
Federal University of Santa Catarina
Laboratory for Software/Hardware Integration

# High-Performance Communication System based on Generic Programming

André Luis Gobbi Sanches
Fernando Roberto Secco
Antônio Augusto Fröhlich

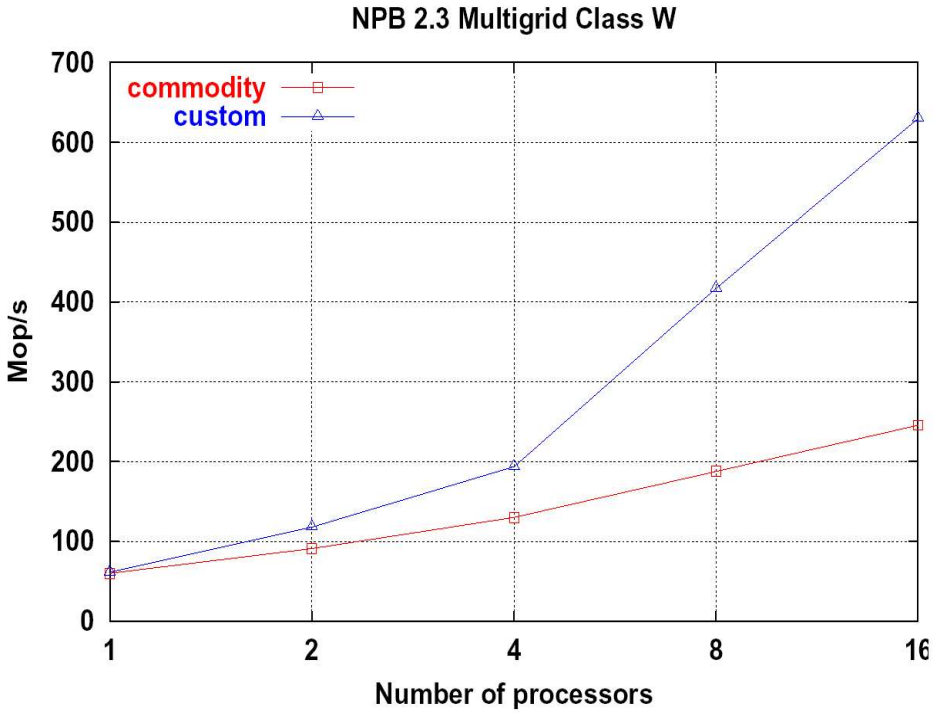
snow@lisha.ufsc.br
http://snow.lisha.ufsc.br/

# Outline

- Motivation
- The SNOW Project
- Application-Oriented System Design
- The EPOS System
  - Overview
  - Communication system
  - Performance
- Conclusions

# Motivation

- Cluster computing performance as of 2000
  - a cluster of commodity workstations
  - running a same application
  - on commodity and custom run-time systems



