

Distributing OS Functionality to Enhance Application Performance

March 2002

Arthur B. Maccabe

maccabe@cs.unm.edu

Scalable Systems Lab

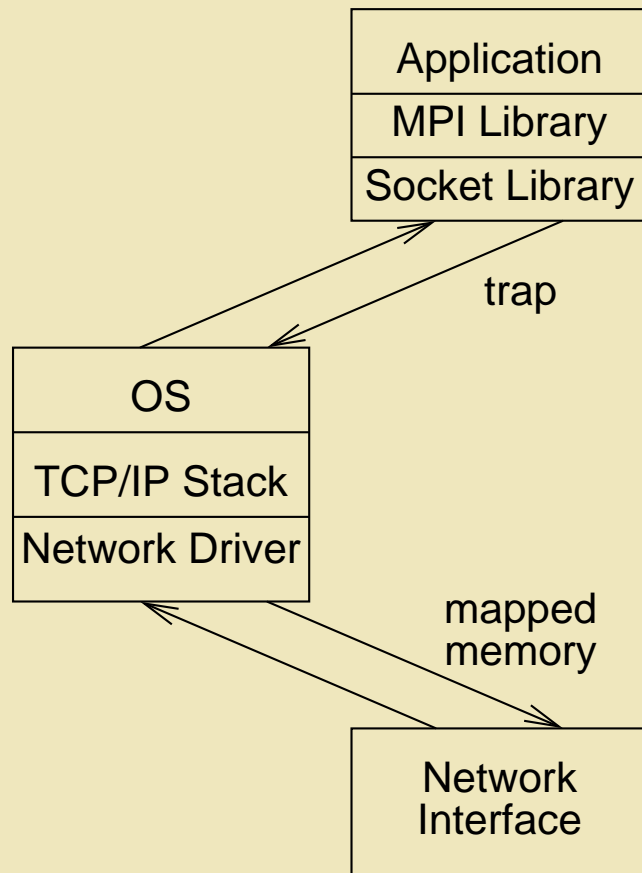
Computer Science Department

The University of New Mexico

Where you place functionality matters

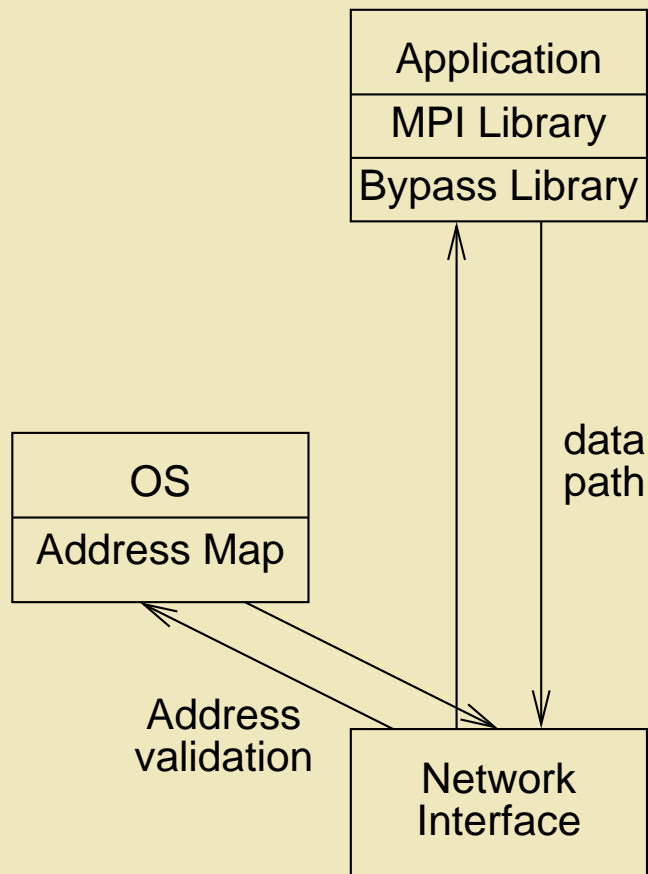
- OS Bypass in MPI Implementations
- OS Offload, a different perspective
- Application Offload
- Double buffering benchmark
- Application Bypass benchmark

