Chapter 3. Using Message Passing to Make TCP/IP to Run as a User Process

In most operating systems, TCP/IP protocol resides in the kernel of operating system. For this reason, students are difficult to have hands-on experiences over TCP/IP source code. For example, if students want to modify the source codes of a protocol of TCP/IP on Linux, they have to modify the Linux kernel. To modify the kernel of operating systems directly is unrealistic for students because of the complexity of modifying operating system kernel.

In this thesis, we implement a system, called Pin/Xinu, which allows TCP/IP protocol runs as user process under Linux. In order to run TCP/IP as user process, we use *message-passing* to simulate the host-to-host packet transmission, we drew out TCP/IP protocol suite from Xinu [10], and then modify the TCP/IP protocol to become a user-level program..

Message-passing is a fundamental module of *interprocess communication*, which provides a mechanism to allow processes to communicate and to synchronize their actions. In this thesis, we build virtual network interfaces, virtual hosts and virtual network topology, and then use message-passing mechanism to simulate packet transmission on network. To put it briefly, Pin/Xinu build a virtual network environment to simulate network activities.

In Pin/Xinu, whole TCP/IP protocol suite runs in user-level, so students can easily modify TCP/IP source code and then compile and debug the system with general compiler and debugger, such as ***gcc*** and ***gdb***. In above reason, Pin/Xinu is a suitable hands-on courseware for teaching TCP/IP networking course.

In this chapter, we briefly give an overview of Xinu [10] firstly, and then illustrate how use message passing to make TCP/IP run as user process under Linux and make a description of the design and the architecture of PIN/Xinu.

3.1. Xinu

Xinu was developed for teaching operating system and TCP/IP protocol. In the previous chapter, we pointed out Xinu [10] includes all the essential operating system components which are mentioned in textbook. Xinu [10] retains the essential functionality of operating system and remove unnecessary details as far as possible. Therefore Xinu [10] has a smaller code size than commercial operating systems. In view of networking, Xinu [10] contains complete source code of TCP/IP protocols, so students can understand the implementation of protocols by reading it. Xinu [10] provides a TCP/IP example implementation that students can understand how multiple protocols operate together and they can modify the source code of TCP/IP protocols to implement their own TCP/IP software.

The architecture of Xinu [10] is shown in Figure 3.1. Xinu [10] has to runs on a bare machine because it is a kernel-level software like a commercial operating systems. TCP/IP protocol is resided in the kernel of Xinu [10]. That is, Xinu [10] contains a single copy of the codes for TCP/IP protocols and allow multiple program invoke the codes. Xinu [10] provides several function calls that TCP/IP protocols use to make a concise source code. Chapter 4 will give an overview of operating system concepts that are used in TCP/IP protocols in Xinu [10], which are the parts that we need to modify.

