Operating System Support for Multimedia

Research Issue

- Objective: deliver the service guarantees provided by the network and file server to applications
- Need for OS support
- Two problems:
  - Efficient techniques for streaming of data in and out of the system
  - Processor scheduling

I/O Subsystem

- Applications interact with I/O devices ⇒ control and data information must be moved between the kernel and application (user) protection domains
- I/O devices reside in the kernel domain and include
  - video and audio capture (analog to digital converters)
  - disk controllers
  - network interfaces
  - frame-buffers (video display subsystem)
- Data transfer across the user–kernel boundary involves:
  - Control and synchronization: e.g., initiation, producer-consumer synchronization
  - Buffer specification: e.g., location of data buffers in the user space
  - Data transfer: either copying or re-mapping of data buffers

Technology Trends

- Network and processor speeds are increasing at rapid pace . . .
  - 200MHz CPUs can operate on Gb/s data stream (several instructions per word)
  - ATM OC12 operates at 620 Mb/s, OC3 at 155 Mb/s, Fast–ethernet at 100 Mb/s
- . . . but memory speeds are not keeping pace
  - Effective DRAM copy rate (burst) is around 400 Mb/s
- Goal: deliver the available network/CPU bandwidth to applications

Dealing with Slower Memory Speeds

- Conventional approach:
  - Improve performance through the use of *cache*
  - Key issue: exploit the spatial and temporal locality inherent in program
- I/O data buffers generated by OS or applications do not have sufficient locality $\Rightarrow$ caching is not effective
- Causes:
  - Data movement (copying)
  - Inappropriate scheduling of various processing steps
- Remedy: minimize CPU/memory traffic during processing of network I/O

Network I/O: Fundamentals

- Two basic techniques: DMA and PIO
- Direct Memory Access (DMA):
  - Advantages
    - Concurrent operation
    - High data transfer rate (burst)
  - Disadvantages
    - Increased complexity of devices
    - Requires cache coherency protocols (hardware or software)
    - Data location and alignment constraints
    - Interference between device and CPU access to main memory

Network I/O: Fundamentals (continued)

- Programmed IO (PIO):
  - Advantages
    - Cache coherency automatic
    - *Potential* for increased cache hit–ratio during subsequent processing
  - Disadvantages
    - Processor handles data (at word or even half–word granularity)
    - Two transactions on the bus (e.g., read into cache from device register, write out to main memory)

Cross Domain Transfer

- Multiplexing necessitates transfer of data across protection domains (e.g., user–kernel)
- Problem: Copying aggravates memory bottleneck
- Various remedies:
  - Virtual–memory page re-mapping: move rather than copy semantics
  - Emulated copy: copy only if and when necessary (enhanced COW)
  - Shared memory: physical page mapped into two or more virtual address spaces

Fast Buffers (fbufs)

- High bandwidth cross domain transfer and buffer management facility
- Definitions
  - Fbufs: one of more contiguous virtual memory pages
  - Buffer aggregates: list of multiple discontinuous buffers
- Key concepts
  - Page remapping: avoid copies by mapping fbufs from one domain to another
  - Shared memory: share memory pages across domains

Fbufs: Optimizations

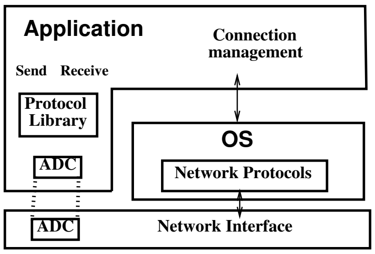
- Restricted dynamic read sharing
  - Limits the virtual address space that can be remapped to a fbufs region
    - Map a page at the same virtual address in all domains
    - Eliminates searching for a free virtual address
  - Assumes fbufs are immutable
    - Only the originator can write into the fbuf
    - No subsequent modifications are allowed
    - Eliminates the need for copy-on-write

Fbufs: Optimizations (contd)

- Fbufs caching
  - Takes advantage of locality in interprocess communication
  - Does not deallocate fbufs after data is consumed
  - Return fbuf to originator for reuse
  - Maintain mappings across domains to avoid remapping

Application Device Channels

- *Problem:* Operating system is in the critical path of I/O operations
- *Remedy:* Application Device Channels (ADC)
  - Allow applications to perform network sends and receives without OS intervention
  - OS is involved only in connection establishment and termination



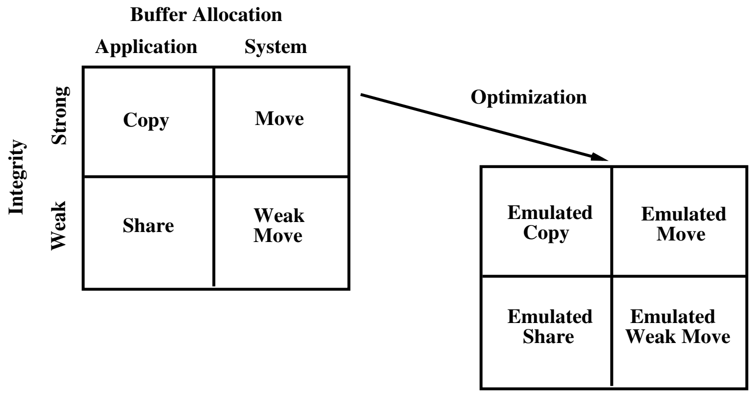
ADC Requirements

- User-level implementation of the network protocol stack
  - Improves performance: can use application-specific optimizations
  - Security implications: malicious user-level protocol libraries
- Shared memory channel between application and network interface card (NIC)
  - Memory on the NIC must be accessible to the host
  - Alternative: Map kernel memory into application space and use DMA fron NIC
- Network adaptor support
  - NIC firmware needs to perform early demultiplexing
  - Primitive NIC: use kernel thread to perform early demux