Basic MPI Stack



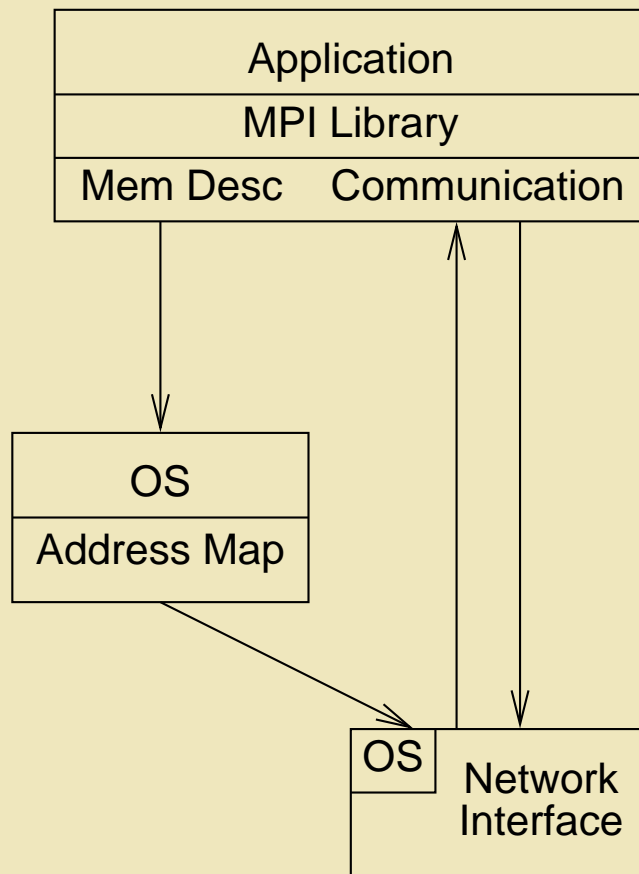
- **Memory copies**
 - within the application
 - in kernel stack
 - between NIC and kernel
 - between application and kernel
- **Latency**
- **Interrupt pressure**
- **Overhead**

OS Bypass



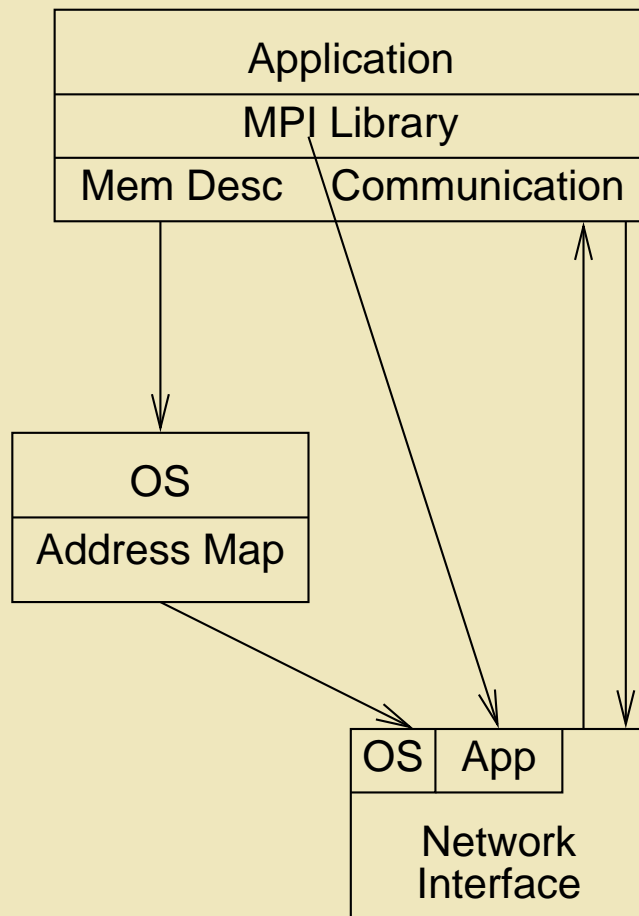
- **Memory copies**
 - between NIC and application
 - within application
- **Latency**
- **Interrupt pressure**
- **Overhead**

OS Offload



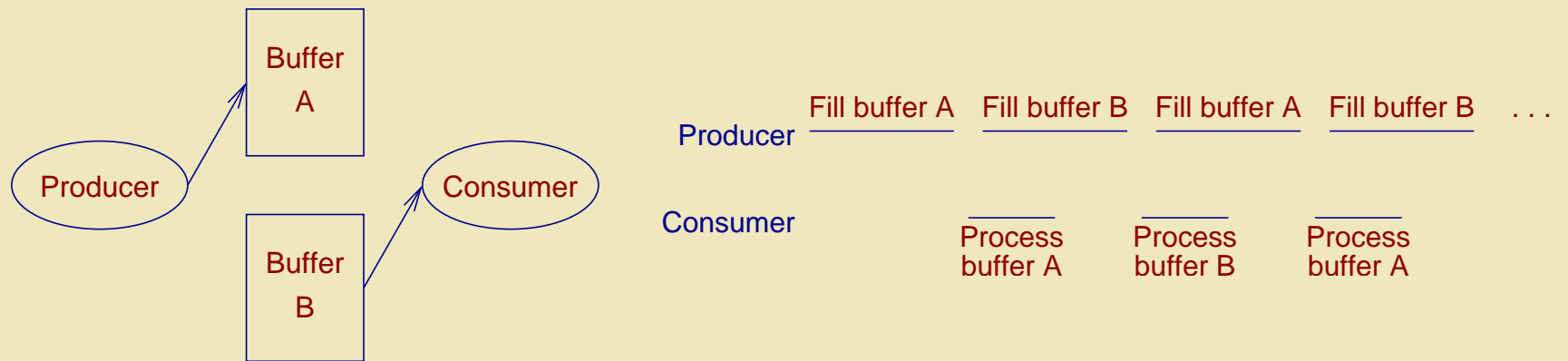
- Different Perspective
- You never really bypass the OS
- A bit of the OS goes onto the NIC

