Taming Mobile Hardware & OS Diversity for Comprehensive Software Analysis

Ardalan Amiri Sani*, Zhiyun Qian†

*University of California, Irvine, †University of California, Riverside

Problem Statement

- Mobile devices are everywhere!
- Software running in these device must be tested for:
  - Functionality, e.g., crash analysis
  - Security, e.g., vulnerability analysis
- Challenge: these devices and their software are extremely diverse, requiring expensive and time-consuming device-specific testing.

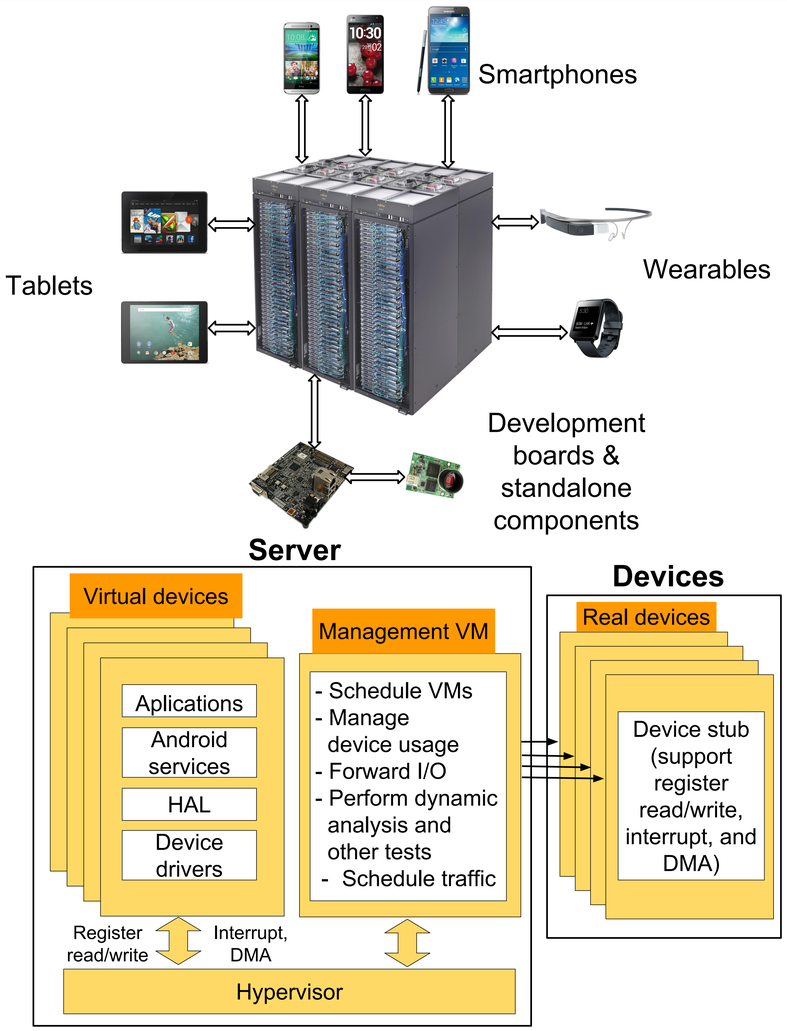
Tests should be performed on plenty of devices

$$ Expensive and Time consuming

Solution: Hybrid Mobile Farm

- Main idea: use Mobile Virtual Instances in servers for testing
- A Mobile Virtual Instance resembles a real mobile device
- Challenge: supporting Input/Output (I/O) for these instances.
- Solution: remote I/O to access real I/O devices in real devices

Architecture



- **Advantage 1: using VMs for testing**
  - Full control over all layers of software stack
  - Running analysis on a powerful server
  - Hardware consolidation by using VMs rather than mobile devices
- **Advantage2: using real device's software**
  - Finding device specific bugs and vulnerabilities
  - Much better testing platform for kernel and driver code (e.g., inspect crashes)

Design Goals

- Analyze known or unknown kernel/driver exploits. Adapt exploits from one platform to another.
- Perform dynamic analysis (e.g., fuzzing) on different parts of system software for a variety of devices
  - Focus on device specific parts of the system (e.g., device drivers)
- Optimize the mobile farm for the speed of testing and high degree of hardware consolidation

Technical Challenges

- Booting an unmodified mobile OS image of an ARM based mobile device in an x86 server
  - The source code for device OS may not be available
- Timing differences, which may change the behavior of the kernel/driver
  - Slow remote I/O can cause timeouts.
- Optimizing communication bandwidth
  - Specially for high throughput
  - I/O devices, such as camera and GPU

Acknowledgements

- Supported by NSF CNS-1617481