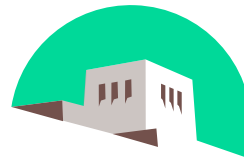


Splintered Operating Systems

Arthur B. Maccabe

maccabe@cs.unm.edu

Computer Science Department
The University of New Mexico



July 2002

Principles

- Cynical

The less I do, the less I can do wrong

Principles

- Cynical

The less I do, the less I can do wrong

- Noble

... make a habit of two things—to help, or at least do no harm.

Hippocrates, The Epidemics

Principles

- **Cynical**

The less I do, the less I can do wrong

- **Noble**

... make a habit of two things—to help, or at least do no harm.

Hippocrates, The Epidemics

- **Must be able to run “hero” codes**

Goal

Design something that could be built tomorrow

- **Problems with Puma/Cougar**

- only supports MPI programming model
- limited features

- **Systems**

- today's machines
- tomorrow's machines
- 2010 machines

- **Tailor OS to needs of application**

OS / Runtime

- Both provide abstract resources
- Both manage resources
- An OS manages *competition* for shared resources
- Shared resources may also be handled by remote servers

Factors that Influence OS Structure

- **Programming model**
 - explicit message passing
 - shared memory
 - message driven
- **System usage model**
 - batch queue
 - space sharing
 - interactive use
- **Architectural characteristics**
 - special purpose processors (NICs, PIMs, etc)
 - communication structures

Splintering

Move functionality to most appropriate place

- Consider sockets and TCP
- Does the application need sockets API or TCP inter-operability?
- Implement locally, in the kernel
- Implement locally, in user space
- Remote socket server

