Critique Week 1

Andres Calderon - SID:861243796

January 11, 2016

Exokernel: An Operating System Architecture for Application-Level Resource Management (Engler et al, 1995)

Fixed abstractions in traditional operative systems tended to limit the functionality and performance of applications. To address the problem, the paper proposed an exokernel with the main goal to separate protection and management. The small exokernel will provide secure low-level primitives and multiplexing over the available resources. High-level abstractions can be implemented efficiently by each application managing some resources according to their needs.

Traditional operative systems provide abstractions through definition of a virtual machine which applications can use. However, those abstractions are the same for all the applications which limit their performance and flexibility. I find practical the idea of the authors to use a minimal kernel (exokernel) to securely multiplex hardware resources but leave freedom to applications to use libraries for the more common abstractions or even code their owns. The paper states that the exokernel exports resources rather than emulate them. It uses three techniques to achieve that goal: secure binding, visible revocation and an abort protocol.

As the paper mentions: "Applications knows better than O.S. what they need" and it should be at that level where specific resources should be managed. Leaving the implementation of high-level abstractions to applications allows the kernel to focus on secure and protected primitives making its design simpler. Finally, the paper presents a prototype for the exokernel (Aegis) and libraries (ExOS) and a comparison to Ultrix (a mature version of Unix). The results seems to prove the advantages and assumptions of the authors.

However, even the proposal of the paper are logical and, indeed prove to be valid, the concept of exokernels do not really take off. Although the proposal of exokernel are relatively recent (mids 90's), the presence of well-established monolithic kernels seems to limit its adoption. Users and companies are reluctant to change a complete set of applications already deployed under the traditional approach. In addition, although the freedom to create custom abstractions to cover specific need sound tempting, it also could lead to an explosion of different and quite diverse implementations. The high amount and variety of libraries would make difficult their own maintenance. Curiously, the intention of making a simpler kernel could lead to a more complicate set of libraries.