Application Offload



- Why not offload part of the Application?
- Just enough to decide where to put messages
- matching

Double Buffering



- Overlap communication with processing
- Latency hiding

Double Buffering

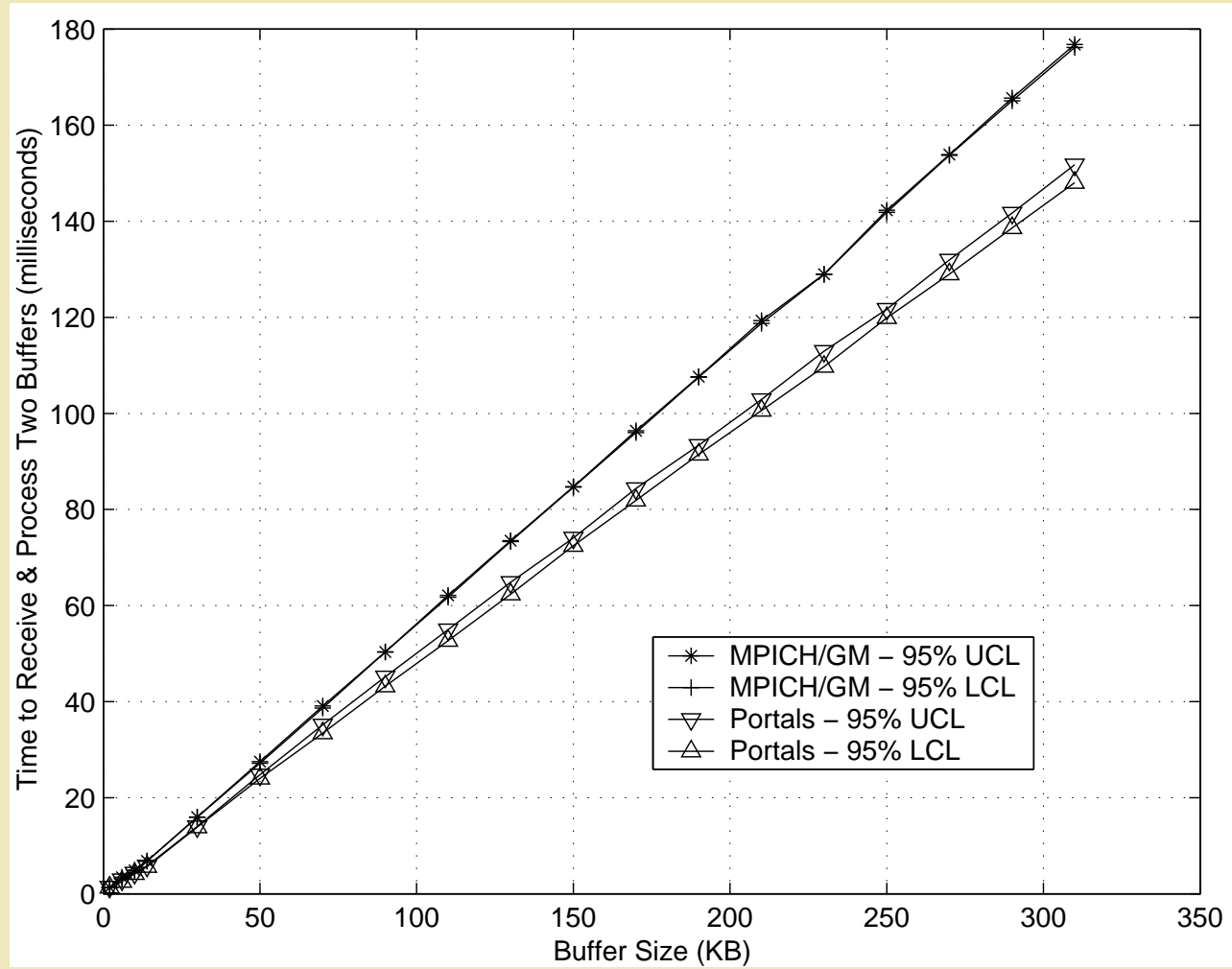
Producer

```
for( i = 0 ; i < n-1 ; i++ ) {  
    fill A; wait CTS A;  
    isend A;  
  
    fill B; wait CTS B;  
    isend B;  
}
```

