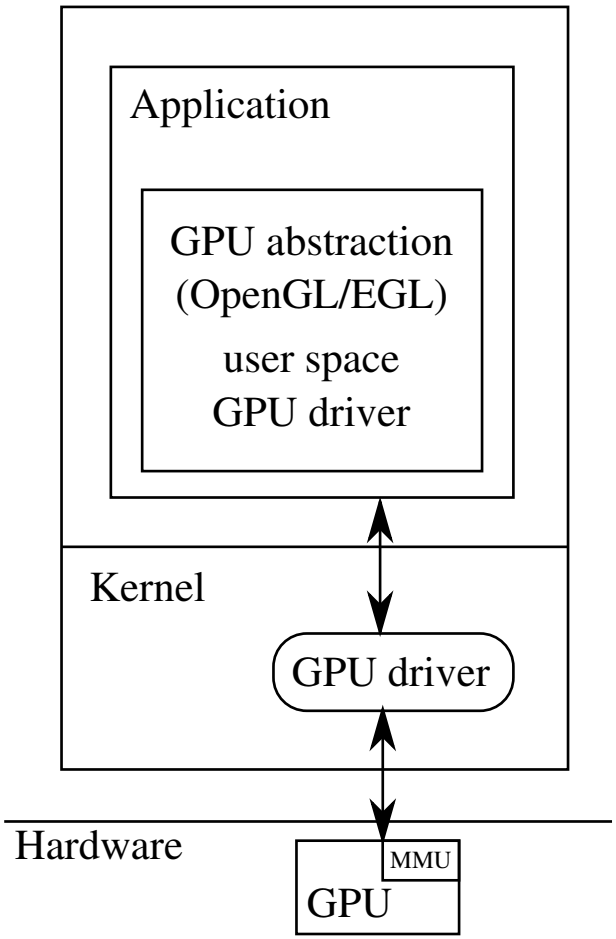
- Unforgeable labels
  $\rightarrow$ prevents impersonation
- Private framebuffers and exclusive input routing
  $\rightarrow$ prevent eavesdropping
- Zero copy with hardware overlays
  $\rightarrow$ low CPU load and low complexity

Mobile GPU Driver Stack



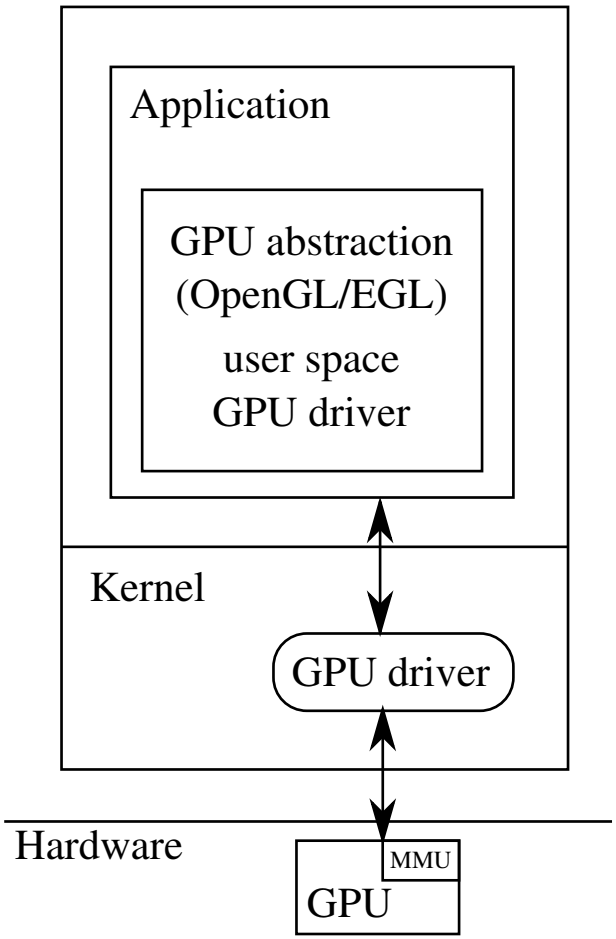
- User-space driver
  - Provides: OpenGL/EGL abstraction
  - Comprises: shader compiler, linker, ...
- Kernel-space driver
  - Schedules rendering tasks
  - Protects memory

