Cross-layer Instrumentation for Smartphones

James Snee, Dr Andrew Rice

The Problem

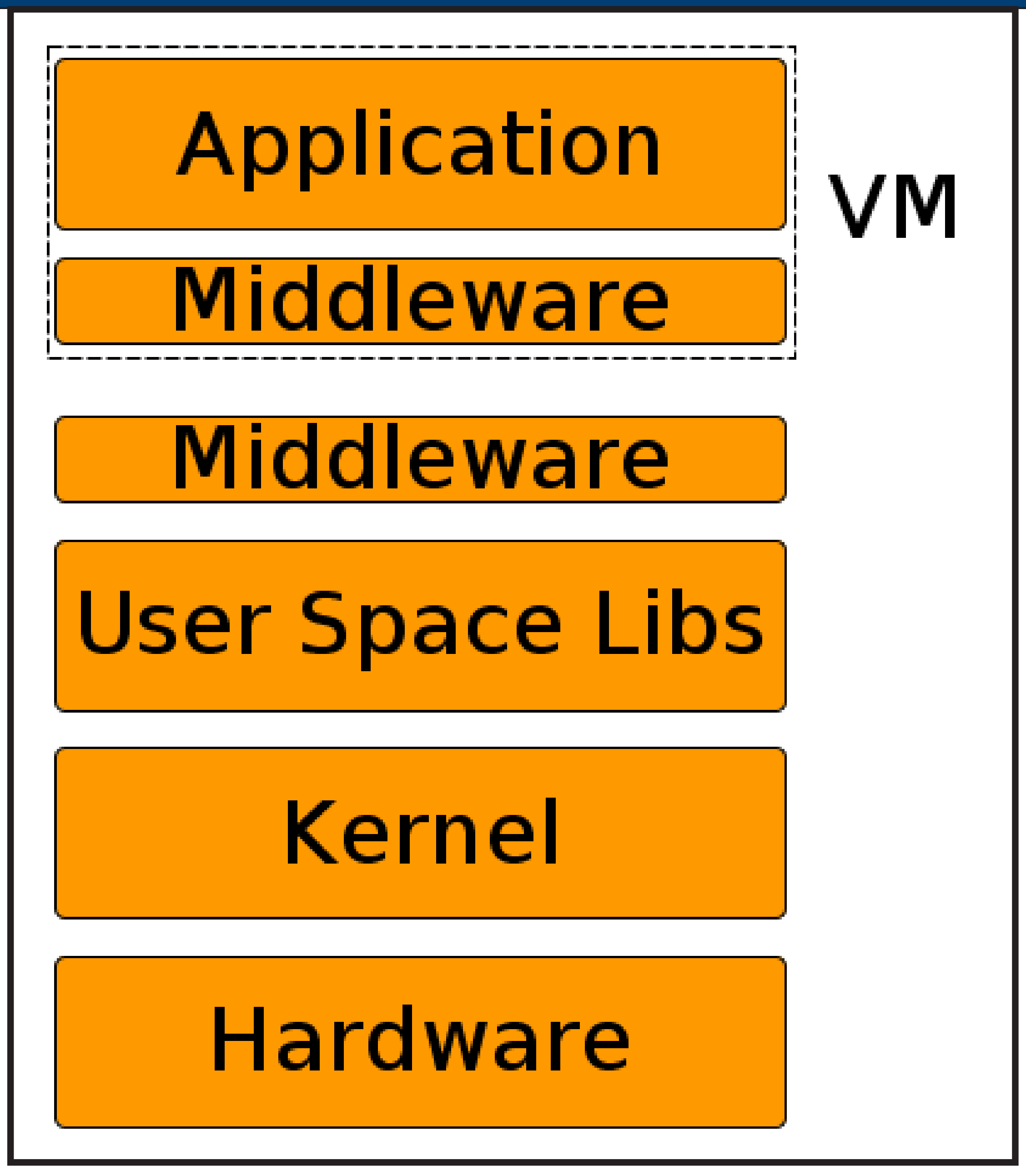

Figure 1. The typical layers of a modern computer system.

When testing software, we often see outliers in the experimental results. Investigating these anomalies is difficult given the number of interacting subsystems and layers of indirection that make up modern computer systems. Figure 1 shows a simplified view of a typical, deep-layered system. These layers exist primarily to abstract away the complexity of controlling the hardware directly, therefore providing a simpler programming interface to the user. But all of these interacting layers make it difficult to identify exactly why two executions of the same application may perform differently.

We hypothesise that the shared state inherent within the many layers of the system stack, directly affects the behaviour of the running application. We believe that this is important to investigate since many current smartphone software stacks make use of shared *services* to provide anything from location updates to network state monitoring and that their internal state directly influences the behaviour of the application making use of them.

Existing profiling and debugging tools often only provide execution traces at a single layer and at a fixed granularity and therefore aren't able to identify these cross-layer bugs.

Our Solution

We propose a method that relies on the analysis of execution traces recorded at different layers of the system in order to identify a typical execution, or the execution's *fast-path*. These traces are then collated and anomalies can be identified and investigated by characterising the execution of the application and looking for those traces that don't fit the character. The investigation of these anomalies can also lead to new, more targeted instrumentation being placed into the system, therefore making the investigation into anomalous behaviour more granular. Figure 2 shows the required steps in our method.

