A virtual machine monitor introduces a significant network communication overhead. For example, it

is reported that the CPU utilization of a VMware Workstation 2.0 system running Linux 2.2.17 was 5

to 6 times higher than that of the native system (Linux 2.2.17) in saturating a 100 Mbps network [338]. In other words, handling the same amount of traffic as the native system to saturate the network, the

VMM executes a much larger number of instructions – 5 to 6 times larger.

Similar overheads are reported for other VMMs and, in particular, for Xen 2.0 [241,242]. To understand

the sources of the network overhead, we examine the basic network architecture of Xen [see

Figure 5.8(a)]. Recall that privileged operations, including I/O, are executed by Dom0 on behalf of

a guest operating system. In this context we shall refer to it as the driver domain called to execute

networking operations on behalf of the guest domain. The driver domain uses the native Linux driver

for the network interface controller, which in turn communicates with the physical NIC, also called the

network adapter. Recall from Section 5.8 that the guest domain communicates with the driver domain

through an I/O channel; more precisely, the guest OS in the guest domain uses a virtual interface to

send/receive data to/from the back-end interface in the driver domain.

Recall that a bridge in a LAN uses broadcast to identify the MAC address of a destination system.

Once this address is identified, it is added to a table. When the next packet for the same destination

arrives, the bridge uses the link layer protocol to send the packet to the proper MAC address rather than

broadcast it. The bridge in the driver domain performs a multiplexing/demultiplexing function; packets

received from the NIC have to be demultiplexed and sent to different VMs running under the VMM.

Similarly, packets arriving from multiple VMs have to be multiplexed into a single stream before being

transmitted to the network adaptor. In addition to bridging, Xen supports IP routing based on network

address translation (NAT).

Table 5.3 shows the ultimate effect of this longer processing chain for the Xen VMM as well as the

effect of optimizations [242]. The receiving and sending rates from a guest domain are roughly 30%

and 20%, respectively, of the corresponding rates of a native Linux application. Packet multiplexing/demultiplexing accounts for about 40% and 30% of the communication overhead for the incoming

traffic and for the outgoing traffic, respectively.

The Xen network optimization discussed in [242] covers optimization of (i) the virtual interface;

(ii) the I/O channel; and (iii) the virtual memory. The effects of these optimizations are significant for

the send data rate from the optimized Xen guest domain, an increase from 750 to 3, 310 Mbps, and

rather modest for the receive data rate, 970 versus 820 Mbps.

Next we examine briefly each optimization area, starting with the virtual interface. There is a tradeoff

between generality and flexibility on one hand and performance on the other hand. The original virtual

network interface provides the guest domain with the abstraction of a simple low-level network interface

supporting sending and receiving primitives. This design supports a wide range of physical devices

attached to the driver domain but does not take advantage of the capabilities of some physical NICs

such as checksum offload (e.g., TSO12) and scatter-gather DMA support.13 These features are supported

by the high-level virtual interface of the optimized system [see Figure 5.8(b)].

The next target of the optimization effort is the communication between the guest domain and the

driver domain. Rather than copying a data buffer holding a packet, each packet is allocated in a new

page and then the physical page containing the packet is remapped into the target domain. For example,

when a packet is received, the physical page is remapped to the guest domain. The optimization is

based on the observation that there is no need to remap the entire packet; for example, when sending a

packet, the network bridge needs to know only the MAC header of the packet. As a result, the optimized

implementation is based on an “out-of-band” channel used by the guest domain to provide the bridge

with the packet MAC header. This strategy contributed to a better than four times increase in the send

data rate compared with the nonoptimized version.

The third optimization covers virtual memory. Virtual memory in Xen 2.0 takes advantage of the

superpage and global page-mapping hardware features available on Pentium and Pentium Pro processors.

A superpage increases the granularity of the dynamic address translation; a superpage entry covers

1, 024 pages of physical memory, and the address translation mechanism maps a set of contiguous

pages to a set of contiguous physical pages. This helps reduce the number of TLB misses. Obviously, all pages of a superpage belong to the same guest OS. When new processes are created, the guest OS

must allocate read-only pages for the page tables of the address spaces running under the guest OS, and

that forces the system to use traditional page mapping rather than superpage mapping. The optimized

version uses a special memory allocator to avoid this problem.