Consumer

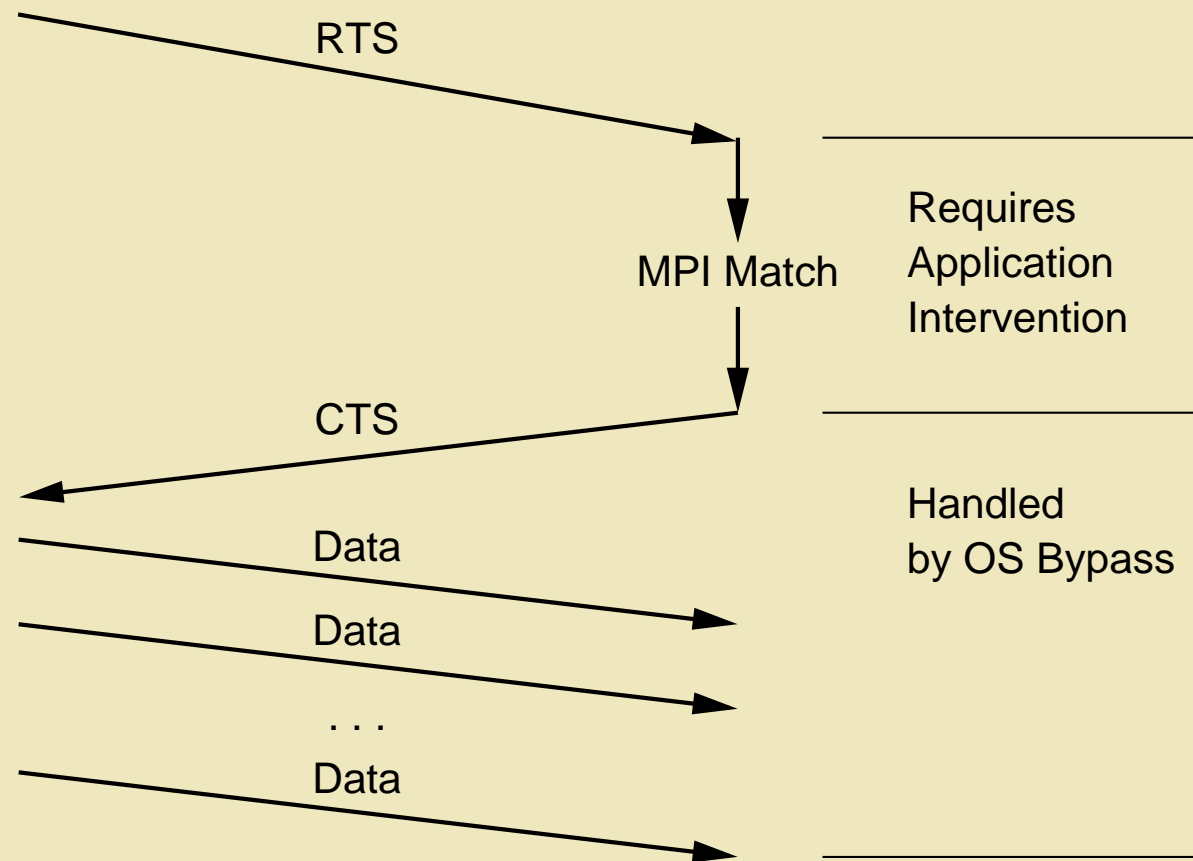
```
ireceive A; isend CTS A;  
ireceive B; isend CTS B;  
for( i = 0 ; i < n ; i++ ) {  
    wait A; sum A;  
    ireceive A; isend CTS A;  
  
    wait B; sum B;  
    ireceive B; isend CTS B;  
}
```

