Poster: A Paravirtualized Android for Next Generation Interactive Automotive Systems

Soham Sinha, Ahmad Golchin, Craig Einstein, Richard West
Department of Computer Science, Boston University
Boston, MA
{soham1, golchin, einstein, richwest}@cs.bu.edu

ABSTRACT

Android's APIs, bluetooth support and smartphone integration provide capabilities for user interaction with In-Vehicle Infotainment (IVI) and vehicle control services. However, Android is not developed to interface with automotive subsystems accessed via CAN bus networks. This work proposes a new automotive system based on our Quest-V partitioning hypervisor, which allows Android to communicate and interact with timing and safety-critical services managed by the Quest real-time OS (RTOS). Quest is used to filter and receive messages from Android applications and to interface with a car's internal CAN bus in a timing predictable manner. Android is then used to host IVI applications and provide a user interface to real-time vehicle services. This system design allows Android to leverage the timing guarantees of Quest, while securely isolating critical hardware components and memory regions.

Quest-V hosts a paravirtualized Android 8.1 (Oreo) guest, which required modification of 126 lines of kernel code. Secure shared memory communication mechanisms between Android and a separate Quest guest provide real-time I/O to CAN bus networks.

ACM Reference Format:
Soham Sinha, Ahmad Golchin, Craig Einstein, Richard West. 2020. Poster: A Paravirtualized Android for Next Generation Interactive Automotive Systems. In *Proceedings of the 21st International Workshop on Mobile Computing Systems and Applications (HotMobile '20), March 3–4, 2020, Austin, TX, USA.* ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/3376897.3379163

DESIGN OVERVIEW

Our interactive automotive system uses Android as the basis for next-generation IVI applications and ADAS user-interface control. Our approach supports the co-existence of the Quest RTOS with Android on the same single-board computer, to manage timing-critical components of the vehicle. For example, an ADAS torque vectoring and traction control service configured for use on wet, dry, or snow-covered roads, must manage updates to wheel torques within specific time bounds to prevent the vehicle skidding out of control. While we want real-time control to be handled by suitably predictable services, the interface to configure ADAS settings will be exposed to Android.

The Quest-V partitioning hypervisor [1] supports the co-existence of Android and Quest, with real-time communication between each guest managed by secure shared memory channels (as shown in Figure 1). Thus, Android is empowered with real-time capabilities afforded by Quest, and Quest is empowered with improved user-interactivity capabilities provided by Android.

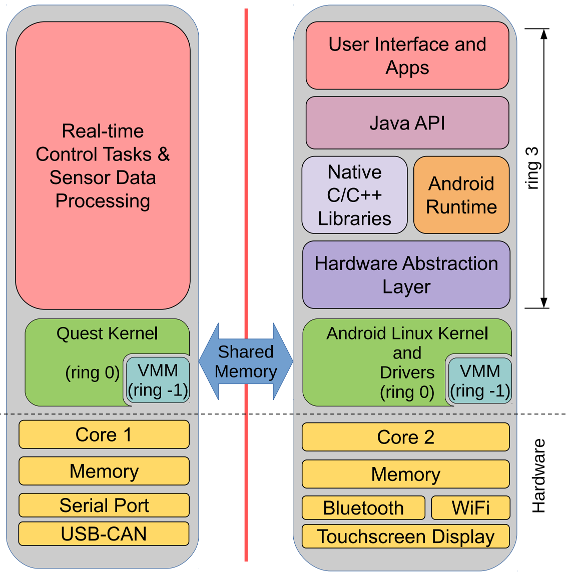

Figure 1: Design of the Quest-V Automotive System

EVALUATION

We prototyped our system on an Up Squared Board, featuring an Intel Apollo Lake Pentium N4200 processor. Experiments on the Android startup time shows that our paravirtualized approach has similar performance to Vanilla Android: the Quest-V paravirtualized Android takes 23.7s to boot, while a vanilla Android takes 16.6s. The reason behind similarity of the performance is the minimal number of VMexits in the paravirtualized Android.

In a range of experiments, we also show that the Quest-V paravirtualized Android has more timing predictable I/O latency and higher I/O throughput for both synchronous and asynchronous communication. We are exploring more real-time I/O and notification capabilities, and secure communication primitives between Android and Quest for complete deployment of the interactive automotive system.

REFERENCES

[1] R. West, Y. Li, E. Missimer, and M. Danish. 2016. A Virtualized Separation Kernel for Mixed-Criticality Systems. *ACM Transactions on Computer Systems (TOCS)* (2016).