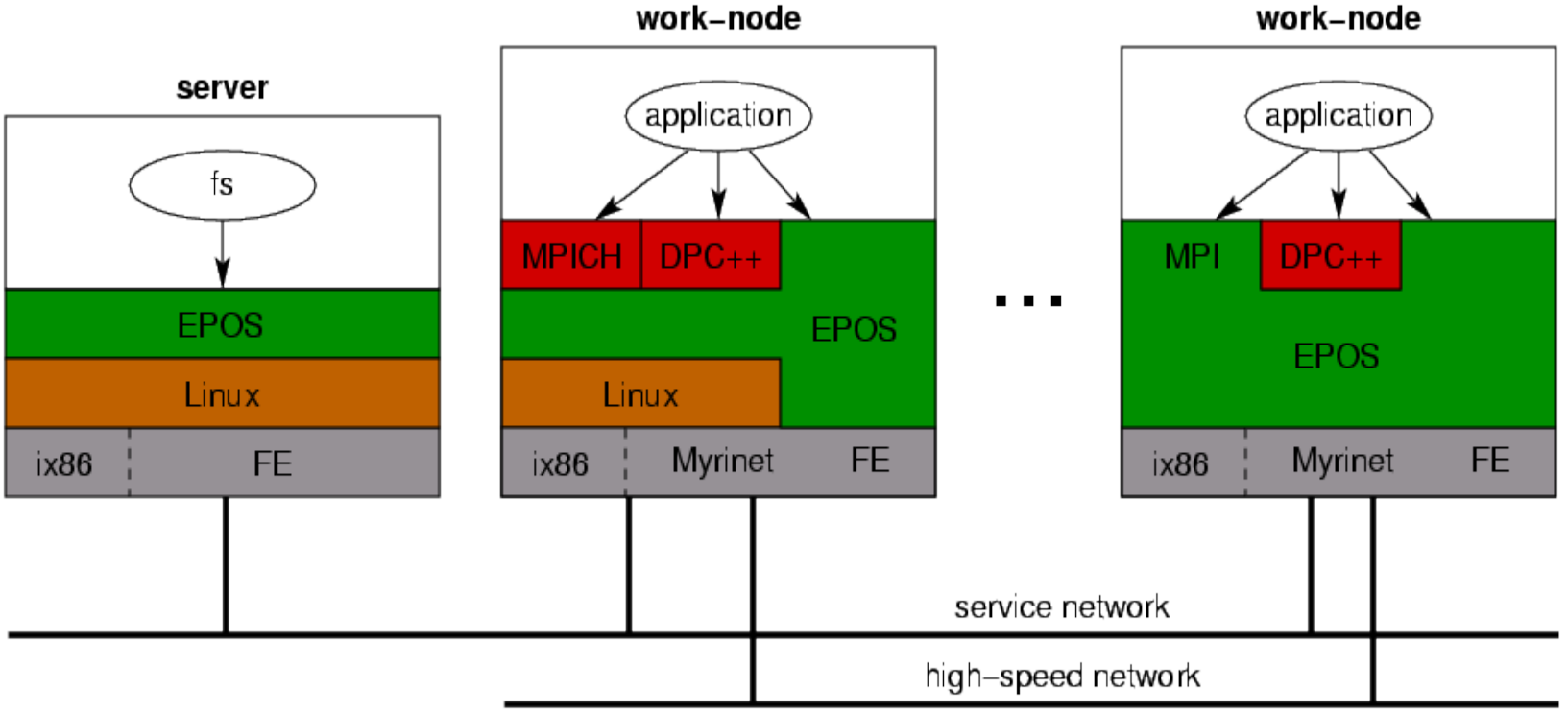
# Commodity Hardware and Custom RTS

- Commodity X Custom hardware
  - Convergence
- Commodity X Custom run-time systems
  - Commodity
    - multi-{user,tasks,...}, interactive, web-aware
    - more distributed than parallel
  - Custom
    - high-performance and low latency
    - specially designed to support parallel computing
- Clusters do need dedicated RTS in order to be as efficient as traditional supercomputers