Double Buffering Performance

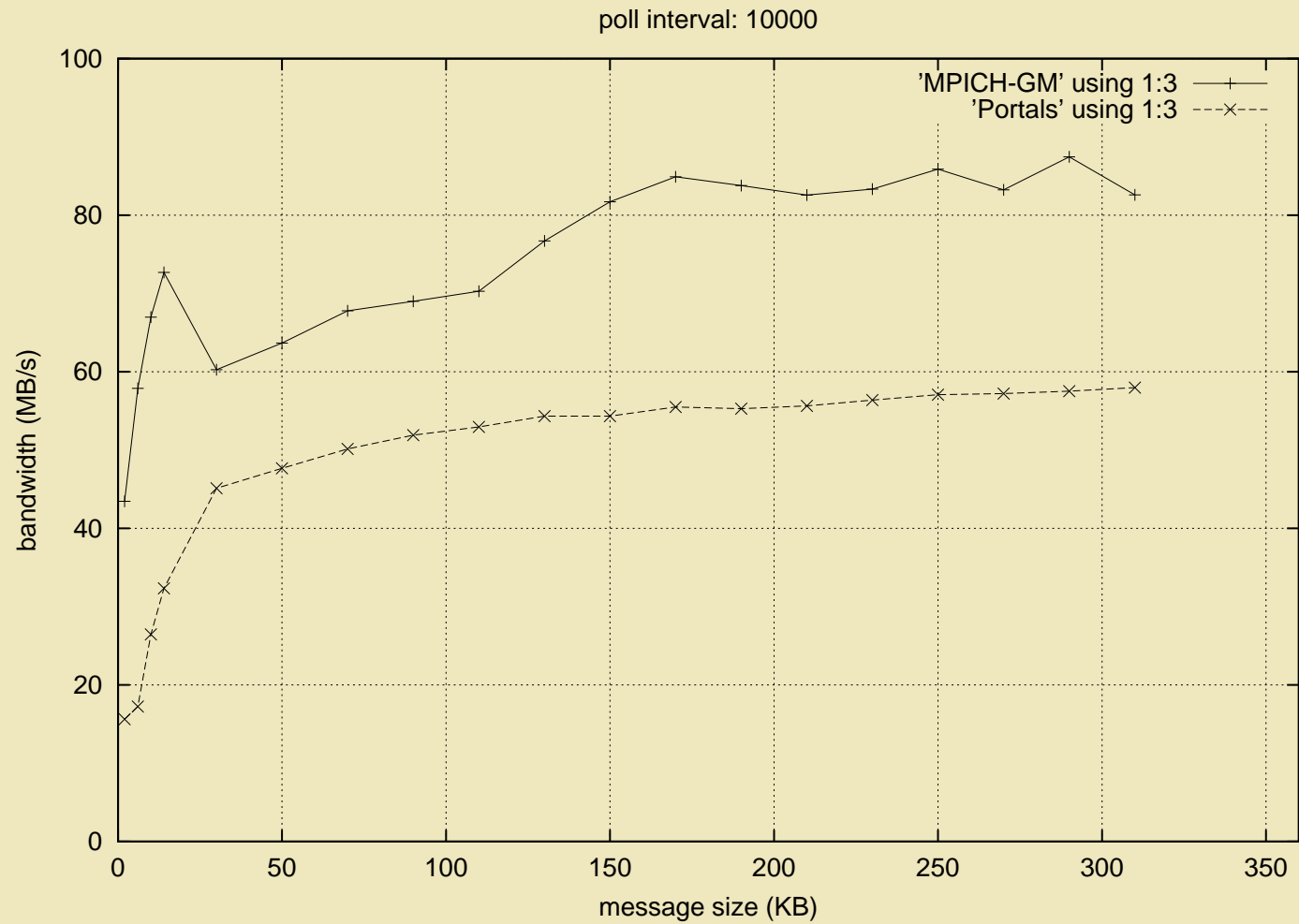


Why is Portals Better?