Mobile GPU Driver Stack

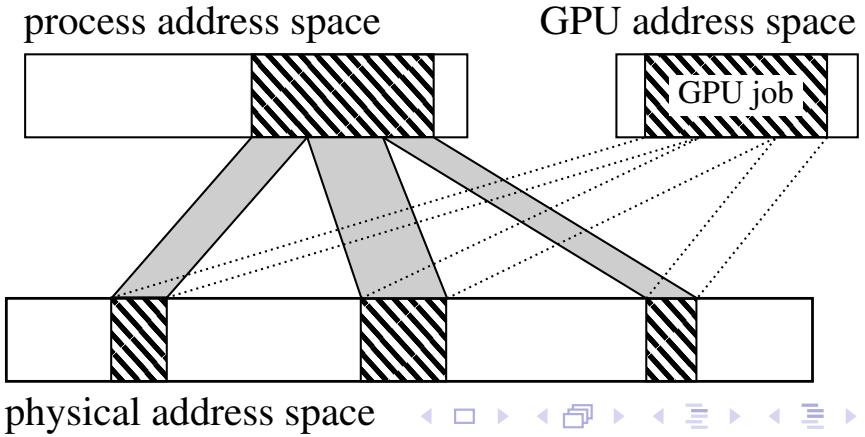


- User-space driver
  - Provides: OpenGL/EGL abstraction
  - Comprises: shader compiler, linker, ...
- Kernel-space driver
  - Schedules rendering tasks
  - Protects memory

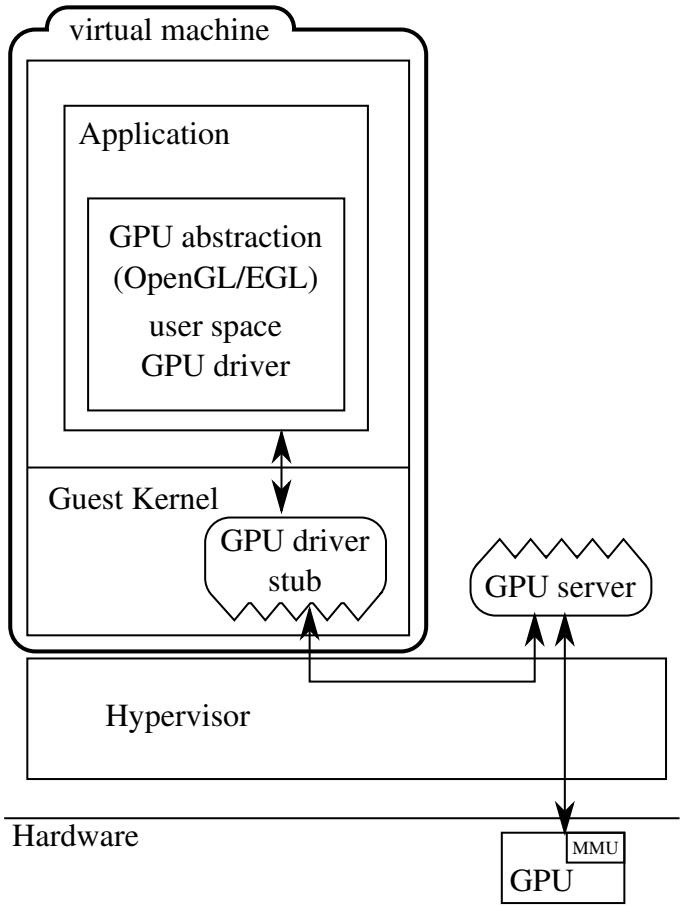
Mobile GPU Driver Stack



- User-space driver
  - Provides: OpenGL/EGL abstraction
  - Comprises: shader compiler, linker, . . .
- Kernel-space driver
  - Schedules rendering tasks
  - Protects memory