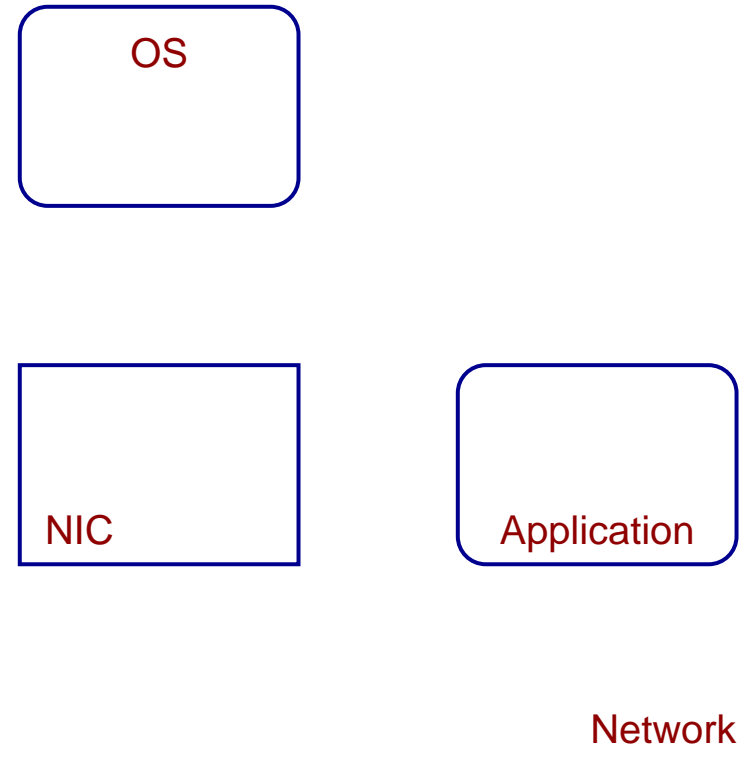
More Detailed Example

Don't **bypass** the OS, use it effectively

- Message reception
- Moving small bits of functionality

Traditional Network Interfaces

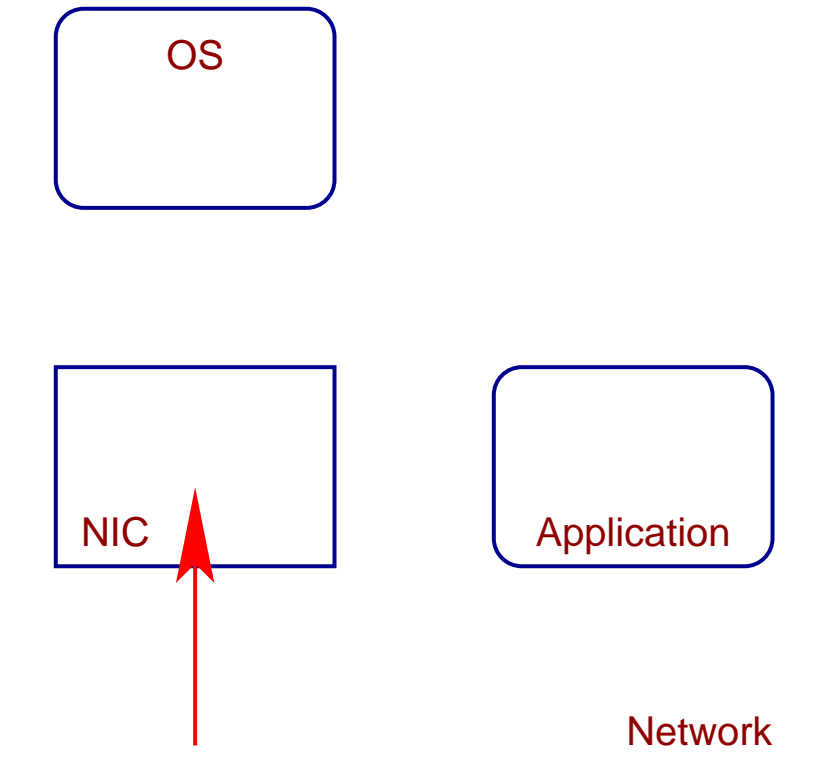
Receive side



Traditional Network Interfaces

Receive side

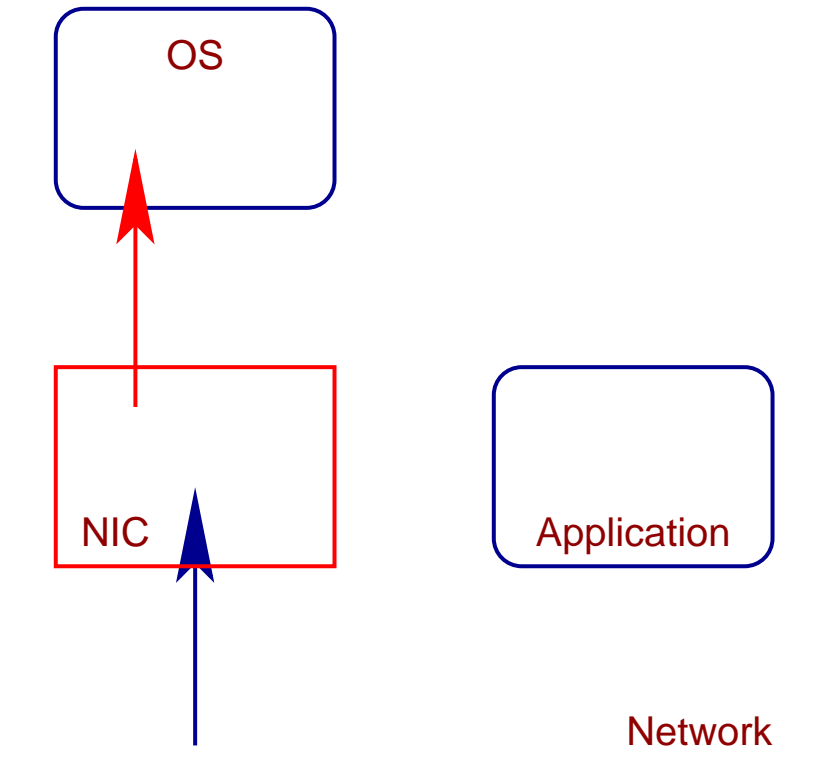
- Packet arrives



Traditional Network Interfaces

Receive side

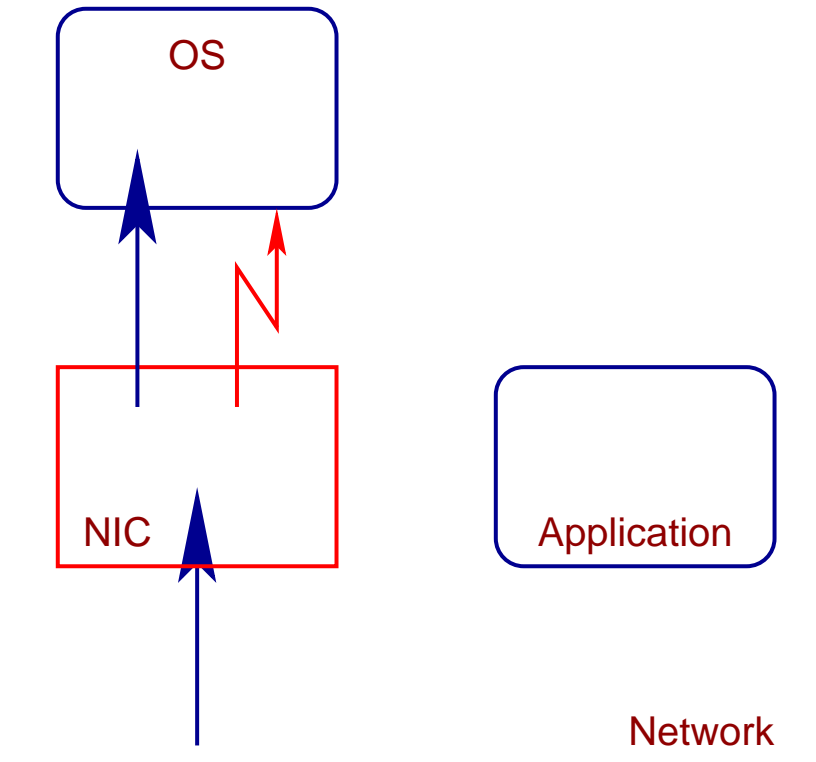
- Packet arrives
- Packet to OS



Traditional Network Interfaces

Receive side

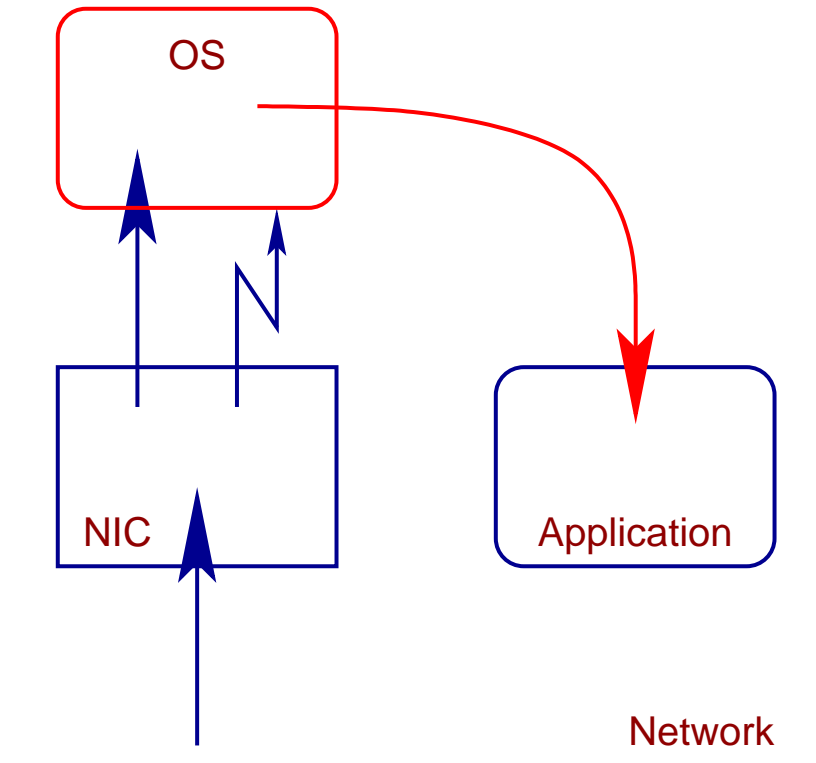
- Packet arrives
- Packet to OS
- Interrupt OS



