SURVEY REPORT

NITHIN JOSHUA STEPHEN

CS14M033

Introduction

All traditional operating systems follow a monolithic kernel architecture in which the entire functions of the operating system reside in a single kernel. The entire operating system functionalities is working in kernel address space. Since the entire functionalities are incorporated into the kernel, the size of the kernel tends to be huge. In contrast, in a microkernel architecture only the essential functionalities needed by the operating system is incorporated into the kernel. These functions work in kernel space and run in supervisor mode. These essential functions include low-level address space management, thread management, and inter-process communication. Other functions like device drivers for sound, network run in user space. This results in a significant reduction of size compared to monolithic kernels. Genode OS follows microkernel architecture with emphasis on security where security critical functions are seperated from the rest of the OS.

Architecture

In genode, a program(components) runs in a sandbox called protection domain. The components need to communicate with each other in a secure manner. These components interact with each other through remote procedural calls(RPC). Each component implements an RPC interface which allows other components to access its functions(capability). So a components is similar to an Object in object oriented terms and capability to a pointer. So only if a component has a capability to other component, then only it can interact with it through remote procedural calls. Capabilities enable components to call methods of RPC objects provided by different protection domains.

In genode, every component other than the top level component has a parent. So when a child component is created, the parent assigns a fraction of its resources to the child. A component which provides a service(say GUI) announces the service to its parent. If a component requires a service, the component sends a session request to its parent. The component pro-

viding the service is termed server and the component availing the service is termed client. The client obtains the capability to the server and avails the service.

References

[1] D.Dolev,N.A.Lynch,S.S.Pinter,E.W.Stark,W.E.Weihl *Reaching approximate agreement in the presence of faults.* J. ACM, 33:499516, May 1986.

[2] L. Tseng and N. H. Vaidya *Iterative Approximate Byzantine Consensus under a Generalized Fault Model.* 2012

[3] M. H. Azadmanesh and R. Kieckhafer *Asynchronous approximate agreement in partially connected networks.* 2002

[4] Nitin Vaidya *Iterative Byzantine Vector Consensus in Incomplete Graphs* 2014

[5] Nitin Vaidya,Lewis Tseng,Guanfeng Liang *Iterative Approximate Byzantine Consensus in Arbitrary Directed Graphs* 2012