Singularity Design Note

8

Target Application Environment

Experimenting with the Home of the Future

1. Introduction

Designing an operating system in the abstract is a daunting task. This paper discusses a target environment for the Singularity research OS.

SDN0 stresses that one of the goals of Singularity is to offer a high-reliability environment through strong isolation and carefully specified interfaces between system components. Although high-reliability is a property that would benefit most computing environments, it has rarely been attempted or provided at the low-end of the computing spectrum. This is particularly true of the home environment.

2. The Home of the Future

Imagine the home computing environment of 7-10 years hence. In all likelihood, computers of various forms will be central to many common activities (and probably many more):

- entertainment (video and audio)
- games
- digital photography (capture and display)
- telephony (managing land-line, cellular, and internet telephone service)
- control applications (heating/lighting/physical security)
- cell-phone/PDA integration (remote application control, etc)
- network applications (storage, backup, background services)
- traditional PC applications (moving application state off the desktop)
- security services (e.g. gateways, firewalls, intrusion detectors)

Computers already support most of these applications, either through the desktop or through stand-alone "closed-box" products like DVRs. However, the recent success of wireless networking along with advances in power-management and affordable portability suggest that the form factor in which many of these applications are realized might be, in future, quite different from what we encounter today. The availability of portable devices implies that computing around the home can be much more disaggregated. Computer applications will be brought to bear where the user wants to employ

them rather than requiring a tether to a monolithic PC. Moreover, increasing computing power and user-interface functionality make hand-held devices an attractive platform for initiating and controlling computing tasks that involve multiple machines.

What then will be the backbone infrastructure that allows all this distributed goodness to happen? One might argue that everything will be coordinated by Internet-based services. This seems unlikely unless there is an unexpected jump in the availability of bandwidth to the home. And, in practice, there are privacy concerns about personal data residing in "the network" that might be hard to overcome. Instead, it seems more likely that we will see high-reliability, always-on appliances that run "in-the-closet" and serve as the nerve-center of the home of the future. Practicality argues that one-off custom boxes, a different one for each application, will not be able to compete with a platform architecture that can run multiple applications and be easily customized to the task at hand. Thus, it is essential that such appliance servers have the following properties:

- *robust* – must run all the time and be resilient to environmental factors;
- *management-free* – may not have a recognizable management user-interface;
- *open* – multiple applications per box; attractive platform for new apps;
- *real-time performant* – applications should receive clear resource guarantees;
- *secure* – high assurance with simple, useful security abstractions;
- *auditable* – run-time monitoring and logging must be easy;
- *application-centric* – applications must be full-fledged system entities.

This last bullet may not be obvious. To be practical in the home environment, it should be possible to arrange that individual applications be managed, in isolation, through network-based subscription services. Thus, it is critical that the application be a central abstraction within the configuration-management and security models. Furthermore, it must be possible to add or modify applications without serious risk of compromising the existing configuration of other applications.

The current design for Singularity aims to provide most if not all of the properties above. Therefore, this application space seems to be an excellent target for Singularity.

3. Exisiting Solutions

The current home marketplace is rife with single-system solutions for many of the application areas itemized above. There is TiVo and the like in the DVR space, various manufacturers gear for home networking needs, X10 based systems for home control, MP3-based home audio systems, and on and on. Most of these systems are based on special-purpose operating systems or Linux. Most of the boxes are highly specialized to the task at hand and designed to run in isolation.

Windows-based home appliance software has appeared for some specific entertainment applications. However, adding more Windows boxes to an existing home network has its drawbacks. Windows security and management is a serious concern. At the very least, owning more PCs implies more system management. At worst, Windows presents a huge attack surface that is not present in systems tailored to specific tasks. And moreover, in the Windows world it is easy to strand critical application state on machines that are not universally accessible. A more suitable model would embed application state in a secure, reliable home storage system where it would always be available.