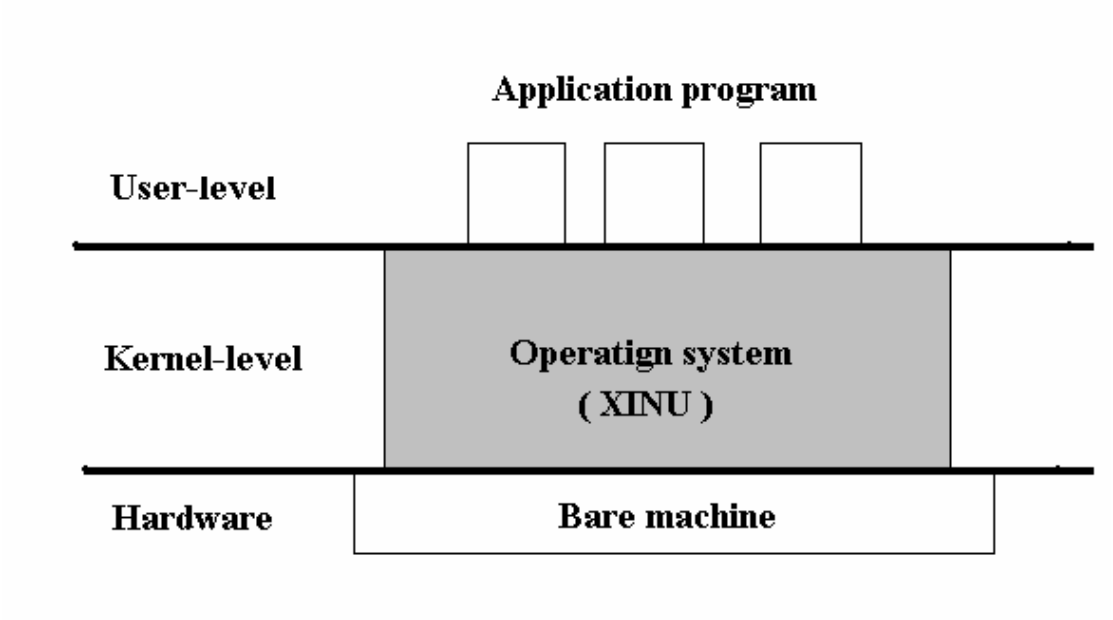

Figure 3.1 The architecture of Xinu

Although Xinu [10] provides an example TCP/IP working system for networking course, Xinu [10] has some problems if using it as networking course assisting courseware. The following illustrations explain why Xinu [10] is not appropriate for TCP/IP course.

For one thing, understanding how to use Xinu [10] is difficult. Because Xinu [10] is a real operating system, it contain many extra knowledge about operating system concept and has many codes which are related to hardware. Before using the system, students need know some operating system concepts which are unrelated to networking. Even if students understand operating system concepts, they also need spend lots of time on learning how to modify the code of the TCP/IP protocol because make an kernel-level program is much more difficult than user-level program.

For another, porting and maintenance of Xinu [10] have to spend lots of time and effort. For some reason, like to fit new machine, we need porting Xinu [10] to different computer system. Porting and maintenance of Xinu [10] are very difficult and need spend much time because there are lots of machine dependence codes in

Xinu [10].

Furthermore, using Xinu [10] as networking courseware to observe how two hosts interact by TCP/IP protocol in network is inconvenient because the laboratory needs to provide a real network topology environment. For example, if students want to observe how an host sends a packet to destination host across an internetwork, they need at least three computer to build the network topology, one as gateway, other two as source host and destination host. Providing enough machines to build a network topology for every student is unrealistic because build a network topology needs many hosts.

As mentioned above, using Xinu [10] as networking courseware has some problems, therefore we design a improving courseware, called PIN/Xinu, for teaching TCP/IP networking. The following illustrate the design of the courseware.

3.2. Using Message Passing to Make TCP/IP as User Process

Although Xinu [10] provides a TCP/IP working system for teaching networking course, above reasons which are mentioned in before section cause that there are few campus using the courseware for teaching networking course. In order to eliminate above problems, we use message passing mechanism to porting Xinu [10] as a user process under LINUX, we called the system as PIN/XINU.

The most different between Xinu [10] and PIN/XINU is one runs on kernel-level and the other is runs on user-level. The architecture of PIN/Xinu is shown in Figure 3.2. Xinu [10] has to runs on a bare machine because it is a kernel-level software like a commercial operating systems, and TCP/IP protocol is resided in the kernel of Xinu [10]. Figure 3.2 shows that PIN/XINU runs on user-level, we can easily find that PIN/XINU is a user program under LINUX. In order to build PIN/XINU system, we

draw out the TCP/IP networking module from Xinu [10], and use message passing mechanism to replace packet transmission, furthermore, we also modify the TCP/IP module for porting TCP/IP protocols into PIN/XINU.

