A Highly-Extensible Architecture for Networked I/O

Cynthia Taylor  
Oberlin College

Joseph Pasquale  
UC San Diego

ICNC  
January 30th 2013
Motivation
Why Remote I/O?
Transparency
Transformation
No Single Solution

- Different devices
- Different applications
- Different network conditions
- Different optimal solutions
Architecture
Networked Device Driver Abstraction for Transparency
Diverse Beneficiaries Require Easy Customization and Extensibility

- Device designers
- Application designers
- Users
Modular Architecture
Need to Connect Device and Application
Device Module
Network Modules
Application Module
Need to Add Data Processing for Network

- Averaging
- Bundling
- Buffering
- Compressing
- Discarding
- Encrypting
- Multiplexing
- Synchronizing
Transformation Module Pairs
Example Module Pairs

- Compression/Decompression
- Bundling/Unbundling
- Encryption/Decryption
Compression
Composability
Summary

• Device driver abstraction supports transparency

• Transformation module pairs allow processing of data

• Modular design supports customization, extension
Implementation
Implementation Goals

- Efficiency
- Ease of implementation
- Leveraging existing mechanisms
Kernel vs user space

- Insecure/buggy code is dangerous to run in kernel
- Allows developers to use any existing tools/libraries
- Copies between process boundaries must go through kernel
Run Predominately in Userspace to Support Extensibility
Modules as Processes Support Customization

- Can compose at run-time
- Scheduled by the kernel
- Automatically block on I/O
- Separate address spaces
Pipes Copy Between Processes
Implementation Summary

- Implemented at user-level whenever possible to support **extensibility**

- Modules are implemented as processes to support **customization**

- Pipes implementation for **ease of implementation**
Performance
Test bed

- Dell Optiplex 320
- Intel Celeron
- 133 Mhz FSB Clock
- Ping time of .12 ms between machines
- 11.3 MB/s throughput
Computing the Base End-to-End Time
Base End-to-End Time Results
Space Navigator
End-to-End Time of the Space Navigator
Overhead of Space Navigator Driver
Buffering Experiment
Buffering Performance
End-to-End Time with Transformation Modules

![Graph showing the relationship between the size of a char array and end-to-end time with different numbers of transformation modules.]
Summary

• Overhead is order of magnitude less than speed of network

• Adding additional transformation modules adds relatively little overhead, especially at small message sizes.
Conclusion
Summary

- System for I/O over network
- Application sees as driver
- Supports Transformation Modules
- Easily customized and extended to new devices and functionality