**BOSS: Building Operating System Services**

Authors: Stephen Dawson-Haggerty, Andrew Krioukov, Jay Taneja, Sagar Karandikar, Gabe Fierro, Nikita Kitaev and David Culler

Reviewer: Wei Li

**Summary**

This paper introduces an operating system, BOSS (Building Operating System Services), which can be used in commercial buildings based on the fact that there are no layers of abstraction between the program and the building control system in existing buildings, i.e. applications are not portable among different environments.

Firstly, this paper presents the concept design of the BOSS architecture. It consists of six main subsystems which are respectively (1) Hardware Presentation Layer (2) Hardware Abstraction Layer (3) Time Series Service (4) Transaction Manager (5) Authorization Service (6) Control Process. The authors describe the requirements and principles of each part and the relationships among them within the whole architecture.

Secondly, this paper presents the general prototype implement of the whole system and detailed implements of four system components, (1), (2), (3) and (4). It also compares the performance of the database, readingdb, which is used for implement of Time Series Service, with the other popular databases, MySQL and PostgreSQL. Additionally, it illustrates the comparison the reversion policy the system adopts and the naive one.

Finally, in order to evaluate the system, authors chose a campus building in their school as the testing building and introduced three applications on this building. They showed us that the system architecture can implement these three applications simpler and more concise as well as energy reduction and coexistence with other applications.

**Contributions**

Compared with previous related works, this paper makes the following contributions.

At first, there have been many researches to develop programmable abstraction in order to provide portability and security to running tasks on the physical devices, especially in homes. Unlike Home Operating System, this system distributed applications and use transaction manager to enforce security strength. This makes system to be configured and cuts the cost.

Secondly, at Hardware Abstraction Layer, the system describes the sense or actuate point by its relationship with other components. Time Series Service part provides efficient access to historical data. These two functions are good for scalability.

Thirdly, previous related works need to specify the detailed model for each aspect of buildings and also define different data representations for applications. But this paper can make applications portable among various building environments and at a higher level of abstraction.

Fourthly, Hardware Presentation Layer provides uniform access to underlying physical resources and maps them into virtual representations while Transaction Manager and Control Process can hold the applications in real. These new places are not exist previously.

**Weaknesses**

There are several weaknesses in this paper.

1. In figure 6, authors take three example queries executed by the time series service to exercise three functions which are storing, selecting and cleaning. But this is not correct. The first example exercises the cleaning function. The following two examples both exercise the selection function. Neither implements the storing function. In detail design part, authors present that time series service consists of two parts: a stream selection language and a data transformation language. But in implement and evaluation part, they don’t show how the transformation language works.
2. When authors compare the readingdb database to other relational databases, for MyISAM, they just give the reason of its cheap insert performance and poor query performance but no statement about why this database scales with the size of the database rather than the size of the result set. In addition, they don’t explain in detail the phenomena that readingdb has the better insert performance and the best query performance. In this paper, from my view, readingdb’s good query performance should be the most important reason that authors adopt it as their database.
3. In transaction manager implement, authors introduce a custom revert action to prevent the unexpected oscillating of damper position when the setpoint action is reversed. But they don’t present how to design and implement the specialized reversion sequences. This is just one example of reversion policy for human failure on setpoint. What should be the reversion sequences if other type of unexpected failures happen? And in common sense, some kinds of failure cannot be predicted, authors don’t state any strategy for unpredicted failures, not to say reversion sequences that can be performed if those happen.
4. To evaluate the BOSS system, authors discuss little about the applications that are implemented based on the new architecture. They don’t explain how these three applications can coexist with each other. In other words, this paper put much words on introduction of designing concept of the BOSS architecture but little words to evaluate it and prove that the architecture provides really efficient abstraction for the control system of commercial building. Also, they have realized that they still need to make more effort to infer the HAL in existing buildings to allow applications to be portable, i.e. programmable buildings.