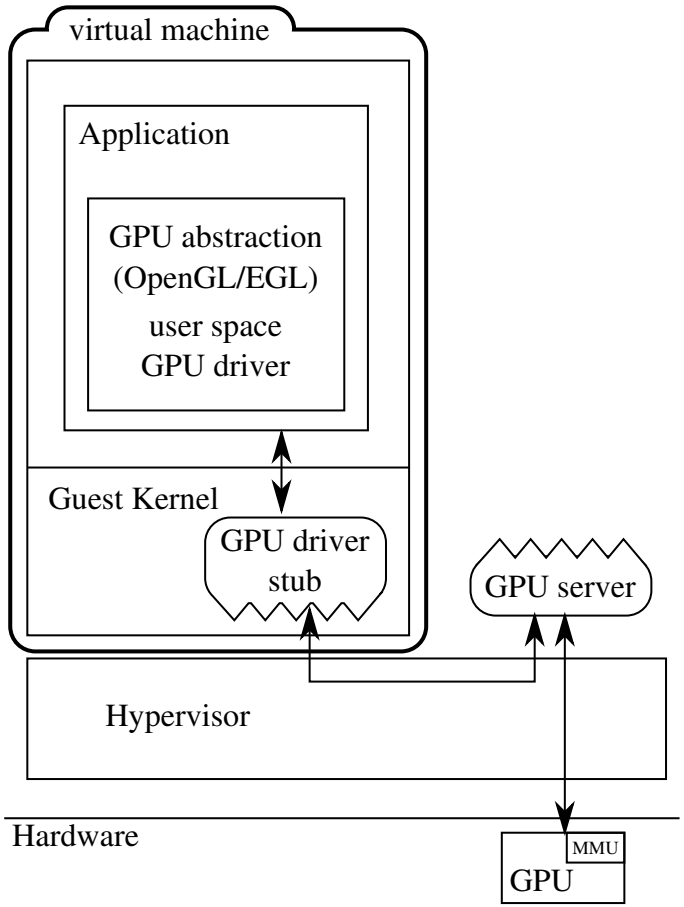


Mobile GPU Driver Stack (paravirtualized)



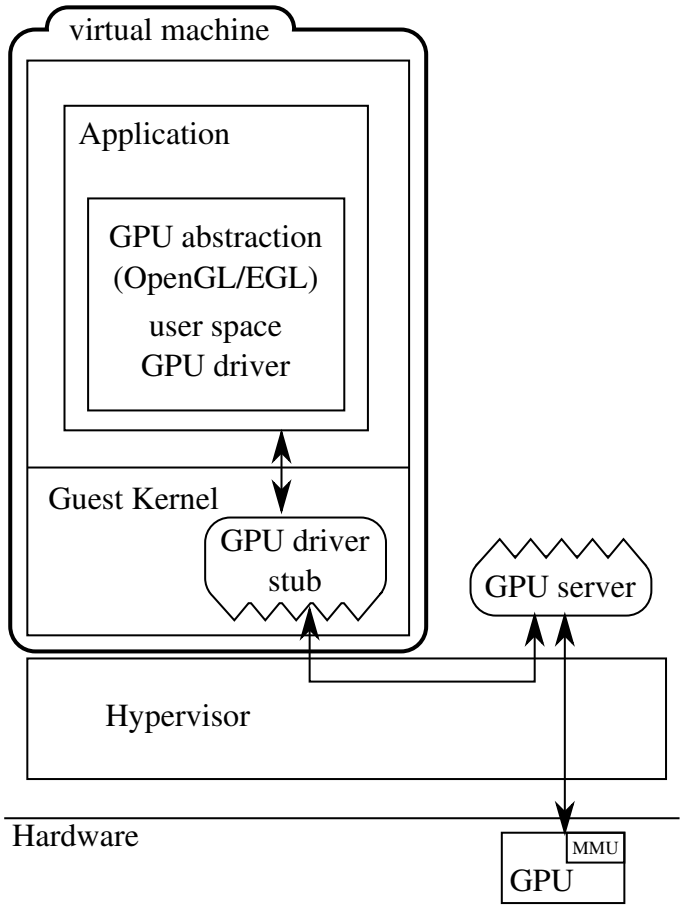
- User-space driver unmodified
- User-kernel interface unmodified
- Custom protocol between GPU driver stub and GPU server
  - No forwarding of high bandwidth data, such as textures, attribute lists, or shader programs
  - Forwards job requests to the GPU server (and job completion notifications to the client)
  - Forwards mapping requests to the GPU server