Traditional Network Interfaces

Receive side

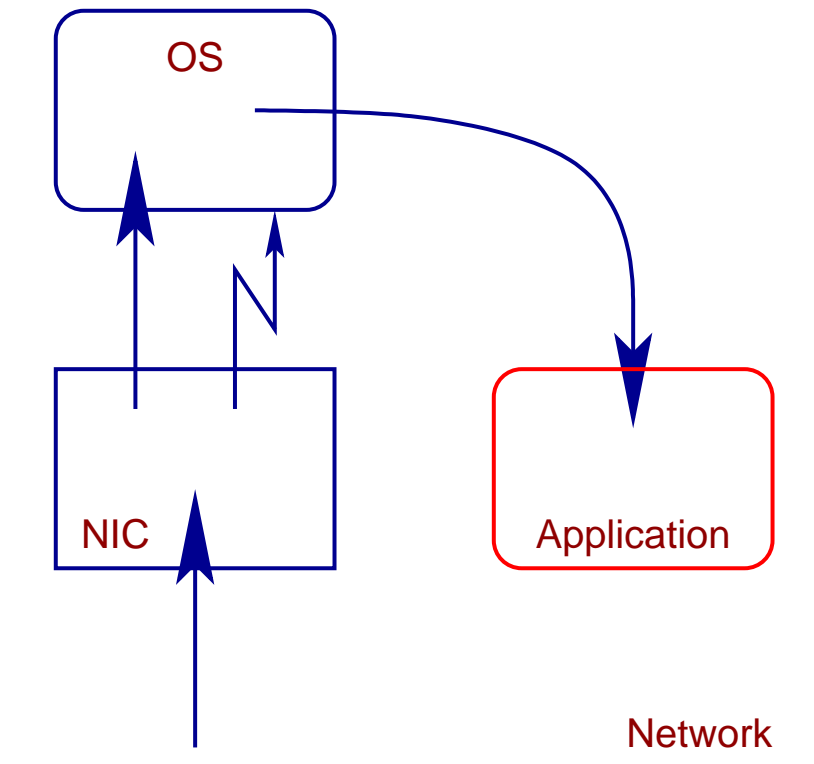
- Packet arrives
- Packet to OS
- Interrupt OS
- Data to app



Traditional Network Interfaces

Receive side

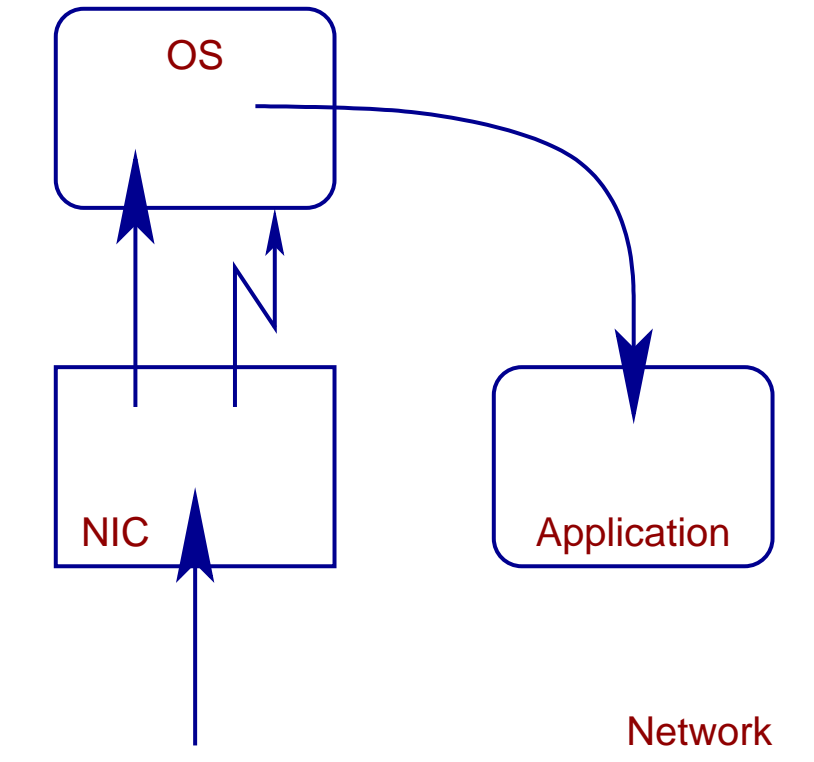
- Packet arrives
- Packet to OS
- Interrupt OS
- Data to app
- Schedule app



Traditional Network Interfaces

Receive side

- Packet arrives
- Packet to OS
- Interrupt OS
- Data to app
- Schedule app
- High Latency