Long message protocol

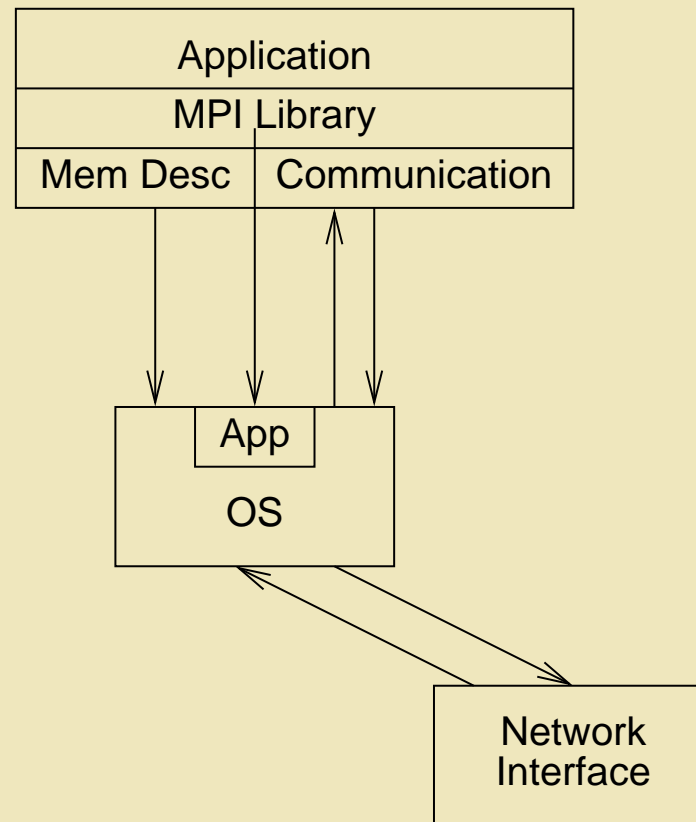


It's even better than it looks! Portals Bandwidth

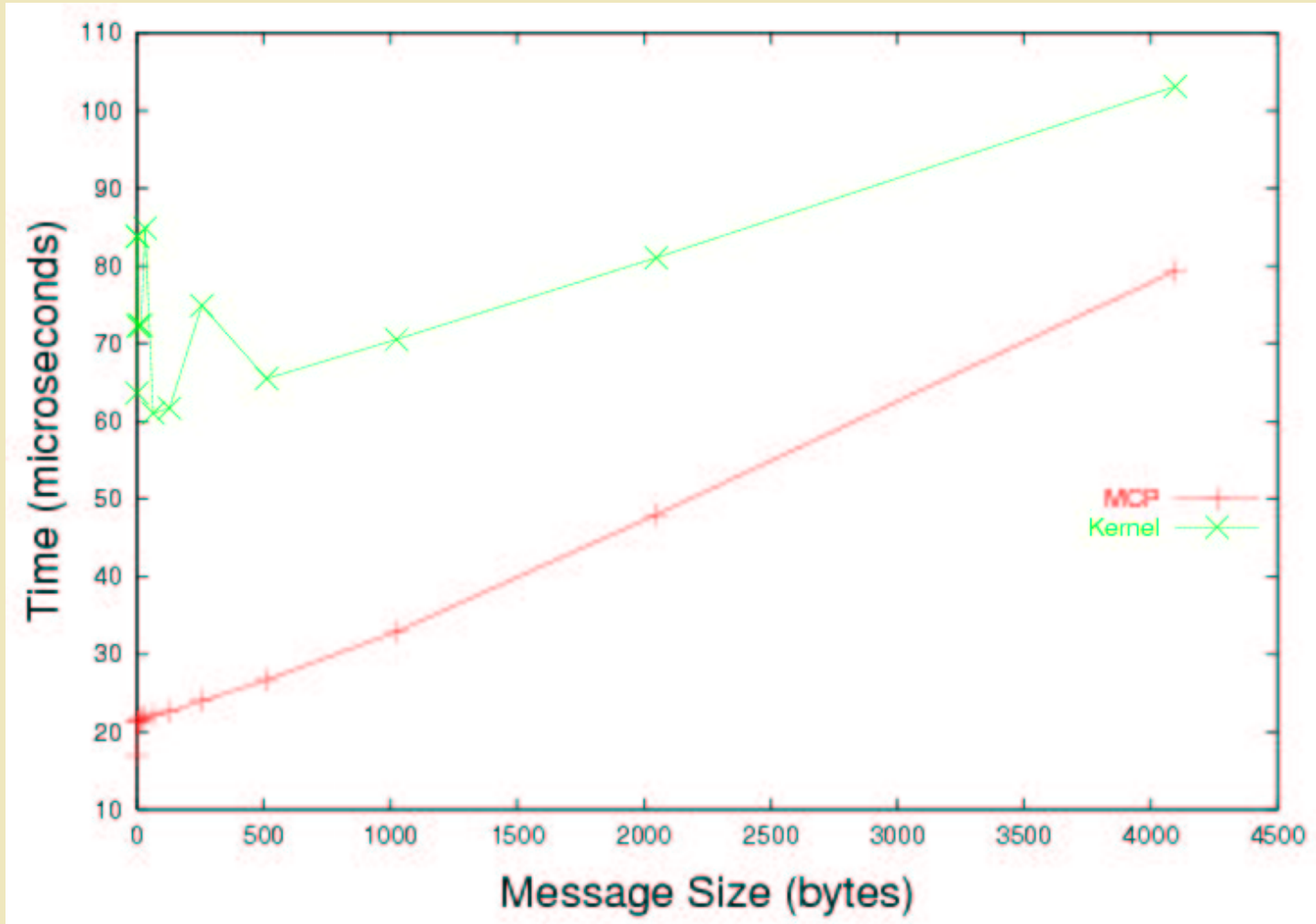


Why is Bandwidth so Bad?

