The Art of Android *Doppelgänger*

Min Hong Yun (Student) and Lin Zhong (Faculty)
Rice University, Houston, TX

Doppelgänger [1] is the system capability of concurrently running multiple instances of an app started from a single installation or forked from the same running instance, all using the same user id. Android does not support *Doppelgänger* because it requires a separate installation and user id for each instance. *Doppelgänger*, however, can be very useful for Android devices. For example, it can launch instances side by side from a single installation and therefore make virtual phones [1] very lightweight, i.e., no need for re-login or changing runtime environments. Even if a networked service such as WeChat or SnapChat messaging service does not allow multiple connections, *Doppelgänger* can keep multiple instances connected to the same service account by forking a running instance that has been already connected. Moreover, *Doppelgänger* enables speculative execution of unmodified apps: a system can fork multiple instances from a running app, feed each a predicted user event, and keep the instance fed with the correct prediction once the actual user event arrives. For another example, *Doppelgänger* allows more effective app testing in real Android devices by running many instances in each device concurrently.

Supporting *Doppelgänger* in Android, however, is non-trivial. First, efficient way to checkpoint multi-threaded app does not exist in general and specifically for Android/Linux. Existing solutions take several seconds or longer because they either wait for safe points to checkpoint, e.g., that used by Respec [2], or copy virtual memory without copy-on-write scheme, e.g., that used by Flux [3]. Second, major Android components are designed assuming an app instance has a unique user id. As a result, instances of an app with the same user id appear to be same to the system and the system cannot select a specific instance, for example, to bring it to the front or send a messge to it. Furthermore, for the speculative execution we mentioned in the previous paragraph, Android needs to be redesigned to feed different events to different instances at the same time.

This poster presents our ongoing effort to realize *Doppelgänger* on Android for unmodified apps. In particular, we present two primitives: `forkall()` and the namespace virtualization. `forkall()` efficiently creates instances of an app by capturing the process virtual memory along with the whole threads' contexts. Rather than waiting for safe points like Respec, it aggressively captures running and sleeping threads. The thread that invoked `forkall()` forks sleeping threads and interrupts running threads to make them fork themselves. Optimizations such as preallocating kernel objects and sharing page tables can improve the efficiency. The namespace virtualization enables the instances to run simultaneously with the limited views of the entire system. Unlike the Linux's kernel-level namespaces, our implementation is above the kernel for sufficient semantics such as window and activity. An instance never sees the other instances of the app, whereas the system services see them all, but in discrete views. Thus, every instance thinks it is the foregrounded instance, stays in the running (resumed) state, and receives user events.

References

[1] J. Andrus, C. Dall, A. V. Hof, O. Laadan, and J. Nieh. Cells: a virtual mobile smartphone architecture. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pages 173–187. ACM, 2011.

[2] D. Lee, B. Wester, K. Veeraraghavan, S. Narayanasamy, P. M. Chen, and J. Flinn. Respec: Efficient online multiprocessor replay via speculation and external determinism. In *Proceedings of the Fifteenth Edition of ASPLOS on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XV, pages 77–90, New York, NY, USA, 2010. ACM.

[3] A. Vant Hof, H. Jamjoom, J. Nieh, and D. Williams. Flux: multi-surface computing in android. In *Proceedings of the Tenth European Conference on Computer Systems*, page 24. ACM, 2015.

[1] According to Oxford Dictionary, doppelänger is "an apparition or double of a living person."