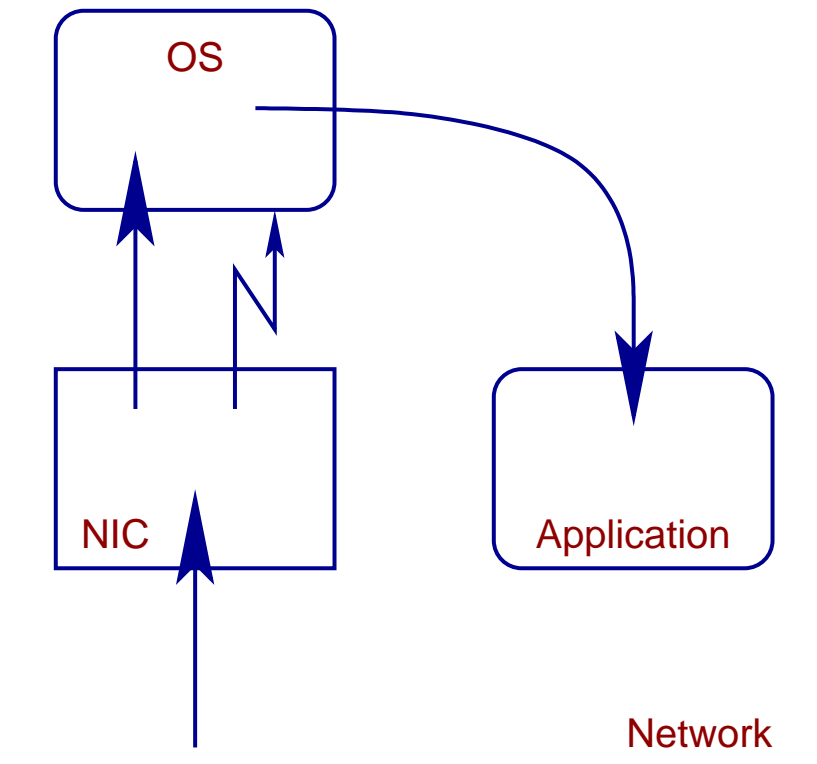


Traditional Network Interfaces

Receive side

- Packet arrives
- Packet to OS
- Interrupt OS
- Data to app
- Schedule app

- High Latency
- Memory Copy

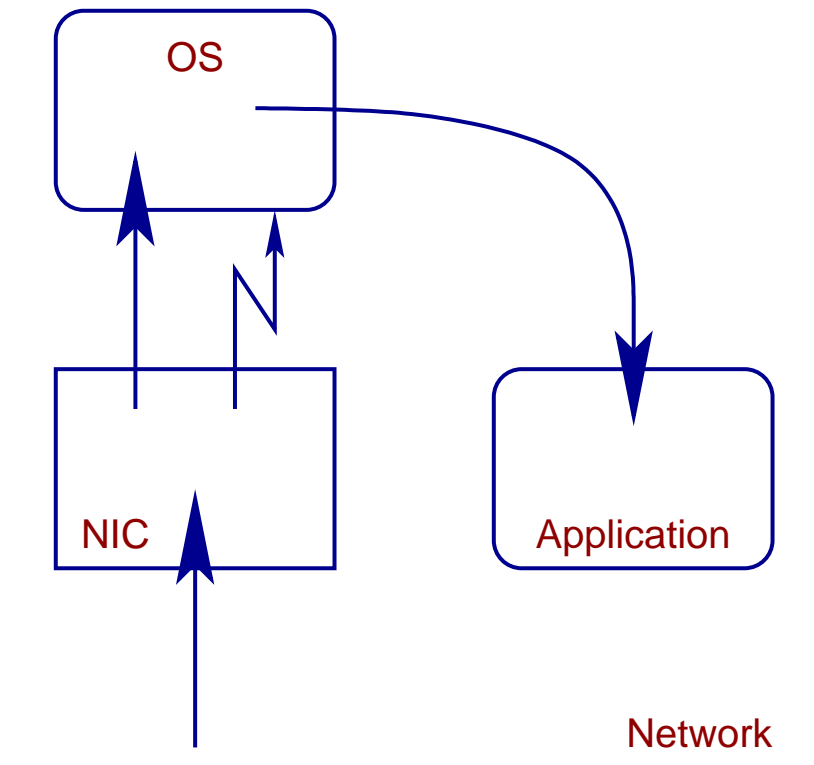


Traditional Network Interfaces

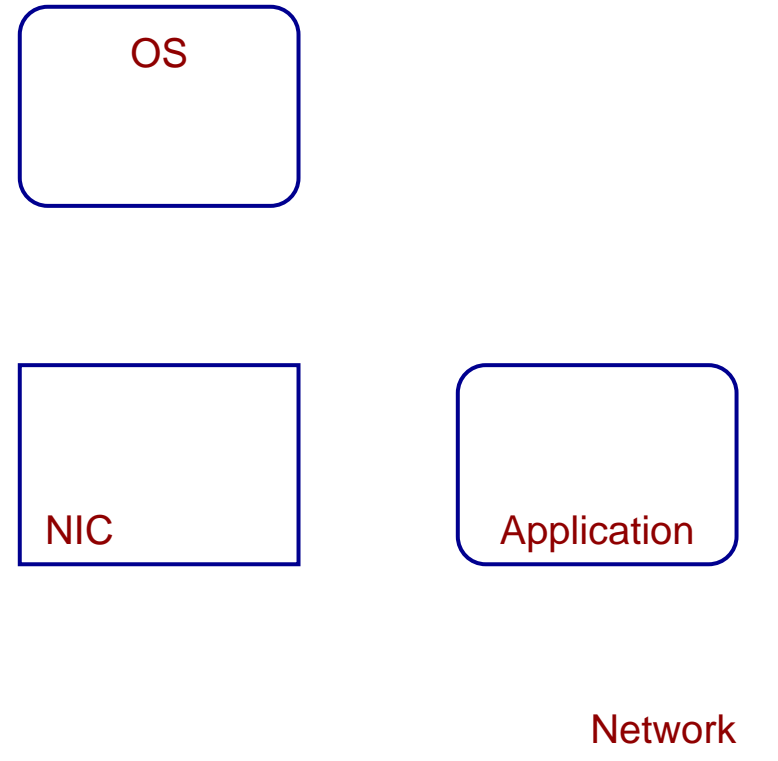
Receive side

- Packet arrives
- Packet to OS
- Interrupt OS
- Data to app