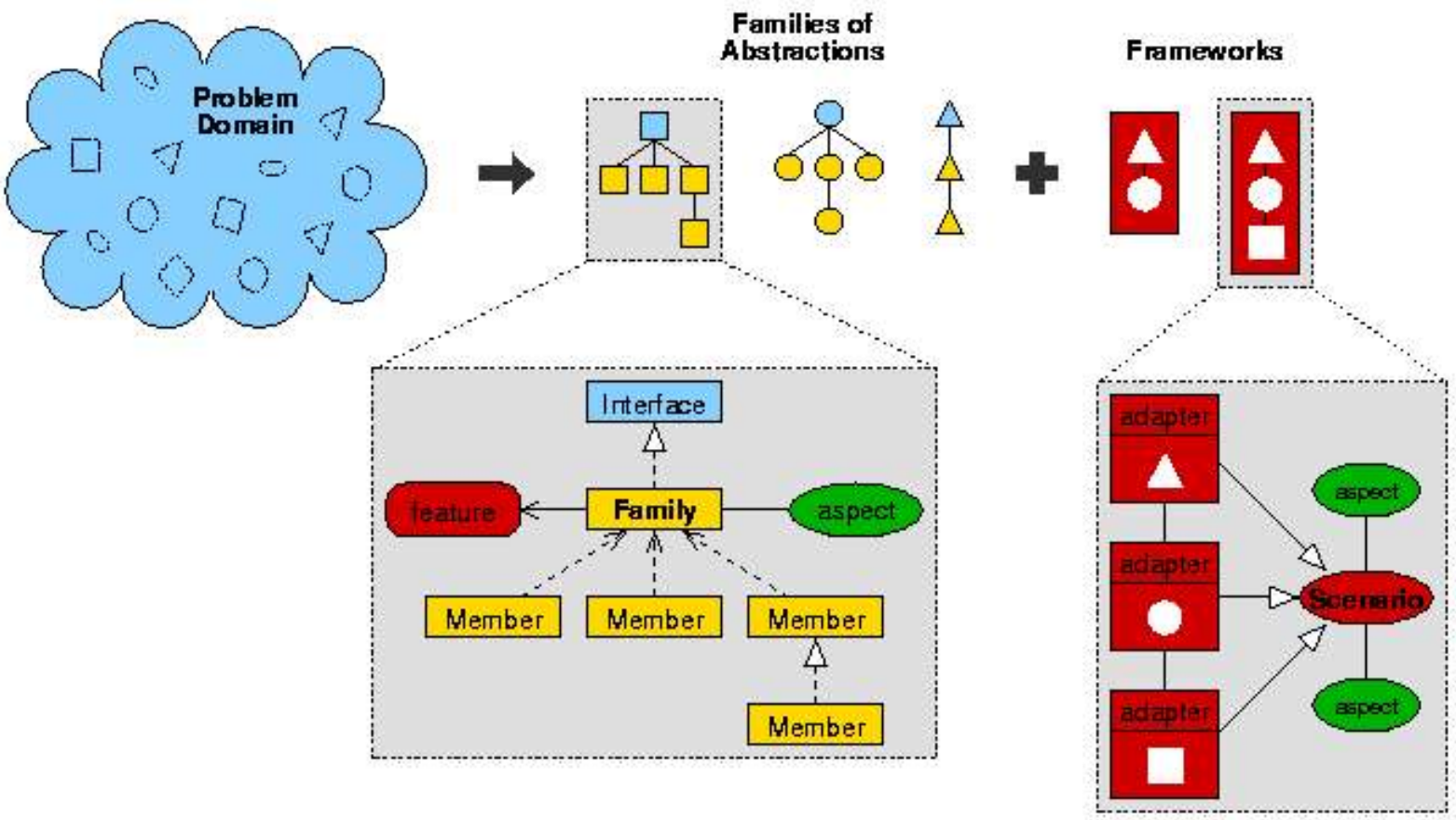
# The SNOW Project

- Aims at developing an application-oriented parallel programming environment for clusters of workstations
  - run-time support system (EPOS/UFSC/FhG)
  - programming language (DPC++/UFRGS)
  - management tools (CODINE/SUN)
- Bringing cluster performance closer to traditional supercomputers
- Validated by selected parallel applications
  - computational biology
  - complex industrial processes

