Chapter 1. Introduction

1.1. Overview

Courseware provides common mediums for teaching Computer Science. This is especially true in an operating systems class, where students are easily confused by the new concepts, such as processes, interrupts, and concurrency. Some operating system courseware, such as Xinu [10] and Minix [28], are designed for teaching operating system. These courseware programs provide several programming assignments related to complex concepts in textbook, which give students hands-on experience to design and modify the kernel of operating systems. Students can understand these difficult operating system concepts more clearly by doing programming assignment.

Networking course also need courseware because networking itself is a very large and complex set of knowledges. An instructional networking system can help students to learn network concepts more clearly In TCP/IP networking, the TCP/IP protocol consists of many protocols, such as *arp*, *ip*, *rip*, etc. Almost all network textbooks discuss each protocol independently, but these textbooks do not illustrate its interaction among protocols. Student could not understand clearly the TCP/IP technology form textbook because they don’t know how multiple protocols run together.

Some operating system courseware, such as Xinu [10] and Minix [28], contain the TCP/IP protocols completely. Although these courseware systems provide complete TCP/IP protocol, however, they are difficult to adopt because these are designed to run directly on a bare machine. A lab full of computers that only running Xinu, is very

expensive. The porting and maintenance can create more serious problems. Furthermore, the laboratory need provide a network topology environment for observing the activities of the TCP/IP network, it is so difficult to laboratories provide a real network topology environment because build a network topology need many computers.

In order to solve above problems, we use message passing to make TCP/IP protocol to run as user processes under Linux, we called the system as PIN/XINU. PIN/XINU drew out the TCP/IP networking module from Xinu [10], and modify the module to become a user-level process under Linux. Xinu [10] provides a TCP/IP example implementation that students can understand how multiple protocols operate together and contains complete source code of TCP/IP protocols which students can modify to implement their own TCP/IP software for learning TCP/IP protocol, so we choose porting the system into user process under Linux to assist in teaching TCP/IP networking.

The difference between Xinu [10] and PIN/XINU is one runs on kernel-level and the other is runs on user-level. Xinu [10] has to run on a bare machine, but PIN/XINU just needs one Linux based computer to run it because it is an application program. In order to run TCP/IP as user process, we use *message-passing* to simulate the host-to-host packet transmission.

Message passing provides a mechanism to allow processes to communicate with other processes. Because the behavior of processes communication in message passing is like packet transmission in network environment, we use message passing mechanism to simulate the behavior of packet transmission; that is, we replace the underlying hardware functions by message passing. In PIN/XINU, a PIN/XINU process represents a host, message communication between processes represents packet transmission between hosts.

PIN/XINU has some benefits for teaching networking course, it provides a TCP/IP example working system. Source code for the example system allows students to understand how the protocols interact. Furthermore, because PIN/XINU is just a user-level process, which isolates the TCP/IP working system from low-level machine-dependent concerns, its porting and maintenance are easier and more efficiency than Xinu [10]. Students can learn quickly in how to use PIN/XINU and could concentrate on the essential concepts of TCP/IP protocols in textbook.

PIN/XINU also provides a network topology simulator which can build communication links via message passing. Before using message passing to simulate packet transmission, PIN/XINU need to build the communication links for sending or receiving messages. Network topology simulator can build a virtual network topology environment according the *network topology description file* for students to experiment their TCP/IP implementation. The network topology description file defines a format, students can easily write the description file to build a virtual network topology for running TCP/IP protocol according the format.

PIN/XINU also provides hands-on experiences to modify the codes of TCP/IP protocols. The instructor can design some programming assignments which are related to TCP/IP networking course. By doing these projects, students can understand TCP/IP networking concept more clearly. Due to PIN/XINU is a user-level program, students can use this courseware easily because they do not need other knowledge unrelated networking, such as CPU, memory, and device driver, they just need basic C programming knowledge.

With PIN/XINU, it is reasonable to expect that a motivated undergraduate, even a C programmer who never study operating systems course, will be able to use and extend this system in a one semester course. This was the motivation for using PIN/XINU to be the courseware of networking course.

1.2. Organization

This thesis is organized as follow. In chapter 2, we describe background of operating system courseware, illustrate several well-known courseware, and then briefly explain networking course. In chapter 3, we describe how using message passing to make TCP/IP protocol to run as user process under Linux. Chapter4 illustrate the implementation of our system. Chapter 5 describes some sample assignments. Finally, we end the thesis with discussion, and conclusion in chapter 6.