- Schedule app

- High Latency
- Memory Copy
- High Overhead

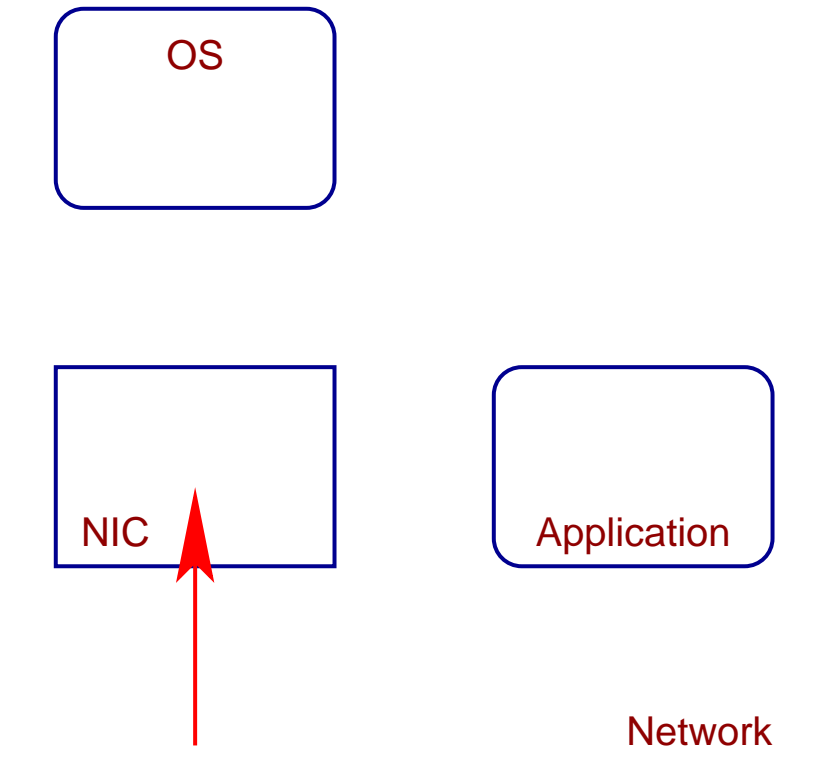


OS Bypass Receive side



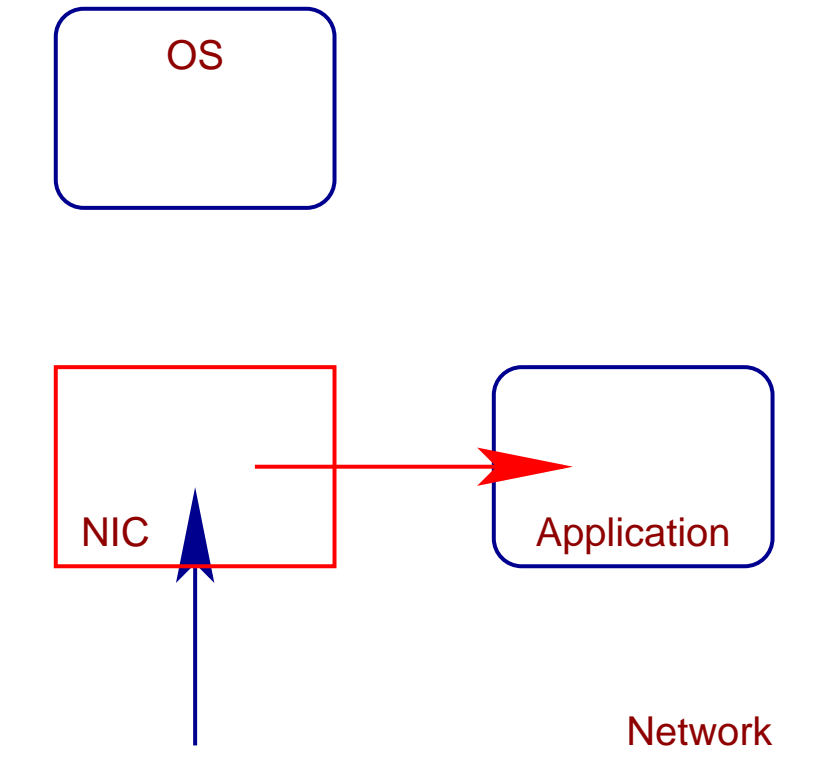
OS Bypass Receive side

- Packet arrives



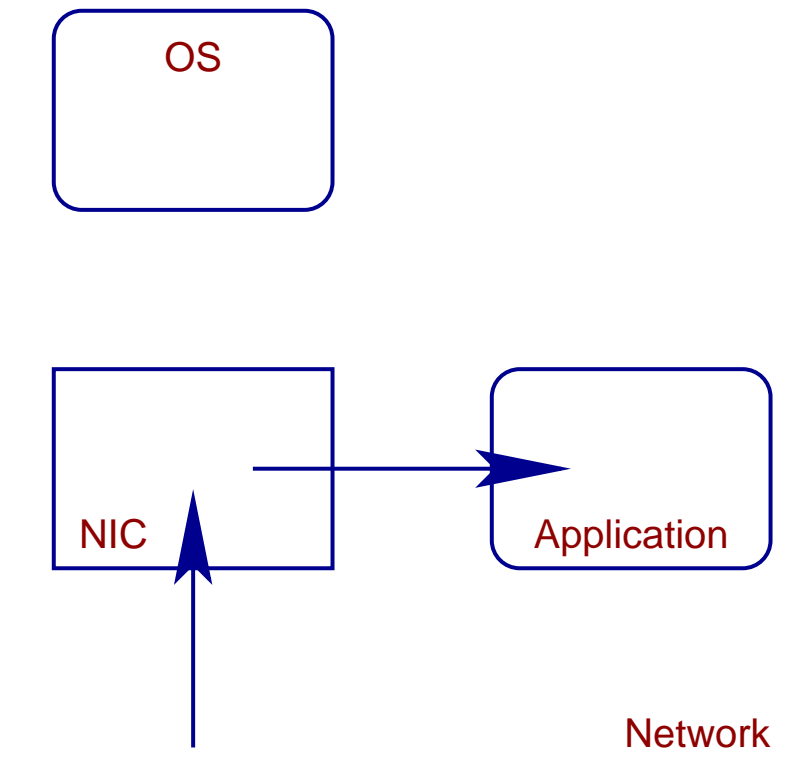
OS Bypass Receive side

- Packet arrives
- Packet to app

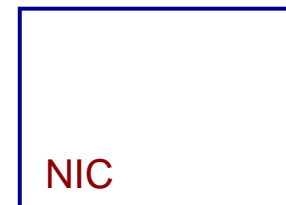


OS Bypass Receive side