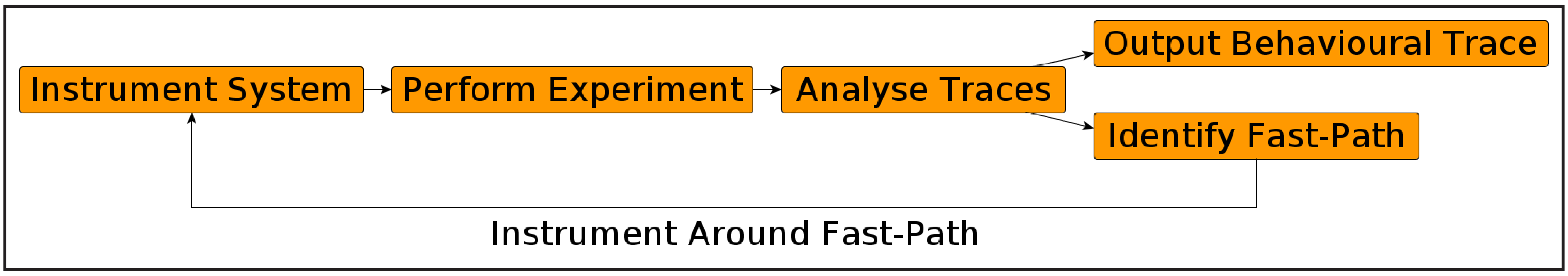

Figure 2. The various layers at which we gather execution traces and the methods we use to capture them.

We are currently testing our method using the Android mobile software stack. Figure 3 gives a simplified view of the various layers inside Android and the methods we are using or intend to use, to capture execution behaviour.

The novelty in our method is in the cross-layered approach to system tracing and the way we use algorithms prevelant in Intrusion Detection to identify anomalous behaviour in the entire system trace.

With this system, we hope to analyse anomalies ranging from simply understanding exactly what the idle state of one of these devices is, to the improper use of system resources.

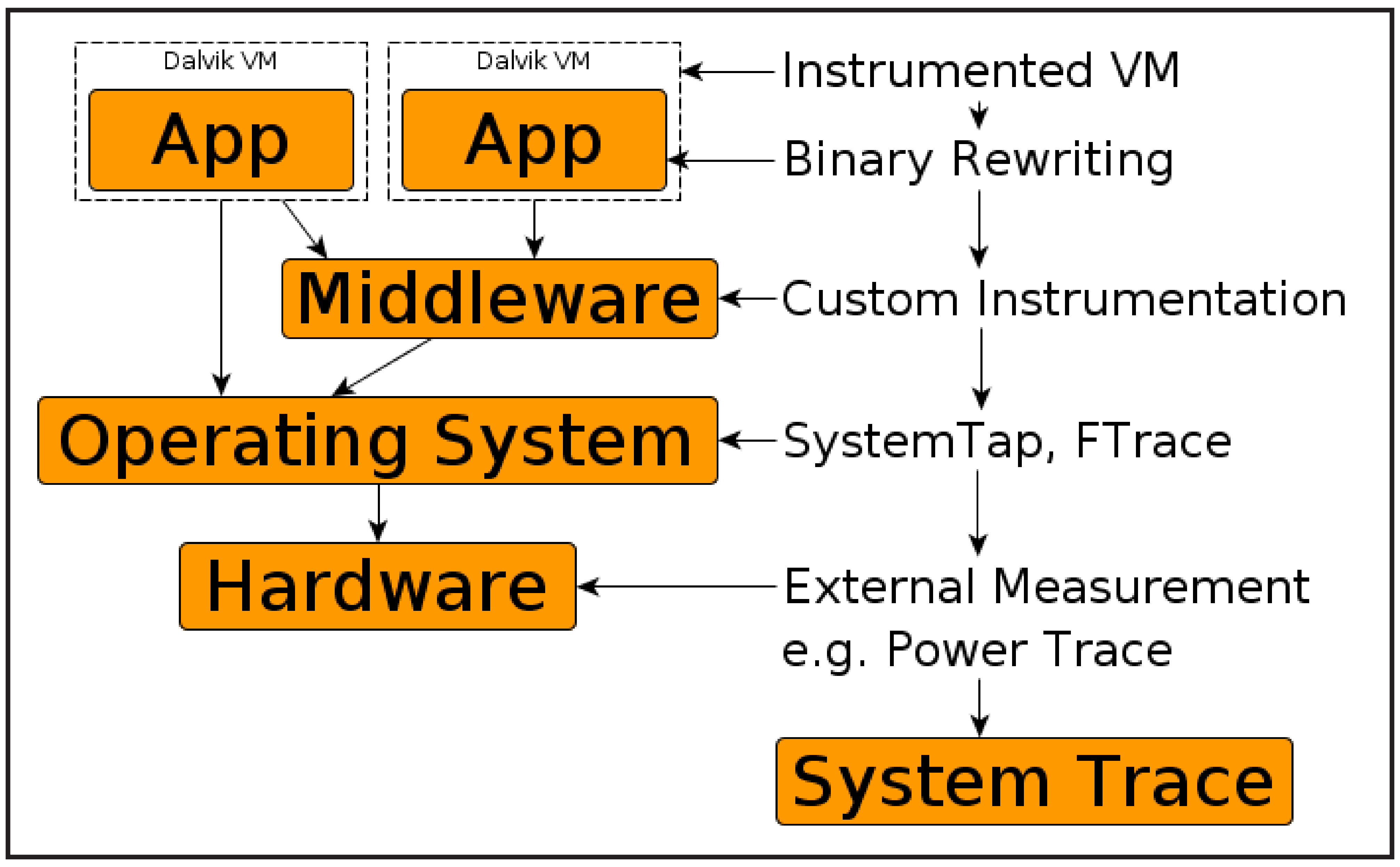

Figure 3. The various layers at which we gather execution traces and the methods we use to capture them.