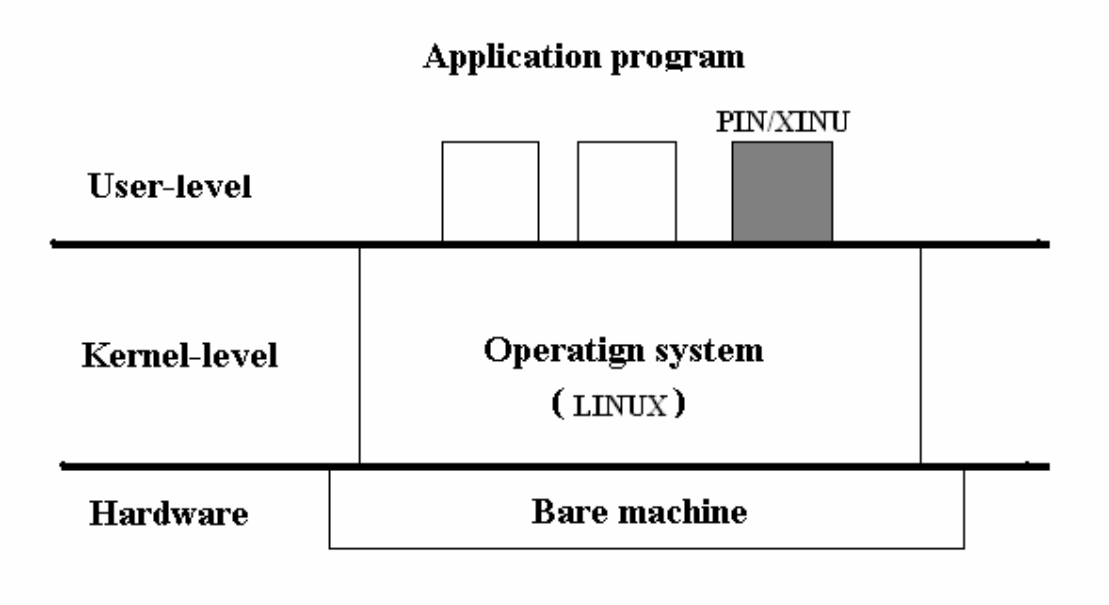

Figure 3.2 The architecture of PIN/Xinu

In general, TCP/IP protocol software resides in the operating system kernel. However, PIN/XINU is a user-level program and PIN/XINU contains whole TCP/IP protocols, that is to say whole TCP/IP protocols run on user-level. In order to let whole TCP/IP protocols run on user-level, we have to rewrite the module of TCP/IP.

The TCP/IP protocol can be viewed as a set of four layers, which are Application Layer, Transport Layer, Internet Layer, and Network Interface Layer. Each layer solves a set of problems involving the data transmission and provides services to the upper layer protocols. In TCP/IP layer module, network interface layer provides a network interface abstraction, which defines the interface between protocol software in the operating system and underlying hardware. The interface defines a data structure that allows protocol software to interact with the hardware primarily through this data structure. The following we will describe the part which we modify.

Figure 3.3 shows the data transmission through layers of the TCP/IP protocol. In Figure 3.3, the coarse line separates two layers, software layer and hardware layer, the four layer of TCP/IP protocol are in software layer and physical devices are in hardware layer, including physical network interface and wires. Figure 3.4 show the data transmission through layers of TCP/IP protocols which we had modified in PIN/XINU, we build a virtual network interface to replace physical network interface and use message passing mechanism to build a virtual wire to simulate physical wire. The dotted line in Figure 3.4 is represented original coarse line which separates software layer from hardware layer in Figure 3.3, so the part between dotted line and coarse line is that we had to modify.