- Packet arrives
- Packet to app
- App must poll



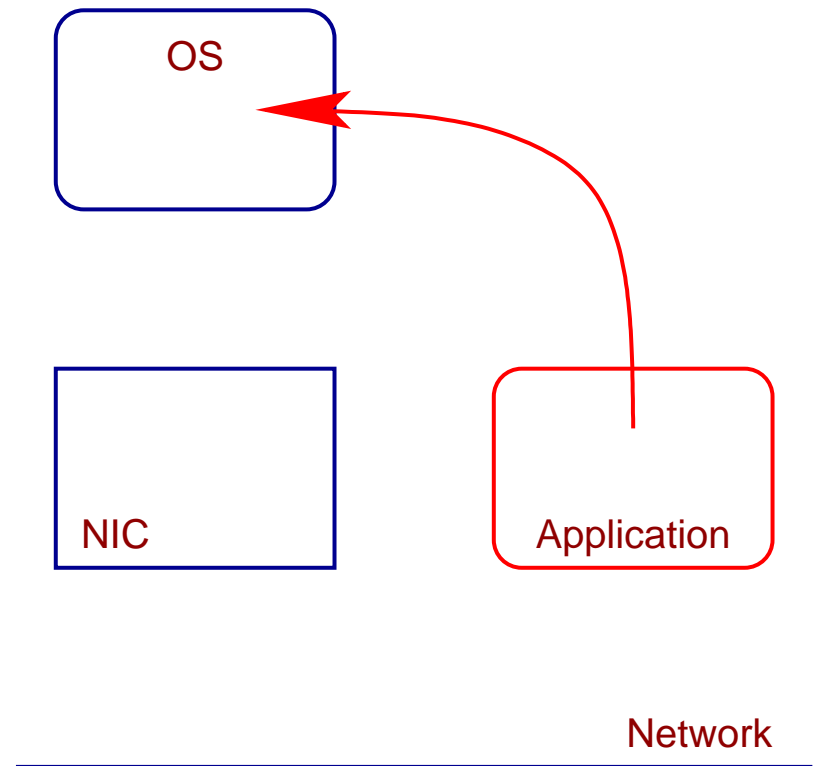
Splintered Receive



Network

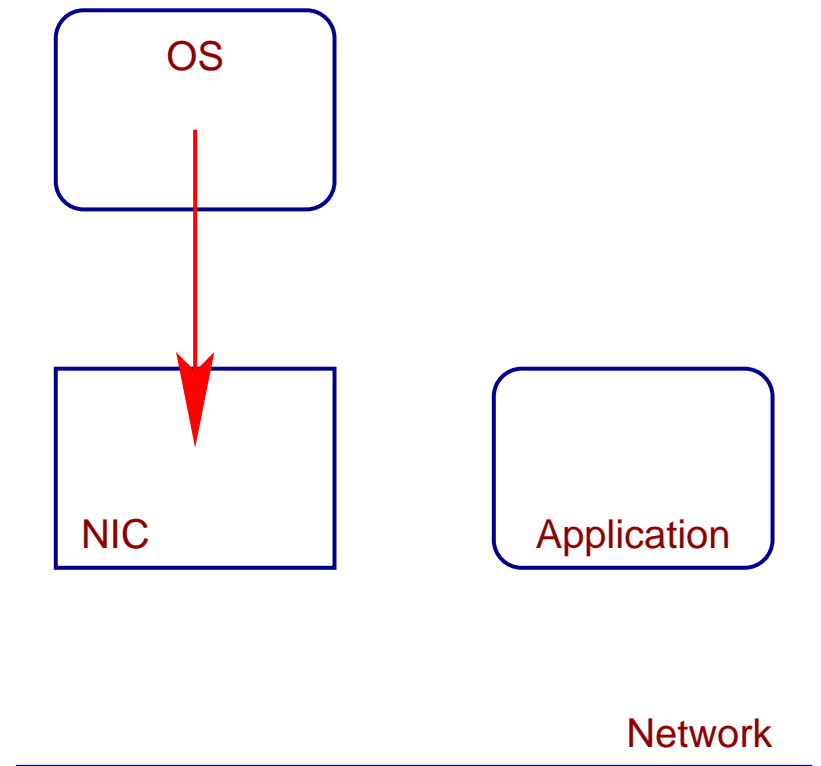
Splintered Receive

- App posts receive



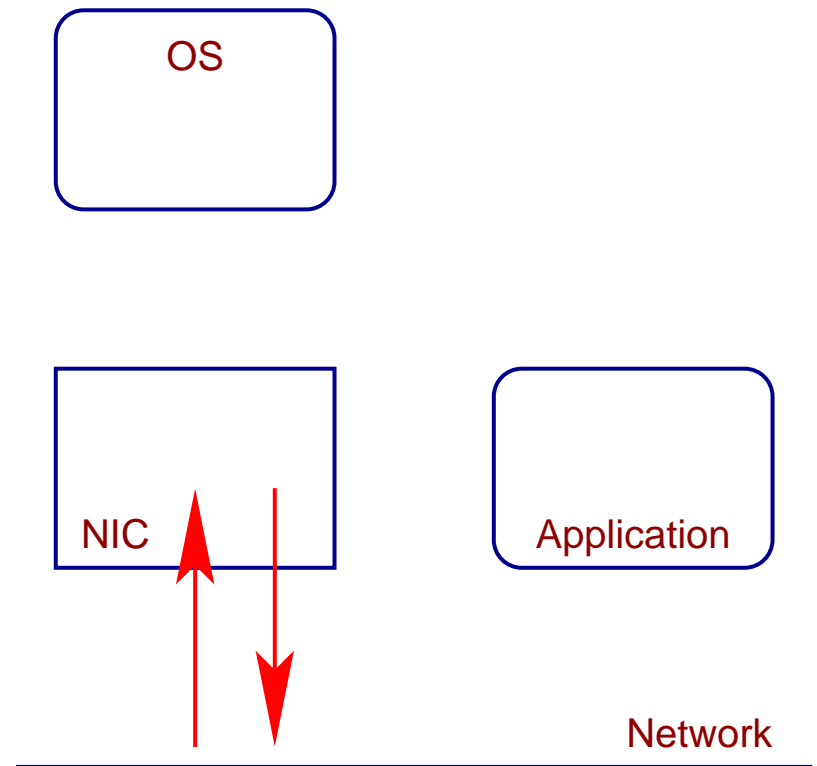
Splintered Receive

- App posts receive
- OS puts descriptor on NIC



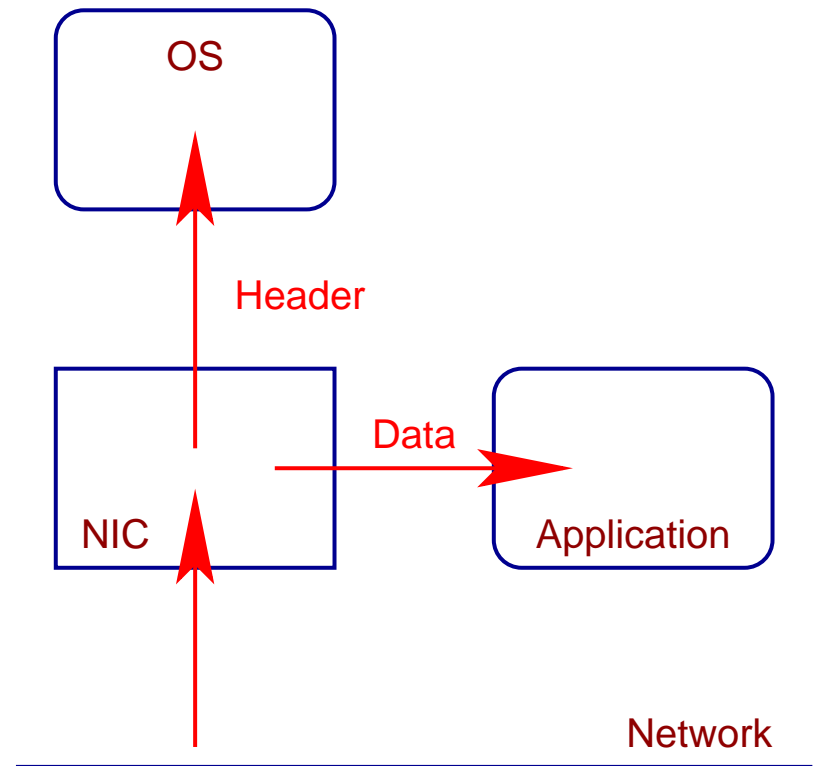
Splintered Receive