Taxonomy of Data Passing Semantics

- Two dimensions: buffer allocation and data integrity
- Buffer allocation
  - Application allocated buffering
  - System allocated buffering
- Data integrity
  - Strong integrity: data provided at output invocation is unaffected by subsequent overwrites
  - Weak integrity: buffer overwrites can cause errors

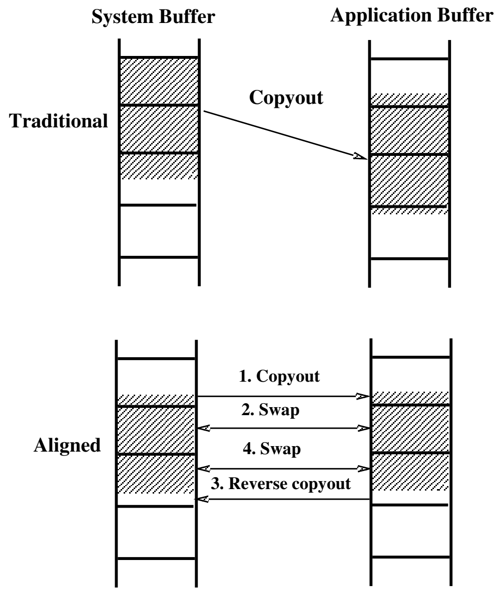
Taxonomy: Overview



Emulated Copy

- Copy on write
  - Remove write permissions for a fbuf on domain transfer
  - Application writes to the fbuf are handled by copying the page
  - Eliminates unnecessary copying overhead
- Page swapping
  - Swap application supplied fbuf with a kernel fbuf
  - Needs page aligned buffers

Example

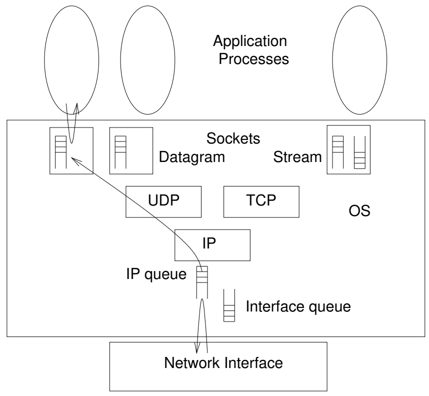


Emulated Move and Weak Move

- Emulated move implemented using region hiding
  - Remove read and write permissions on output
  - Reuse fbuf on input and reinstate read/write permission to domain
- Emulated weak move implemented using region caching
  - Maintain page mapping after a domain transfer
  - Reuse mappings for repeated I/O operations
  - Similar to fbufs caching

Lazy Receiver Processing

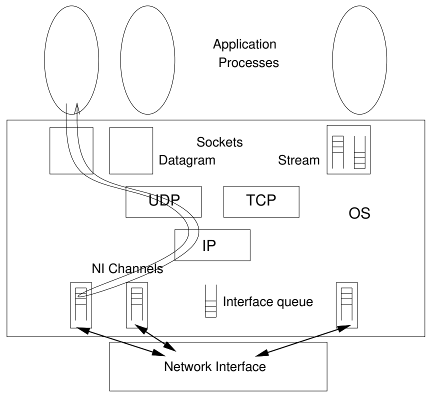
Unix Network Processing



Unix Network Processing: Limitations

- Eager receiver processing
  - Packet processing is interrupt driven (has higher priority than user processes)
- Lack of effective load shedding
  - Packets are dropped only after significant resources have been invested
- Lack of traffic separation
  - Misbehaving flows can affect other flows
- Inappropriate resource scheduling and accounting
  - Interrupt processing is charged to currently executing process

Lazy Receiver Processing: Architecture



LRP: Key Features

- Lazy receiver processing
  - Protocol processing delayed until receive time
- Per-socket queues
  - Allows early packet discard and eliminates unnecessary processing overhead for dropped packets
- Early demultiplexing
  - Eliminates interference across flows
- Proper resource accounting
  - CPU time spent in protocol processing is charged to appropriate process

LRP for UDP and TCP

- UDP
  - Transmit side is unchanged
  - Receive side uses LRP: protocol processing delayed until receive time
- TCP
  - Problem: TCP acks and retransmits $\Rightarrow$ difficult to delay protocol processing until receive time
  - Solution: Asynchronous protocol processing
    - Assign a kernel thread to each socket
    - Thread performs asynchronous operations (acks, retransmits)
    - CPU usage charged to the appropriate process