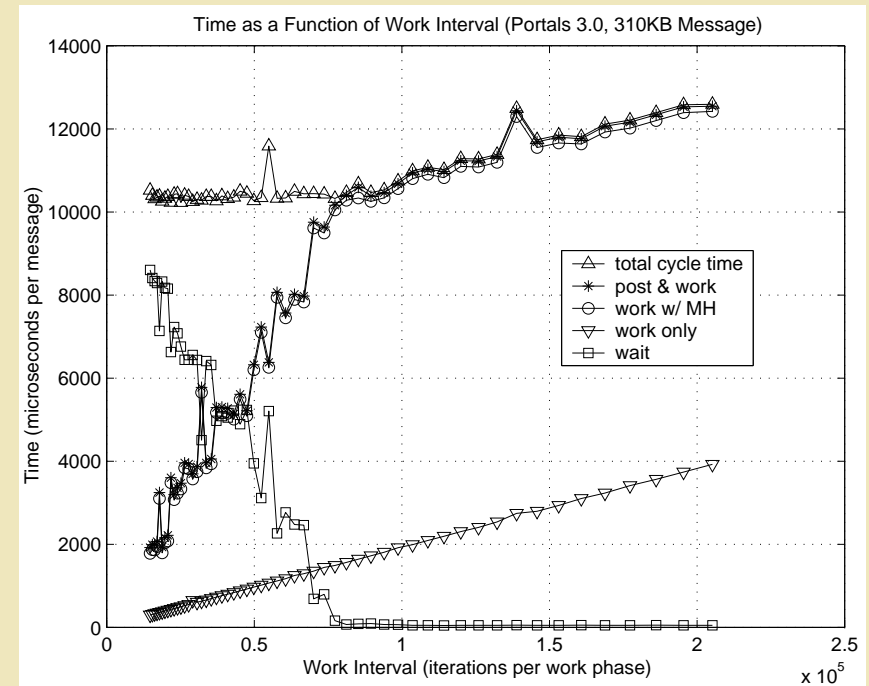
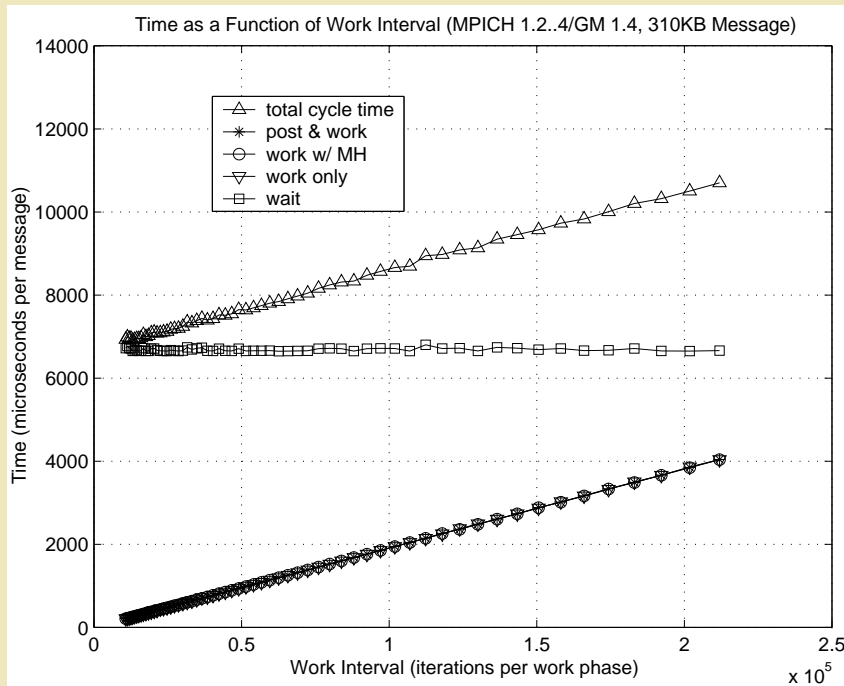
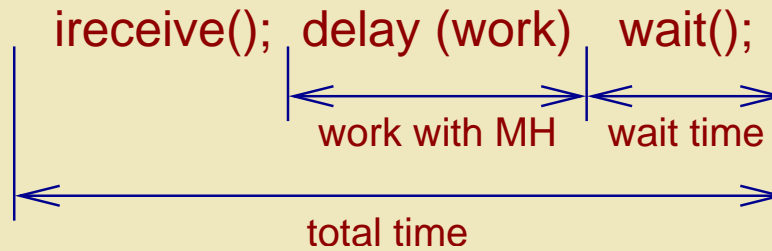
The current implementation uses the kernel for everything