Mobile GPU Driver Stack (paravirtualized)



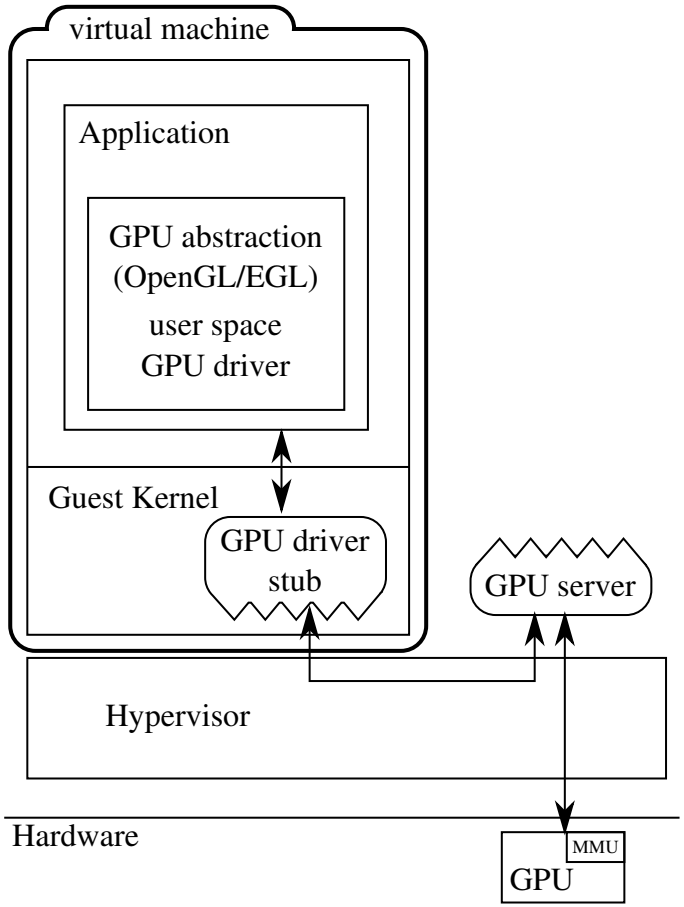
- User-space driver unmodified
- User-kernel interface unmodified
- Custom protocol between GPU driver stub and GPU server
  - No forwarding of high bandwidth data, such as textures, attribute lists, or shader programs
  - Forwards job requests to the GPU server (and job completion notifications to the client)
  - Forwards mapping requests to the GPU server

