DesktopIP research ideas

August 3, 2017

1 Undergraduate project / thesis

1.1 Web application front-end for DesktopIP Storage-OS

"DesktopIP Storage-OS" is a specialized operating system on top of Debian that will act as a system management for DesktopIP supported storage file-system (imagine FreeNAS but with other compatible file-system). Some of the supported file-system is ZFS, NAS (FreeNAS and TrueNAS), Samba, etc. Storage-OS itself also must be compatible with other DesktopIP applications such as the virtualization management application. The proposed web interface is supposed to be developed in HTML5 and has an intuitive user experience. The application will be deployed in a server for 24/7, so its stability and performance need to be evaluated as well. The current implementation of Storage-OS is a text-based interface (TUI). Most probably using curses/ncurses library. Further and detailed requirements will be discussed with DesktopIP team when students showed their interest.

1.2 Single-board PC benchmarks

After a discussion with DesktopIP, deploying their native client on a Raspberry Pi is not recommended. The main reason is, the single-board PC (especially the ones that is based on ARM processor) is considered not reliable in the long term. In the current specification, they used Intel Atom processor in the client unit. Benchmarking some of the single-board PC (Raspberry Pi, Arduino, Intel Galileo) is beneficial for DesktopIP to optimize their client unit. Moreover, this research is useful for students to understand the bottom limit of virtualization on the client side. The first phase of this research is to port and benchmark for one type of board. In the next phase, comparing one porting with another and evaluate each of them in a more general perspective is necessary. One key point of the benchmark is the user experience in a longer interaction period (12-24 hours). Other key points are to be defined in further discussion.

1.3 Designing open source project of DesktopIP

DesktopIP has the vision to publish two types of their application: community version and commercial version. Community version is an open source based version that contains the server side of the application. This version includes the virtualization management (allocating and managing virtual machine) and other basic features. However, there is no client side in the community version. Because of feature disparity and the community version is not yet launched, the roadmap regarding this version is necessary. Students can analyze both community and commercial code, incorporate DesktopIP vision on both versions, and find the best way of the roadmap/timeline/feature addition of the community version.

2 Graduate Thesis

2.1 Modularize/Port DesktopIP native client application to popular Desktop Environment (DE) in Linux

Currently, DesktopIP uses Linux debian on their thin client. The implementation on its thin client is very dependant on a single DE (Desktop Environment), which is LXDE. It is very beneficial for the community or the company itself to remove this dependency. In a more general view, there are possibilities that many DesktopIP components are less flexible (i.e., highly dependent on one of the external factors). The student task is to find, analyze, and remove those dependencies if necessary. Specifically, the student can remove the DE dependency for DesktopIP to be able to use other DE. Alternatively, the student can port DesktopIP client application adjusting another DE. If necessary, other students can analyze other parts of DesktopIP application that still highly dependent. A student that interested in this project should be familiar with Linux architecture and system programming.

2.2 DesktopIP load balancer implementation

DesktopIP needs a load balancer to route their packet to its respective worker machine. The student task is to design the load balancer prototype. In a real case, the load balancer needs to detect different packet type (TCP, RDP, etc.), parse each of the packets, and forward it to the worker. Those packets need to be analyzed and processed accordingly. The student that implemented this project should perform a simulation, specifically in stress-testing the load balancer. Expected background is related to socket programming. Prior knowledge of many packet types on many protocols is desired. The load balancer front-end (GUI) need to be implemented in HTML5 and has an intuitive user experience. Further requirements and details will be informed by DesktopIP.

2.3 Finding optimal allocation method in Fasilkom UI's case

Towards the implementation of DesktopIP in Fasilkom UI, it is still unknown whether the default scheduling and allocation system in DesktopIP will match with Fasilkom UI's specific needs. Each of the virtual machines (VM) may be launched from a different physical server with a different load. The VM allocation can be based on the simpler algorithm such as round-robin or FCFS, or even a complex one such as using a fuzzy algorithm with multiple parameters (current load, task priority, etc.). In the first year/semester after the implementation, the usage log will be collected. Based on this, it is necessary to find out what is the optimal scheduling and allocating method specifically in Fasilkom UI's case.