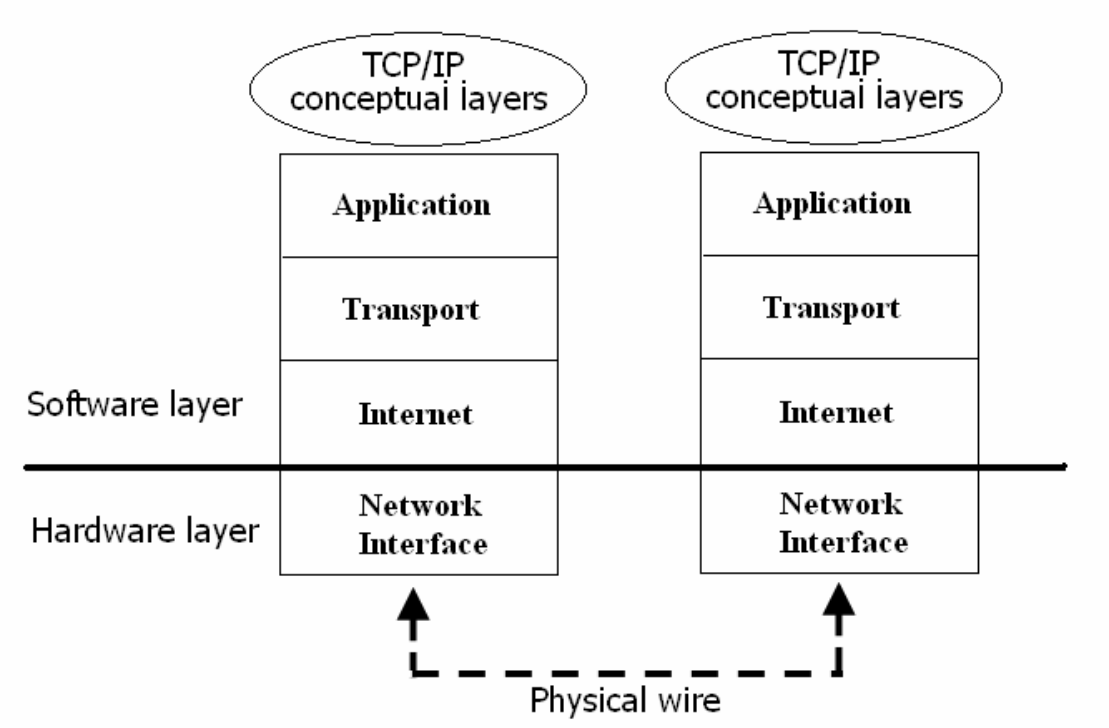

Figure 3.3 The layer module of TCP/IP

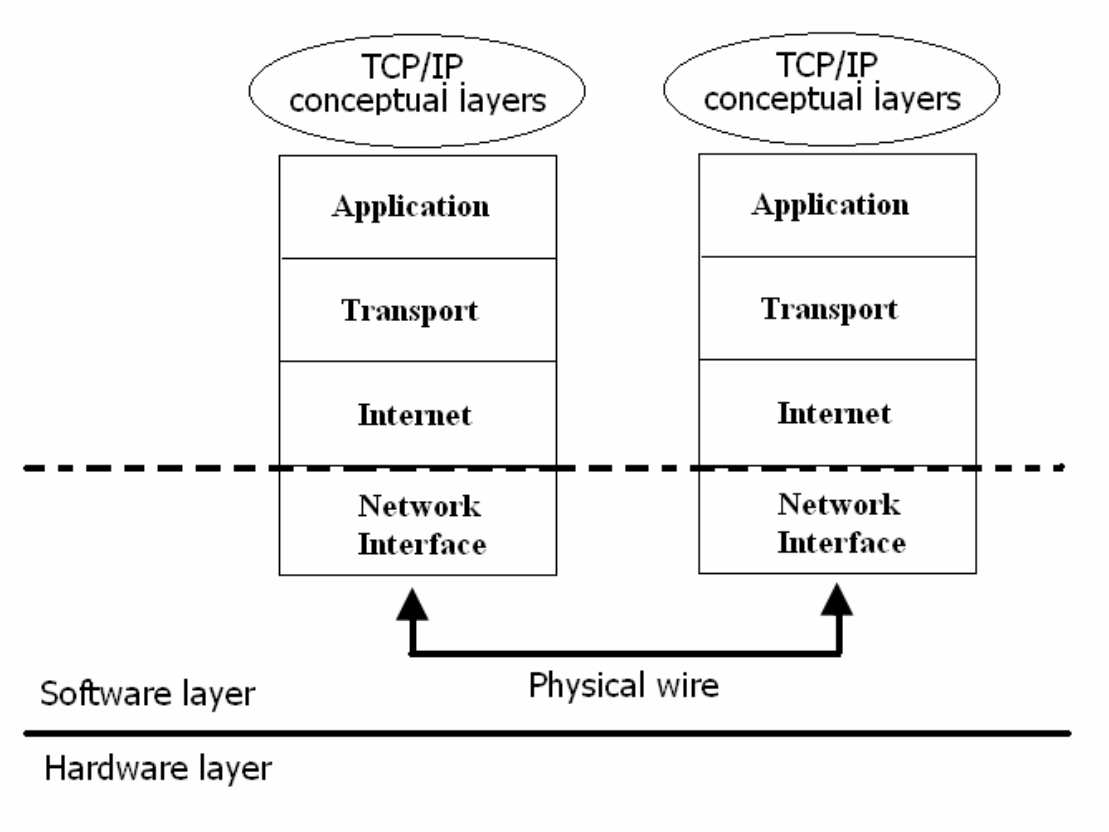

Figure 3.4 The Layer Module of TCP/IP in Pin/Xinu

Because message passing provides a mechanism to allow a process communicates to another process each other, we use message passing to simulate packet transmission in real network environment. Message passing has two categories, direct and indirect, here we choose indirect message passing system, and the system module is shown in Figure 3.5.