It might be possible to use virtual machine technology as a platform for supporting multiple closed-box applications in a single machine. While the application isolation that can be achieved with virtual machine technology is certainly quite strong, the infrastructure for sharing resources cannot be easily

rationalized. In order to accomplish sufficient resource sharing for an open appliance server platform it might well be necessary to re-implement a number of key operating systems abstractions at the level of the hypervisor. This seems counterproductive. But moreover, it is likely that a set of independent closed-box applications will not be a truly compelling model for the home server. It's not hard to imagine how audio/video systems, telephony, remotes, sensor network, and home control devices might interact seamlessly to beneficial effect. A platform is needed to enable such interaction without sacrificing the benefits of strong isolation and carefully focused applications.

4. The Case for Singularity

Although the home of the future will most certainly contain a distributed computing environment, a fundamental rationale for Singularity is that a reliable distributed system must be constructed from nodes that are themselves reliable. Here we consider some specific requirements for an operating system capable of hosting an appliance server platform. The design goals of Singularity offer a good fit for all of these requirements. Since Singularity is a research operating system, these points will also serve to accent specific areas through which we expect to demonstrate innovation in this application domain.

1. *Zero or near-zero on-site administration*
2. *Seamless update/upgrade*
3. *Separated administrative responsibilities (per application)*

   Applications are first class entities in Singularity. The Singularity security and process/application model will maintain a strong notion of the rights and extent of an application. This, combined with a meta-data store that supports application independence, will give us the right tools to build applications with low-overhead administration and easy, reliable update. Similarly, meta-data for a Singularity application can be administered in isolation which allows for clean separation of management responsibilities between applications, whether the administrator is local or remote.

4. *Simple security model that allows user to control access to their private information*
5. *Access to shared information assets like calendars, etc.*

   The number one goal for the Singularity security subsystem is to present users and administrators with a simple, easy-to-use conceptual model. It seems inevitable that data sharing will be necessary between home applications. For example, the home heating system might well interact with the family calendar to determine an appropriate schedule. A key research problem will be how to control access to shared information without requiring a complex user interface or programming model.

6. *High assurance software*

   System and application software written in a strongly typed language with interactions between components tightly controlled through the use of checked contracts should give us a much higher level of confidence in the correctness of our software than with traditional techniques.

7. *Predictable reliability in the face of application extensibility*

   In Singularity, applications are extended by the introduction of new processes that interact through well-specified contracts. This will provide an important building block for establishing

new functionality in the context of existing applications.

8. *Soft real time and GC.*

   Soft real-time guarantees and suitable garbage collector performance for real-time applications are major research goals for Singularity.

9. *Rich sharing of peripheral devices (like speakers, screens, and microphones)*

   (Galen or Paul B. might be able to provide words here.)

An additional pragmatic incentive comes from the fact that there is relatively little legacy software extant for the home server market. Thus, the fact that Singularity offers little support for backward compatibility will not be as detrimental as it might be in other application domains.

5. Next steps

If we agree that a home appliance platform is a suitable testbed for our emerging Singularity prototype system, how should we proceed?

The first step is to identify one or two specific applications within the home appliance space that exercise as many of the requirements in the preceding section as possible. Because we have only limited programming resources, it will be a delicate task to determine applications that can be built within a reasonable time frame. Whatever applications are chosen, we should make sure to give high priority and attention to the elements of system design for which the Singularity OS represents enabling technology. The points enumerated in the previous section represent a first guess at what these elements might be.

With the experience of one or two application prototypes behind us, we should be able to identify and consolidate the salient properties of a home appliance platform that would be suitable for a building larger collection of applications.

As the Singularity design matures, it will inevitably become necessary to make design choices based on the target application environment. For example, in the security space, it might be more natural to base trust between local machines on physical proximity and ownership rather than organized certification hierarchy. We should not be afraid to make such design choices as long as we understand that we are making them. When in doubt, simple and specific solutions should trump complex and generic ones.