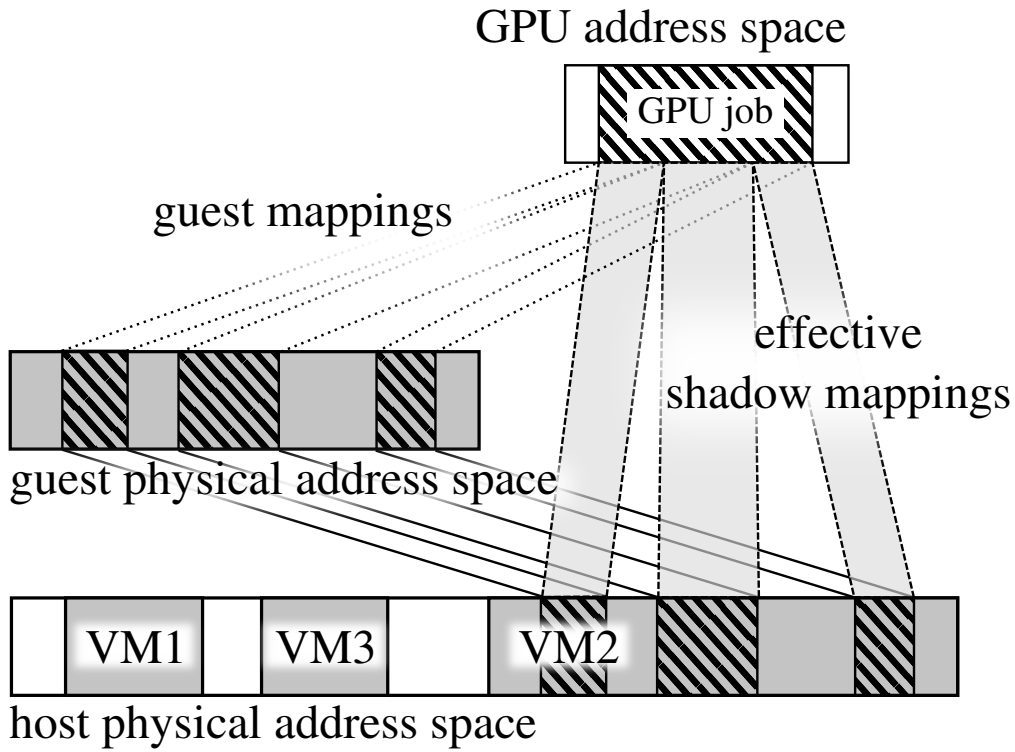
Mobile GPU Driver Stack (paravirtualized)



- User-space driver unmodified
- User-kernel interface unmodified
- Custom protocol between GPU driver stub and GPU server
  - No forwarding of high bandwidth data, such as textures, attribute lists, or shader programs
  - Forwards job requests to the GPU server (and job completion notifications to the client)
  - Forwards mapping requests to the GPU server

Mobile GPU Driver Stack (paravirtualized)





Prototype

Hardware

Samsung Galaxy SIII

- Exynos4412 SoC
- 4 $\times$ ARM Cortex A9 @ 1.4 GHz
- ARM Mali 400 MP4 GPU

Software

- Fiasco.OC (based on rev. 38)
- L4Re (based on rev. 38)
- L4Linux (based on Linux 3.0.101)
- Cyanogenmod CM-10.1.3

TCB impact

Module	**SLOC**[1]
GPU-RG[2]	2,679
display driver	2,382
framebuffer switch	548
input driver	710
input switch	539
total	6,858

[1] Source lines of code measured with David A. Wheeler's "SLOCCount"
[2] GPU-RG: Name of our GPU-server (RG is for resource governor)