Indirect message passing use an intermediate queue, which is called a *mailbox* or a *port*, to store the sending message which send from sender, and then when the receiver want to receive the message, the receiver sends a request to obtain the message. With indirect communication, the messages are sent to and received from *port*. A port can be viewed abstractly as an object into which messages can be placed by processes and from which messages can be removed. Indirect message passing provides two functions, *send* and *receive*, for communication. The *send* and *receive*

functions are defined as follows:

send(Y, message) --- Send a message to port Y.

receive(Y, message) --- Receive a message from port Y.

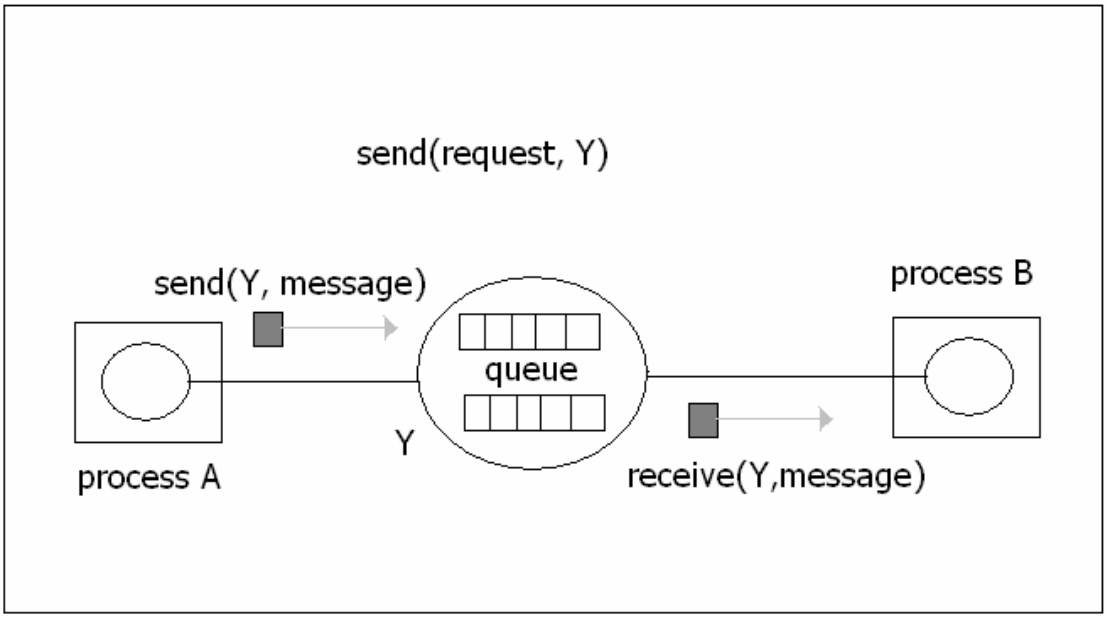

Figure 3.5 The module of indirect message passing system

Because the behavior of processes communication in message passing mechanism is like packet transmission in network environment, we designed to use message passing to simulate the behavior of packet transmission. In PIN/XINU, a PIN/XINU process represents a host, message communication between processes represents packet transmission between hosts. Incidentally, we also design a network topology simulator, which can build a virtual network topology, students can easily build a network topology, because they just only need write a network topology description file follow the format of the simulator and then use the simulator to build it.

In PIN/XINU, we design a TCP/IP working system as small as possible and let whole TCP/IP protocol and packet transmission to run in user-level, so students can easily use PIN/XINU to understand the TCP/IP networking concept. Students could use this courseware easily because they do not need other knowledge unrelated

networking besides C programming knowledge. In addition, we provide a network topology simulator to simulate network topology environment. Students just need one Linux based computer to run the courseware. To put it briefly, we implement a user-level TCP/IP networking and network topology simulator for networking course.