- App posts receive
- OS puts descriptor on NIC
- RTS arrives, CTS sent



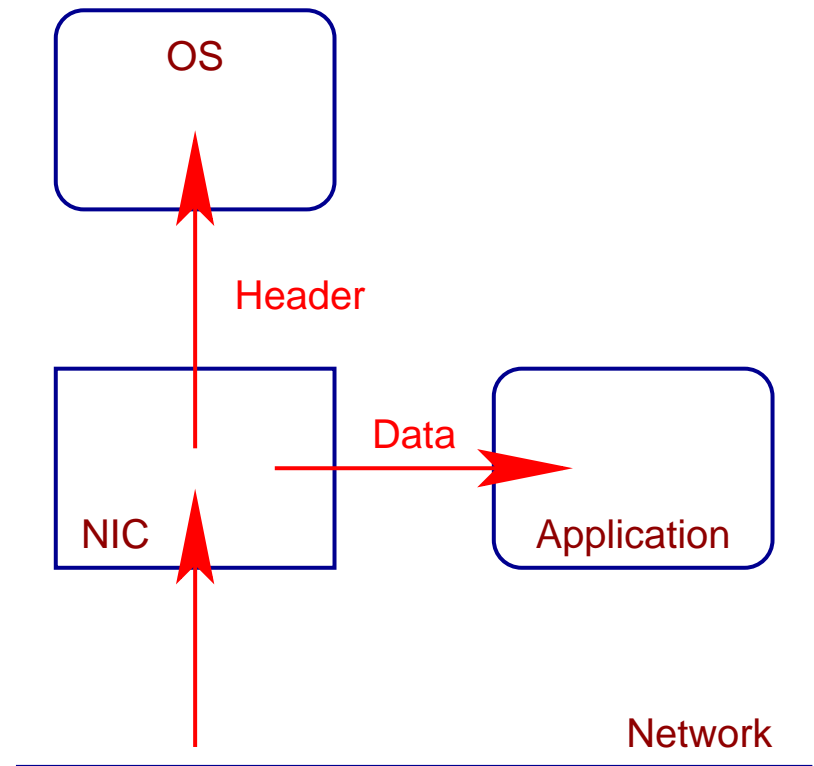
Splintered Receive

- App posts receive
- OS puts descriptor on NIC
- RTS arrives, CTS sent
- Data packet



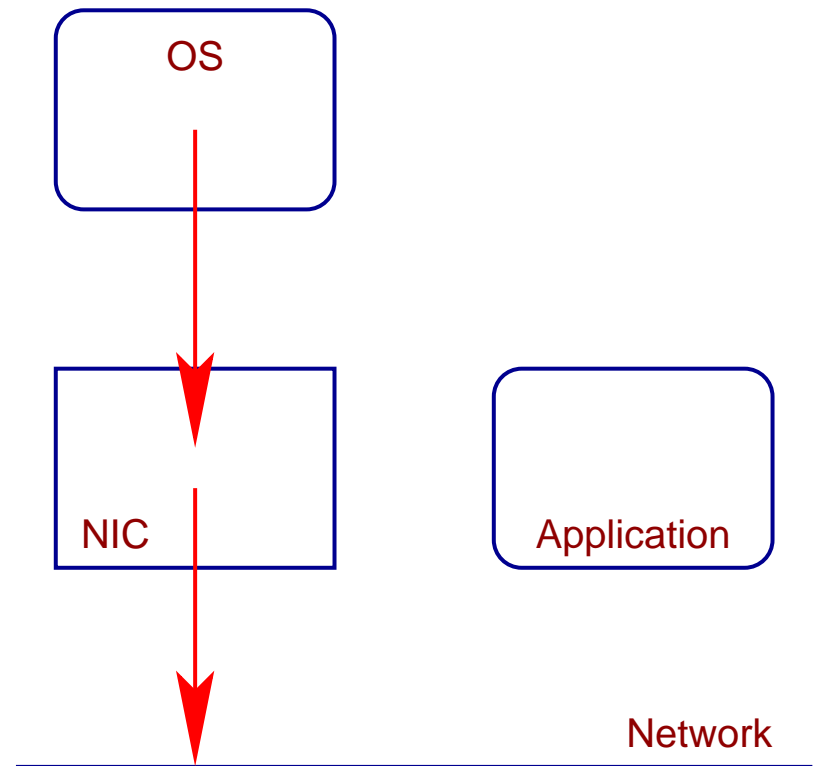
Splintered Receive

- App posts receive
- OS puts descriptor on NIC
- RTS arrives, CTS sent
- Data packet
- etc