Performance evaluation — experiments

Native

Cyanogenmod on Linux on bare metal

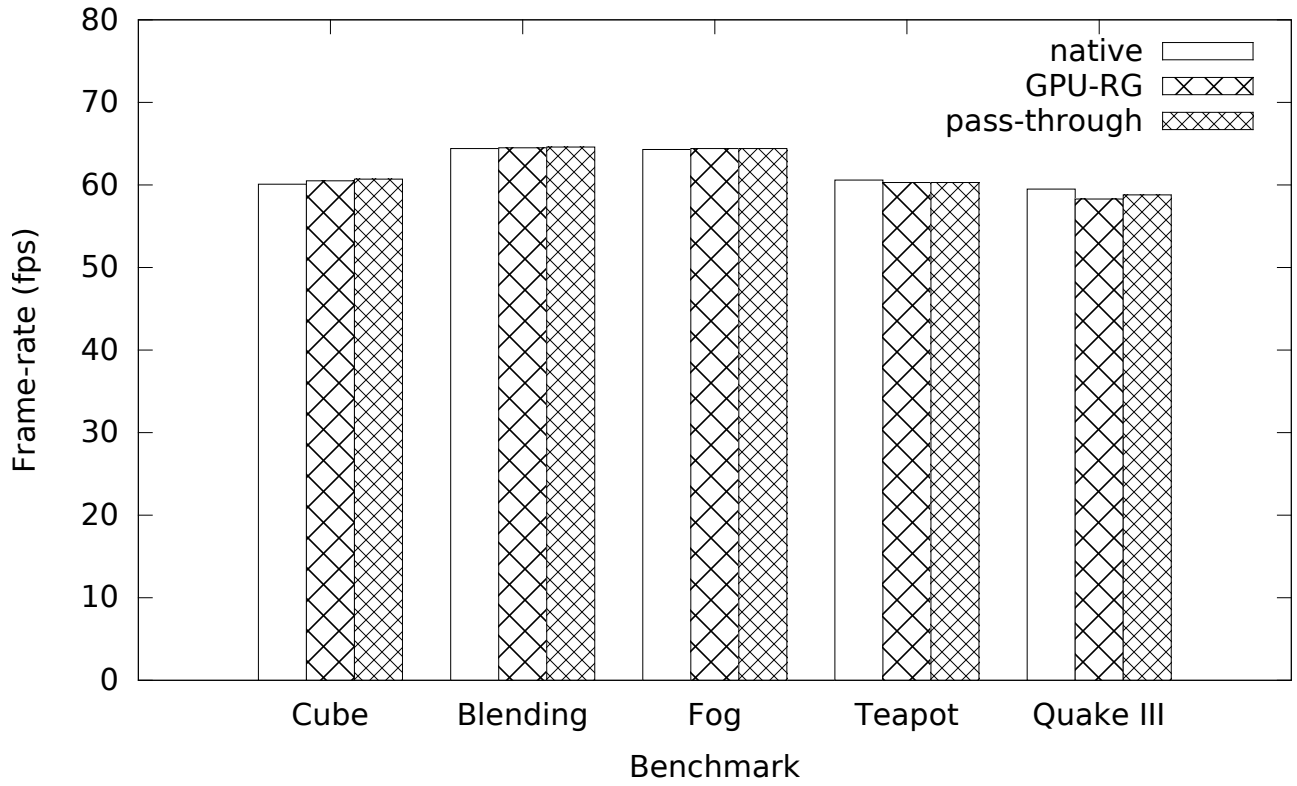
Pass-through

Cyanogenmod on L4Linux on Fiasco.OC
GPU driven by the guest kernel

GPU-RG

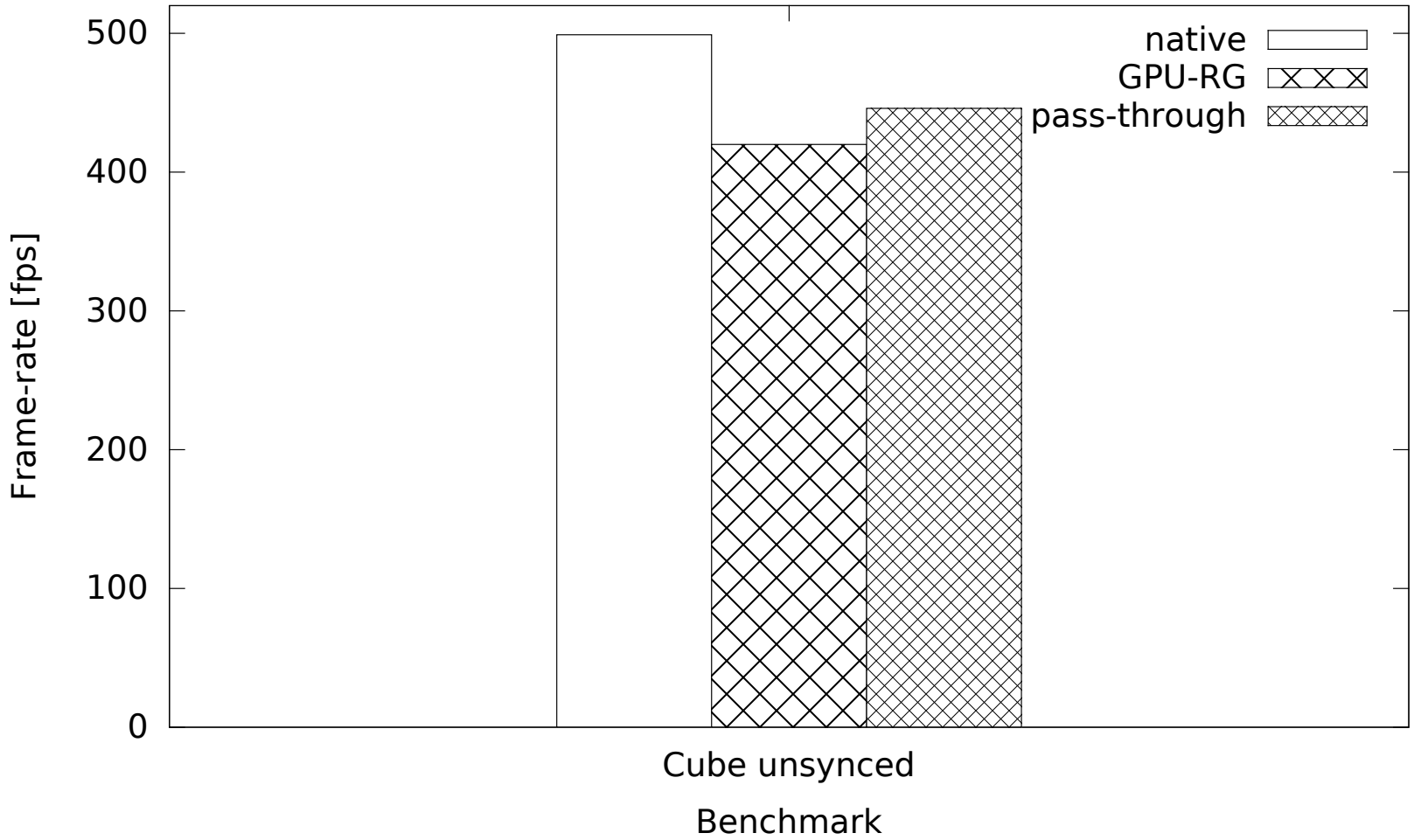
Cyanogenmod on L4Linux on Fiasco.OC
GPU driven by GPU-RG

