**System Overview**

**The system consists of four major components, a) App Analysis; b) Real Physical Interaction discovery; c) Runtime prediction; and d) Policy Specification and Enforcement.**

**App Analysis aims to inspect and analyze Apps’ control flows for gaining the capability to perform dynamic testing in a smart home environment to build physical interaction graph, typically trigger conditions, actions and user settings of apps will be included in the interesting matrix. Moreover, its static analysis module will collect user configurations from deployed apps for generating fine-grained interaction graphs.**

**Real Physical Interaction Discovery is to find out real physical interactions given a smart home in an efficient manner through dynamic testing with respect to conditions and device restrictions. For efficiency, dynamic testing is divided into parallel and sequential testing.**

**Runtime Prediction employs temporal effect physical models to predict future status by continuously feeding temporal effects data from related collectors or sensors, for untested devices and newly added devices, online usage data is used to train the interaction graph and the models, physical models are updated on the change of apps user settings as well.**

**Policy Specification and Enforcement is to leverage a central control server to monitor and identify policy violations and take appropriate preventive measures to eliminate the violations or to stop approaching violations. Policy specifications are predefined user rules through a policy management console.**

**System Design**

**Following system overview, this chapter focuses on system design from a practical implementation point of view, and goes into in-depth details of each component.**

**App Analysis**

**App Analysis is composed of two modules I) Code analysis and instrumentation, II) Interaction graph generation.**

**The name of module Code analysis and instrumentation speaks for itself, it conducts app analysis by code instrumentation into apps to extract interaction graphs, different from traditional or existing works in which apps or devices are virtually conceptional or in laboratory, the targets here are deployed real-world devices. For the analysis, it assumes not only the access to read and scan the source code of apps, but also the permissions and privileges to tamper with the logic, this is the foundation for the success of this module. In the assumption, it injected snippets of code into critical functions of apps, like initializer, event handlers, et cetera.**

**This success of code instrumentation is an unrealistic thus unthinkable assumption, information security attracts tremendous focus in the industry in the past decades and today, big players in smart home like Samsung and Huawei deem it as the highest priority to protect their intellectual properties including source code, and critical precondition for their business success, they have placed all possible efforts and protocols to protect the properties against theft and leakage. In practice, instead of instrumentation, the smart home providers build a complete solution for home, including central management apps and the smart devices or sensors [1], [2], [3]. For now, let’s assume the instrumentation is attainable so that we can continue to the next section, as the paper may still be meaningful as reference for smart home providers in other aspects.**

**Real Physical Interaction Discovery**

**Testing cases can be generated for devices groups based on the static interaction graph from previous section, however, those testing cases are unable to address implicit and joint physical interactions in certain physical channels, for instance, temperature rise can be a joint effort of air conditioner and drier; humidity decrease may be attributed to fan and heater. Therefore, dynamic testing comes, and different from existing approaches, the author innovatively introduced two challenges to be addressed for dynamic testing.**

**One is long test circle due to Temporal Interaction, as testing physical interactions between IoT devices may take time, especially slowly changing status that requires long-time monitoring to identify, like previously mentioned temperature and humidity.  To test numerous such cases one by one is an unthinkably long process, the author promotes parallel testing for independent cases for optimizing cost of time.**

**The other one is safety and security concern during testing, as the whole solution is based on the real home environment, safety and security concern is an inevitable point as there are sensitive home appliances which might run into causing severe damages. So appropriate measures or constraints should be implemented beforehand, 1) Remove some sensitive devices like lock and sprinkler; 2) restrict the devices in safe ranges like target temperature of air conditioner. Since certain risky devices or devices’ conditions are excluded from the testing, we need to make them up by alternate two methods, first let user determine their status in interactions, second is to use runtime data to verify the interactions.**

**Comments: it is considerate to introduce the two challenges and have them addressed, by addressing them, it dramatically boosts the testing progress and kindly reduces the risk. However, they do not look novel as these are common senses to take efficiency and safety & security into consideration in a system design. Except that, It would be good if it elaborates how the user or runtime data could verify the interactions of absent devices.**

**Static testing cases**

**Static testing is to test based on all test cases generated through app analysis, the test cases iterate all possible combinations of modes of each device, one device with on or non-off mode at a time, for those connected to smart plug, there are two modes on and off.**

**Dynamic testing**

**Apart from the static testing, the implicit and joint interactions between devices of certain channels need to be verified too. Dynamic testing will categorize the tests into sequential testing, and parallel testing for independent flows. It discovers physical interactions by creating a normal usage baseline, which defines a reasonable fluctuation of certain devices without implicit or joint impact by others, then tests multiple related devices concurrently, if an abnormal value of a certain channel is captured, it is deemed as an occurrence of real physical interaction.**

**References**

**[1] Huawei smart home app -** [**https://consumer.huawei.com/ca-fr/support/content/en-us00693003/**](https://consumer.huawei.com/ca-fr/support/content/en-us00693003/)

**[2] Samsung smart home app -** [**https://www.samsung.com/us/smartthings/**](https://www.samsung.com/us/smartthings/)

**[3] Bell smart home solution -** [**https://www.bell.ca/Smart-Home/Packages**](https://www.bell.ca/Smart-Home/Packages)