CMPSCI 754 Multimedia Systems	**Spring 2001**

Lecture 15: 3rd April

Lecturer: Prashant Shenoy	*Scribe: Akash Jain*

In this lecture, we will study two techniques that minimize main memory traffic and provide OS support for high speed communication. We will also study a technique called Lazy Receiver Processing which is a network subsystem architecture that provides improved fairness, stability, and increased throughput under high network load.

15.1 Fast Buffers (fbufs)

Traditional systems like UNIX employ memory buffers which require copying between kernel and user space. This makes the I/O slow. Fbuf is a high bandwidth cross domain transfer and buffer management facility which makes high bandwidth I/O more efficient. Fbufs can be viewed as using shared memory (where sharing is read only and page remapping is used to dynamically change the set of pages shared among a set of domains), or using page remapping (where pages that have been mapped into a set of domains are reatined for use by future transfers).

I/O data is stored in memory buffers called fbufs, each of which consist of one or more contiguous virtual memory pages. A protection domain gains access to an fbuf either explicitly by allocating the fbuf, or implicitly by receving the fbuf via IPC. In the former case, the domain is called the *originator* of the fbuf; in the latter case, the domain is the *receiver* of an fbuf.

A virtual page remapping facility logically copies or moves a set of virtual memory pages by modifying virtual memory mappings. This invloves the following steps:

Allocation at originator:

- Find an unused virtual address space.
- Allocate physical memory pages.

Receive at receiver:

- Find and reserve a free virtual address range
- Remap pages from originator to this receiver.

15.1.1 Fbuf Optimizations

15.1.1.1 Restricted Dynamic Read Sharing

- Only pages from a limited range of virtual addresses can be remapped. This address range called *fbuf region*, is globally shared among all domains. This implies that an fbuf is mapped at the same vitrual address in the originator and all receivers which eliminates the need for finding a virtual address space at the receiver.
- Write access to an fbuf by either a receiver, or the originator while the receiver is holding a reference to the fbuf, are illegal. This eliminates the need for a copy on write.

15.1.1.2 Fbuf Caching

The idea here is that once some amount of data has been transmitted, chances of more mappings in near future are high. The following mechanism is used:

- Once an fbuf has been used, ordinarily, the fbuf would now be unmapped from these domains and the physical pages returned to free memory pool. Instead, write permissions are returned to the originator for reuse.
- Maintain the mapping in the receivers virtual address space.
- Reduces the number of page table updates required to two, irrespective of the number of transfers.
- Eliminates expensive clearing of pages.

15.2 Application Device Channels

- This is an approach that gives applications direct access to a network device for common I/O operations, thus bypassing the OS kernel and removing protection boundaries from simple application-network data paths.
- OS is only involved in the establishment and termination of network connections. Based on this information, the network adapter validates send and receive requests and thus maintains protection, safety and fairness.
- A copy of network communication software is placed into each user domain as part of standard library that is linked with application programs.

15.2.1 Advantages of ADC

- Bypassing OS kernel eliminates protection domain boundary crossing which would otherwise entail data transfer operations, domain switching and the associated drop in memory access locality.
- Since ADCs give application domains low level network access, it is possible to use customized network protocols and software.
- Since all processing and resources necessary for network communication are associated with an application process, it eliminates kernel resource constraints and scheduling anomalies.

15.2.2 Implementation of ADC

- User level implementation of the networking software, including device driver, network protocols and communication API.
- Shared memory communication channel between application process and network adapter. If the netwrok adapter contains memory that is accessible on the host I/O bus, a portion of this memory can be divided into VM page-sized portions and mapped individually into user processes. This approach is called *network interface-based ADCs*. Alternatively, the user process can access the ADC through its mapping of a main memory page and the network interface can access the ADC using DMA of the corresponding physical page. This is called *main memory based ADCs*.
- The network adapter must be able to demultiplex incoming network packets to the correct ADC, handle transmission requests on ADCs while exercising appropriate access control, interact with OS kernel and handle exceptional cases.

15.3 Taxonomy of data passing semantics

- *Buffer Allocation:* In application allocated buffering, the application determines the location of its input buffers on input and retains access to its output buffers after output. This is similar to copy semantics of UNIX I/O. In system allocated buffering, the system automatically allocates application input buffers on input and deallocates application output buffers on output and the applications cannot access the output buffers after output. This is typified by the move semantics.
- *Data Integrity:* Strong-integrity buffering guarantees that it will output the data contained in the application output buffer at the time of invocation, unaffected by subsequent overwriting by application. Weak-integrity allows I/O to be performed inplace, directly into or from buffers mapped to the application. The application can corrupt data being output or observe input data in inconsistent state.

15.3.1 Emulated Copy

- *Copy on Write:* The system removes write permissions from the application's mapping of the pages in the region. Application writes to the buffer are handled by copying the page.
- *Page Swapping:* The application buffer is page aligned and of length multiple of a page size. Swapping unmaps application buffer pages and then maps system buffer pages to the same virtual address.

15.4 Lazy Receiver Processing

UNIX based OSs use an interrupt driven network-subsystem architecture. The arrival of a network packet is signalled by an interrupt. The interrupt handler which is a part of the NI device driver, encapsulates the packet in an *mbuf*, queues the packet in the *IP queue*, and posts a software interrupt. The packet is processed by IP and then by TCP or UDP. Finally, the packet is queued on the *socket queue* of the socket that is bound to the packet's destination port. The software interrupt has a higher priority than any user process; therefore, whenever a user process is interrupted by a packet arrival, the protocol processing of the packet occurs before control returns to the user process. There are problem with this scheme as discussed in lecture notes.

The LRP architecture is as follows:

- The IP queue is replaced with a per socket queue that is shared with the network interface.
- The network interface demultiplexes incoming packet according to their destination socket, and places the packet directly on the appropriate receive queue. Packets destined for a socket with a full receiver queue are silently discarded.
- Receiver protocol processing is performed at the priority of the receiver process. Protocol processing for a packet does not occur until the application requests the packet in a receive system call.