Performance evaluation — benchmarks



Cube, Blending, Fog, and Teapot are part of the 0xbench [1] benchmark suite. Quake III is the `FOUR.DM_68` demo of QuakeIII Arena run with QIII4A [2].

Performance evaluation — benchmarks



Job Submission and Notification cost

experiment			GP[1]	PP[1]
native	submit	[μs]	15.0	25.2
pass-through	submit	[μs]	22.1	34.9
	notify	[μs]	3.6	3.2
GPU-RG	submit	[μs]	47.3	67.5
	notify	[μs]	52.8	49.7

Takeaway:

To meet a job submission rate of 60 Hz, an additional 2.3 % of CPU utilization is incurred on one CPU core.

[1] The ARM Mali 400 MP4 GPU has a geometry processor (GP) and 4 pixel presenters (PP)

Conclusion

Secure GUI (Trusted Path) addresses:

- Impersonation attacks
- Eavesdropping attacks
- Impact on CPU load and TCB

Secure GPU virtualization addresses:

- Enforced isolation of GPU jobs
- Low overhead for GPU jobs
- Low impact on TCB

Questions?

References I

[1] 0xbench.
https://code.google.com/p/0xbench/.

[2] Qiii4a.
https://play.google.com/store/apps/details?id=com.n0n3m4.QIII4A&hl=de.

[3] Cve-2014-0972.
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0972, 01 1014.

References II

[4] Rob Clark.
Fire in the (root) hole!
http://bloggingthemonkey.blogspot.de/2014/06/fire-in-root-hole.html.

[5] Rob Clark.
Kilroy.
https://github.com/robclark/kilroy.

References III

[6] Janis Danisevskis, Marta Piekarska, and Jean-Pierre Seifert.
Dark side of the shader: Mobile gpu-aided malware delivery.
In Hyang-Sook Lee and Dong-Guk Han, editors, Information Security and Cryptology - ICISC 2013 - 16th International Conference, Seoul, Korea, November 27-29, 2013, Revised Selected Papers, volume 8565 of Lecture Notes in Computer Science, pages 483–495. Springer, 2013.

[7] Kostya Kortchinsky.
Cloudburst.
Black Hat USA June, 2009.