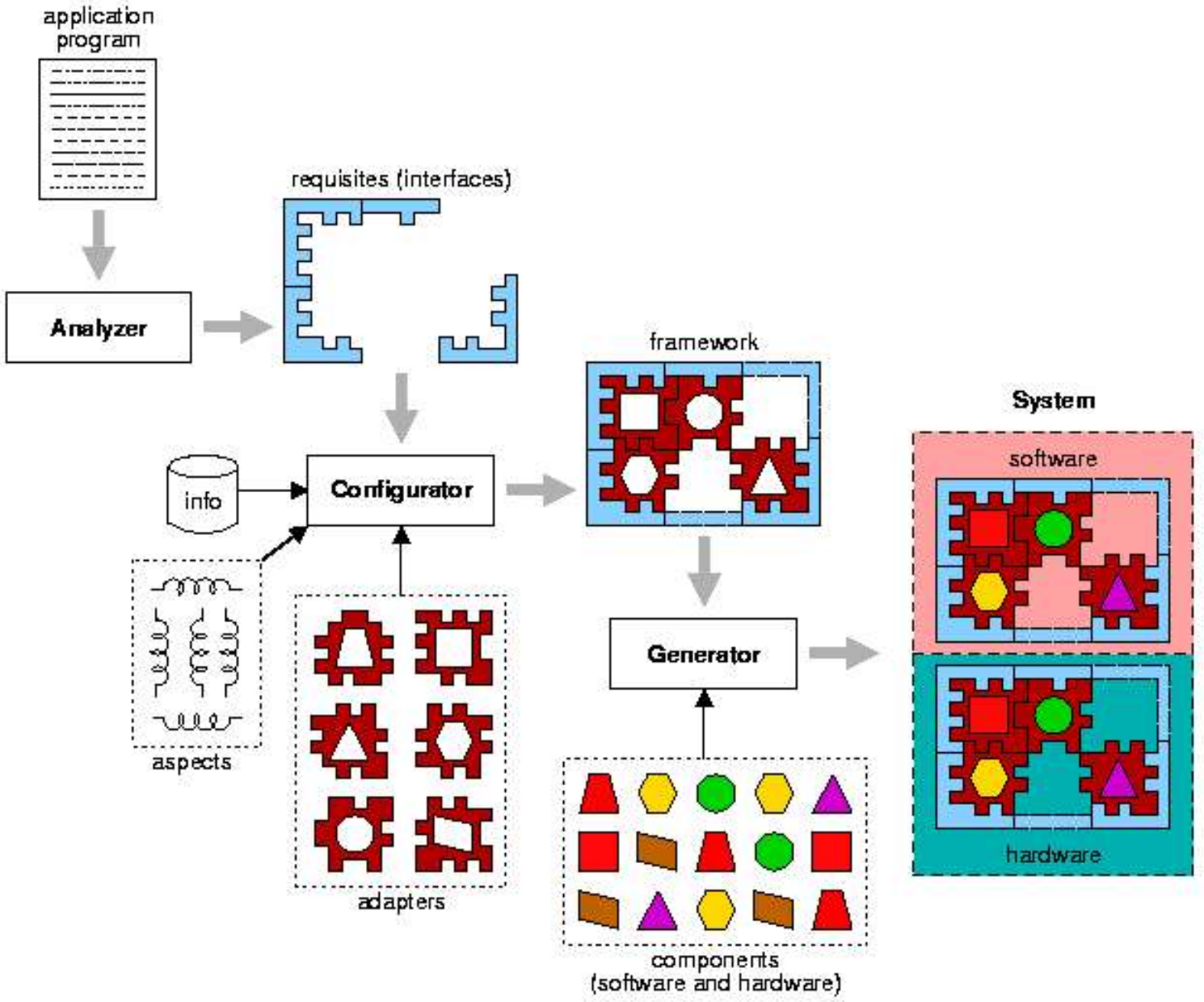
# Overview of a SNOW Cluster



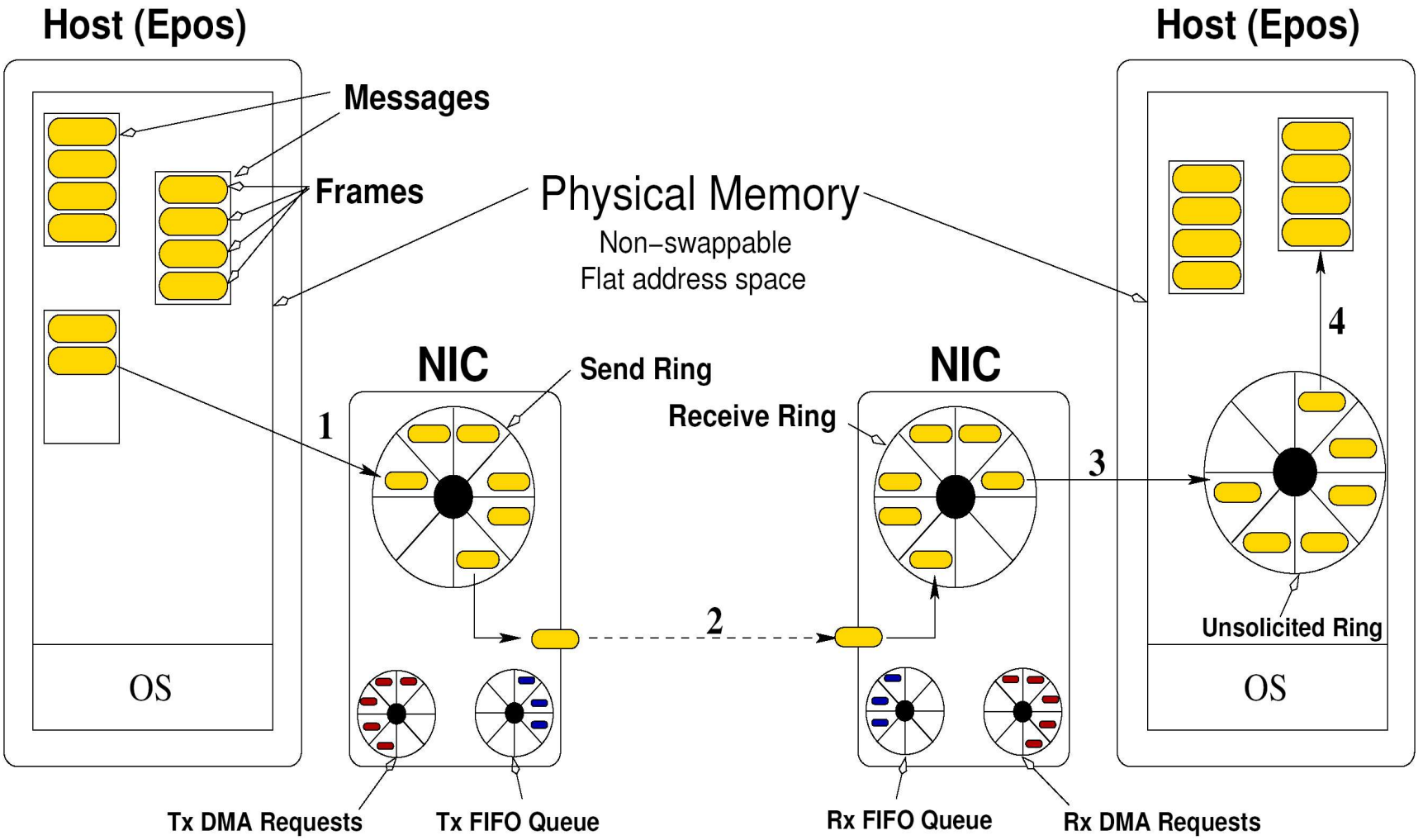
# Application-Oriented System Design



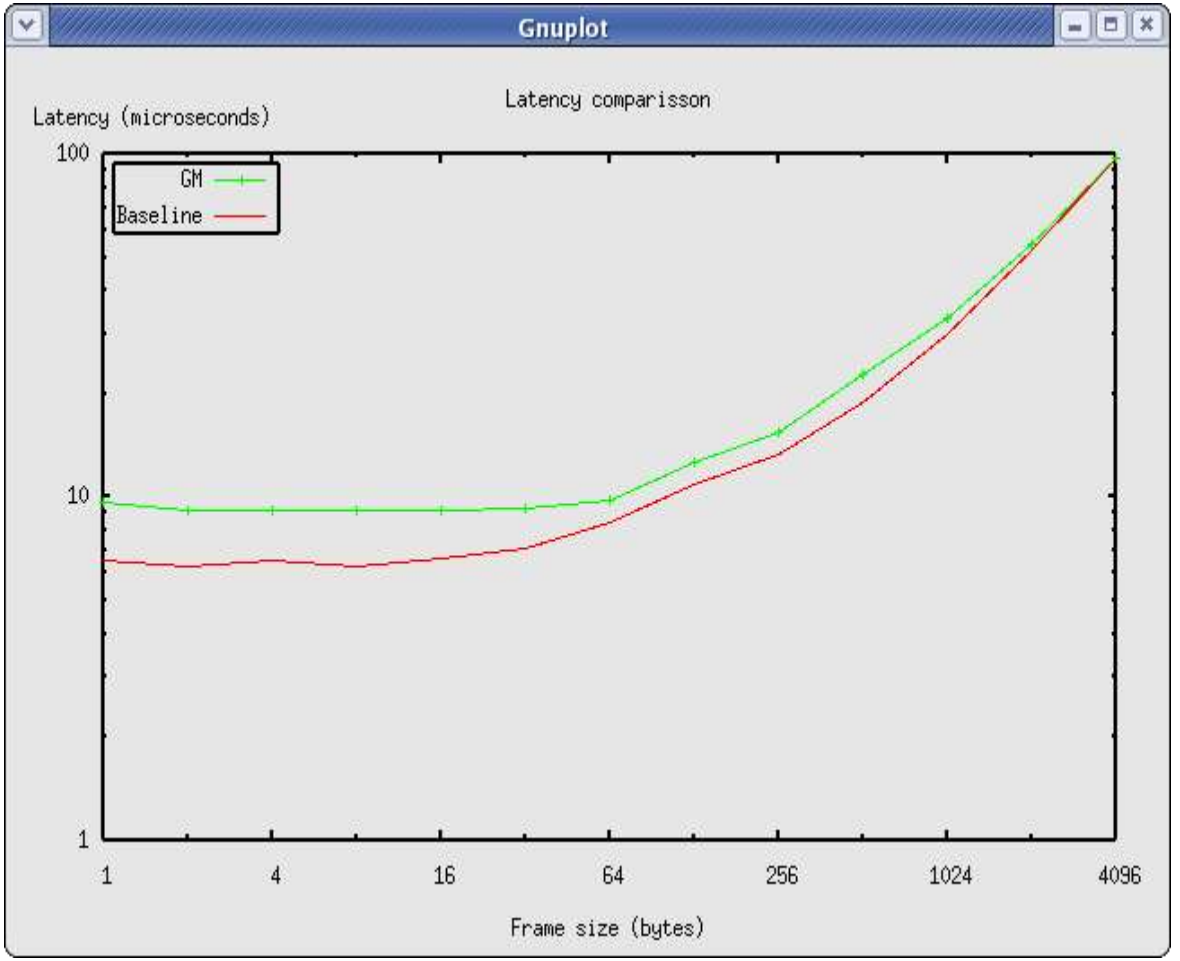
# The EPOS System



# EPOS Communication System for Myrinet



# EPOS Communication System Latency



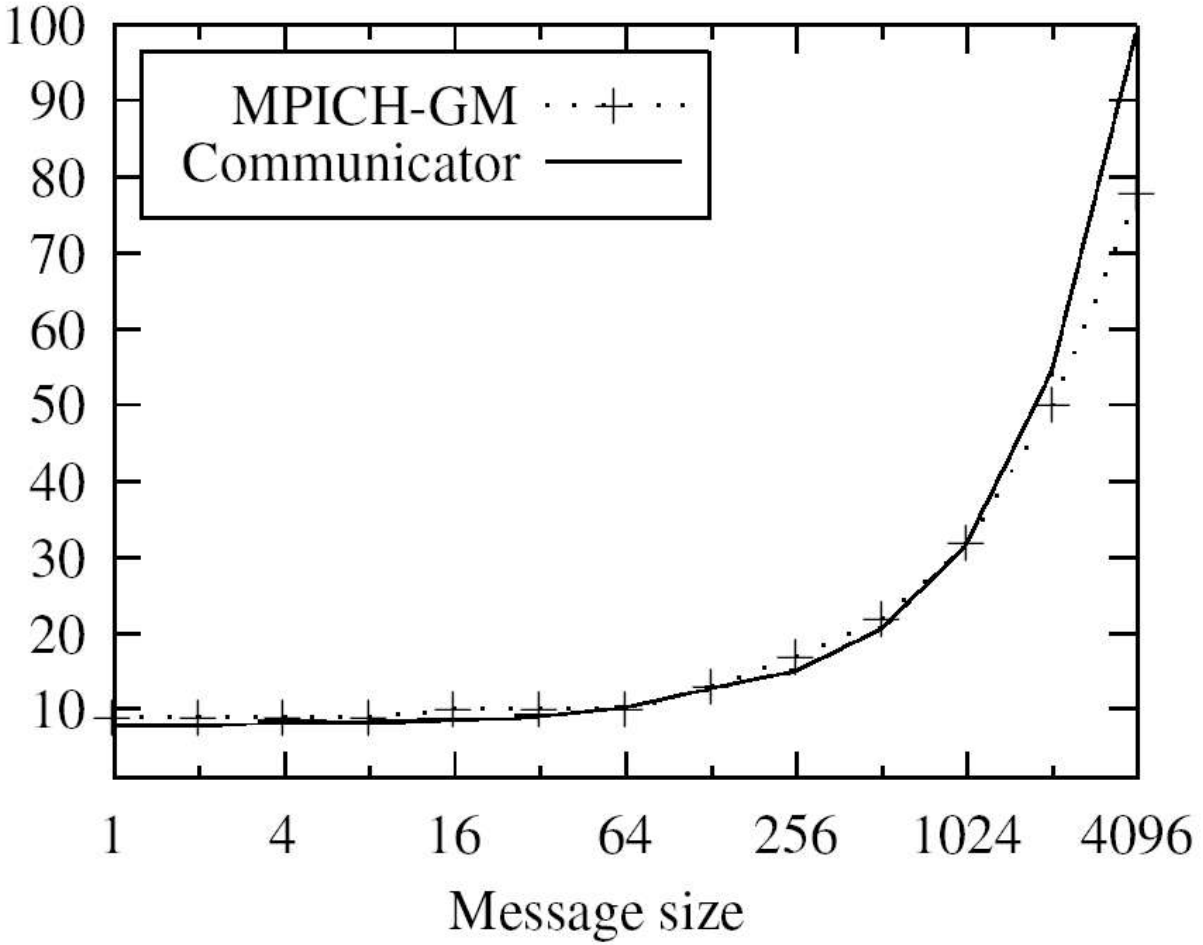
# Traditional MPI Implementations

- Thick layers of software
  - Sometimes designed as a middleware
  - MPICH
    - Channel Interface (5 functions, data link)
    - Abstract Device Interface (> 40 functions, point-to-point)
    - Device Independent Layer (collective communication, data types)
  - MPICH-GM
    - From the 120.000 lines of code, 30.000 are Myrinet specific
- On badly designed operating systems
  - That must be often bypassed
  - Although implementing virtually everything that would be necessary to deliver the Message Passing Interface

# MPI in EPOS

- Basically a matter of interface correlation
  - MPI p2p functions are translated into invocations of EPOS communication system objects through AOP
    - Envelope -> data container
    - Communicator -> user end-point
    - Channel -> protocols
    - Network
- An specific header was added to handle MPI message delivery semantics
- Collective operations are generic programs
  - executed on behalf of “communicators”
  - to generate appropriate MPI headers

# EPOS-MPI X MPICH-GM




- EPOS-MPI ping-pong => ~20 Kb including OS
- MPICH-GM ping-pong => 400 Kb + OS

# Conclusions

- AOSD enables the development of component-based RTS that can be more easily adapted to fulfill the requirement of applications than monolithic or micro-kernel based systems
- MPI is just an interface that can be easily realized by properly designed RTS
  - without requiring thick layers of code
  - nor os-bypass
- Less code not only means less overhead, but also fewer bugs