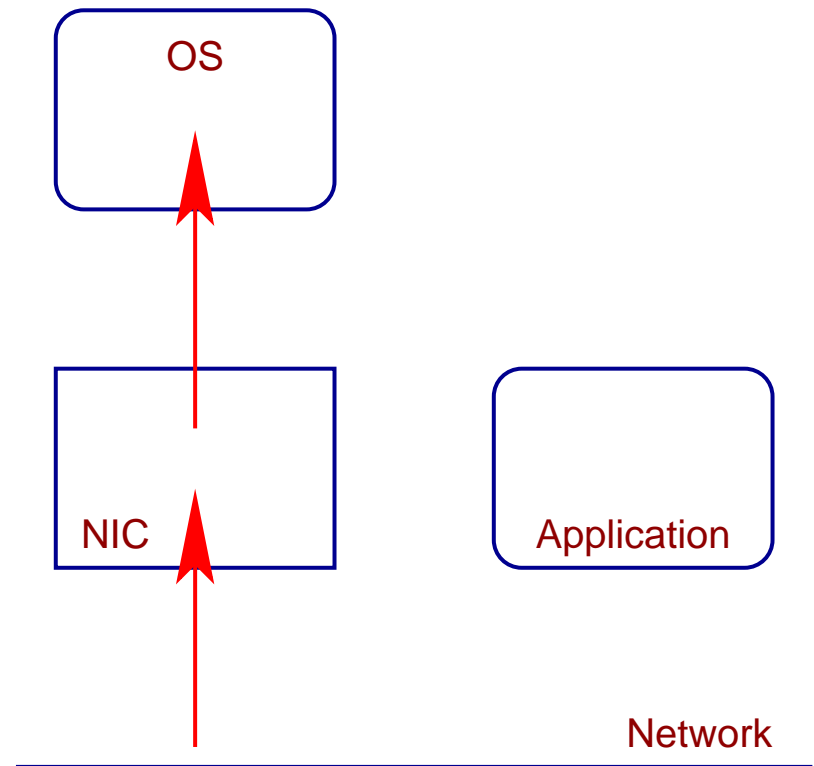
Splintered Receive

- App posts receive
- OS puts descriptor on NIC
- RTS arrives, CTS sent
- Data packet
- etc
- ACK sent



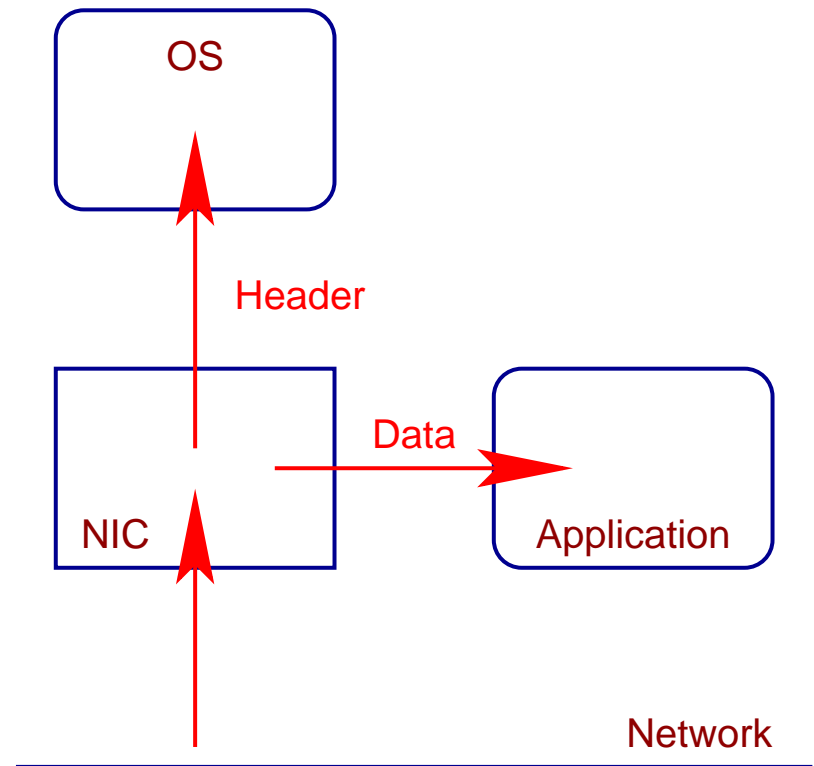
Splintered Receive

- App posts receive
- OS puts descriptor on NIC
- RTS arrives, CTS sent
- Data packet
- etc
- ACK sent
- Cleanup arrives



Splintered Receive

- App posts receive
- OS puts descriptor on NIC
- RTS arrives, CTS sent
- Data packet
- etc
- ACK sent
- Cleanup arrives
- small amount of functionality



Other Examples

- **How to use multiple processors**
 - heater mode
 - message co-processor node
 - compute co-processor mode
 - virtual node mode
- **Demand page virtual memory**
 - make sure it doesn't dominate the design
 - replaceable page replacement strategies
- **Support for message driven computation**

Other Examples of Moving Functionality Around

- NFS: move file system implementation to remote servers
- modular kernels: dynamically move functionality into the kernel
- micro-kernels: move everything to user space
- extensible kernels: allow user code to run in kernel space
- nano (exo) kernels: really move everything to user space

Thoughts on Faults

- Realize that anything can fail and you may not know that it failed
- Try not to slow down the fast case
- Try to make things reliable
- Develop a “faulty mode” for testing robustness