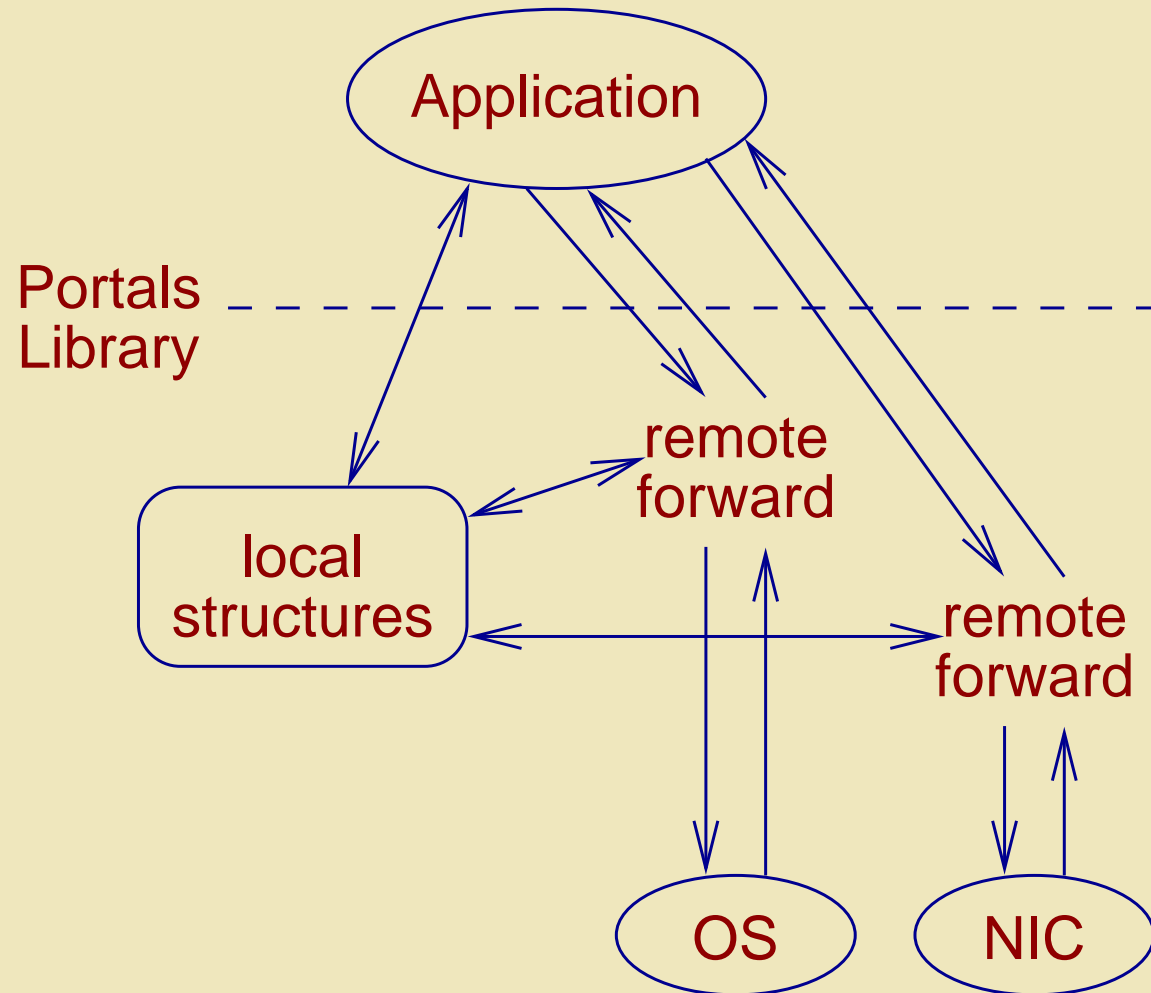
Portals on the NIC



Post-Work-Wait



Portals Implementation Strategy



Placing OS Functionality

- **Host processor**
 - Supervisor mode
 - User mode
- **Co-processor**
 - Compute co-processor (threads)
 - Message co-processor (NIC)
- **Server node**
 - File server
 - TCP (socket) server

Conclusions

- It's easy to design low-level protocols, the trick is effectively supporting higher level protocols
- Latency hiding is critical for applications
- Placement of functionality matters
 - liberating perspective
 - many other opportunities

Acknowledgements

Supported by Sandia National Labs and CSRI

- Bill Lawry and Riley Wilson
- Ron Brightwell and Rolf Riesen
- Patricia Gilfeather, Edgar Leon, Dennis Lucero, Carl Sylvia, and Wenbin Zhu