MPI/RT: Design and Implementation of a Real-Time Message Passing Interface*

Arkady Kanevsky (MITRE)
Anthony Skjellum (MSU)

July 15, 1997

DARPA Quorum Meeting

*Work at MSU and MITRE supported by DARPA under contracts from USAF Rome Laboratories.

Overview

- General History of MPI
- Technical Highlights of MPI/RT
- Scope of Specification and Functionality
- Philosophy
- Paradigms
- Clocks
- Implementations (Just a Little)
- Summary/Conclusions
- Online Access Information

General MPI History

- MPI is a de facto Message Passing Standard
- MPI-1 completed 5/94, revised 6/95 & 7/97
- MPI-2 involves extensions to MPI-1
- MPI/RT is a part of MPI-2 process but not part of standard to be released 7/97 - real-time specification and features
- MPI/RT is not specifically a subset or superset of MPI-1 or MPI-2.
- De facto standards body (MPI-F, 1992-date)

Existing Implementations

- General:
  - IBM, SGI and CRAY, HP and Convex, Fujitsu, NEC, Intel, SUN, MPI Software Technologies, Parsytec
- Embedded:
  - Mercury, Alacron, Hughes, Avalon, CSPI, MPI Software Technology, Lincoln-Labs, Lockheed-Martin

MPI/RT Forum Subcommittee

- Constituted informally in March/April, 1995
- Government, University, Commercial participants
- Has had “special status” in MPI process to allow time for concepts to mature properly
- Has broad scope in terms of augmenting and modifying MPI
- Has focused goal of providing standardized programming environment for RT
- Met at all MPI Forum meetings, and in between
- Like rest of MPI, an open forum, now stand-alone

MPI/RT Technical Highlights

- Embraces MPI-1 as base API and programming strategy, but modifies it for real-time
- Higher optimizability than MPI-1 and MPI-2
- Designed to support several RT paradigms
- Uses channels (point-to-point and collective) with quality of service
- Provides timeouts where appropriate
- Has profile stages to support cost-effective vendor development

MPI/RT Tech. Highlights, II.

- Object-oriented design
  - Cloning and composition
  - Templates, objects, requests
  - Persistent generalized requests
- Deferred early binding philosophy
  - Maximum flexibility to optimize resources
- Attention to multithreaded applications

Scope of MPI/RT Specification

- A full *draft* specification exists, under revision
- Timing requirements of implementation
- API for functions added or modified (to MPI)
- Channel, buffering, and admission test syntax/semantics
- Discussion of three real-time programming models
- Discussion of QoS specifications
- Fault tolerance / fault detection [early stages]
- Instrumentation
- Profiles
- LIS specification, LSB for C, C++, F77/F90 - maybe

Scope of MPI/RT Functionality

- Channels
- Admission Tests
- Reliable, QOS-oriented data transfer and operations.
- Point-to-point operations on channels
- Collective operations on collective channels
- Admissions tests specified in a novel way to the system
- Both on-line and off-line mechanisms for establishing channels+QOS are to be entertained.
- Support for use of base MPI for non-real-time programming within the RT world.

Current Application Development

- Step 1 - Application developers become platform expert, and platform provides message passing API
- Step 2 - Application developers are responsible for achieving desired application performance and real-time guarantees, by experimentation using platform-dependent API
- Inherently platform-dependent, non-portable, and trial-and-error oriented

QoS Guarantees

- Providing QoS guarantee is a difficult schedulability problem, as it involves scheduling different resources:
  - CPUs where send/receiver applications are running
  - Data busses (e.g., PCI setup)
  - DMA engines
  - Memory (SRAM, DRAM)
  - Network interface chips (NICs)
  - Physical network (topology dependencies)
- Different resources are scheduled using real-time paradigms and different scheduling algorithms

MPI/RT Philosophy

- Difficult for application developers to coordinate resource usage to guarantee application QoS
  - Difficult schedulability problem, user solutions not portable
- Users provide detailed application info. - platform either satisfies user requests (dedicates resources) or states that it cannot support requested quality of service
  - Allows MPI/RT implementors best to utilize platform data transfer primitives: two-sided, one-sided, and zero-sided communications
  - MPI/RT encourages this viewpoint but does not require it (that is, one can use priority model, or use “low QoS” specifications)
- Platform and Middleware Designers Exploit this information plus their inherently superior platform knowledge to achieve best performance portability

Semantics

- MPI/RT provides both
  - Early binding
    - Supports implementation guaranteed user QoS - Channels
  - Late binding
    - Provides API to help application developers to satisfy user QoS requirements
- MPI/RT provides an API for timeouts, handlers, instrumentation

Communication “Sidedness”

- MPI-1 supports 2-sided point-to-point.
- MPI/RT supports 2-sided, non-real-time modes of MPI-1.
- MPI/RT supports 1-sided point-to-point puts (and gets) on channels.
- MPI/RT supports 0-sided point-to-point transfers on time-based channels with periodic transfer.
- MPI-2 supports 1-sided window-oriented DSM. MPI/RT has not embraced this.

MPI/RT Paradigms

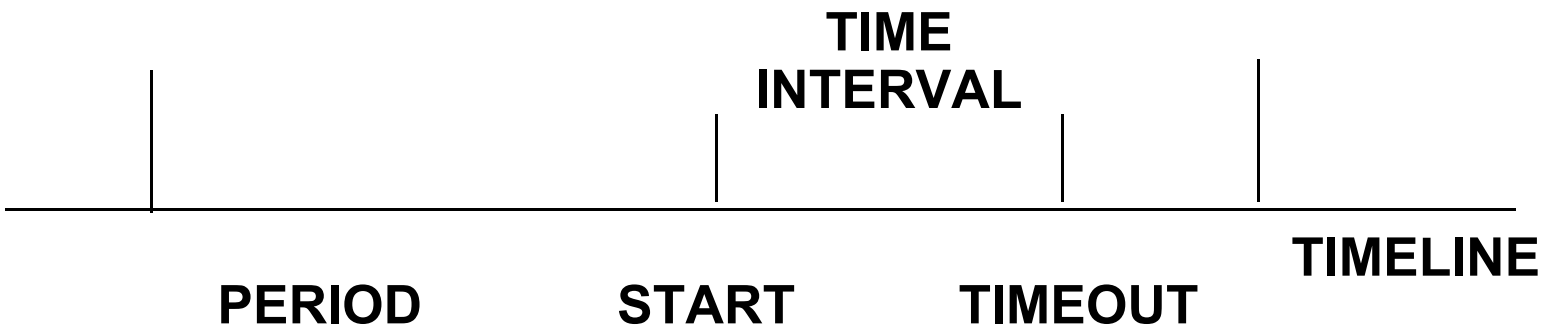
- Real-time paradigms - user perspective:
  - Time-Driven (purely calendar / periodic)
  - Mixed Time-Driven / Event-Driven (Time as special event)
  - Priority Channels with Event-Driven Delivery
  - Priority-Driven with polled delivery
  - High-Level Event-driven Model (Multicast events)
  - Other/Mixed Paradigms (Still being discussed/debated/designed)
  - Soft QOS (best effort)

MPI/RT Paradigms, II.

- Each paradigm has specific QoS characteristics
- Each paradigm has unique additional functionality
- Lower level QoS parameters that are resource specific are not part of the API: (bandwidth, jitter, latency)
- Presented as advice to implementors

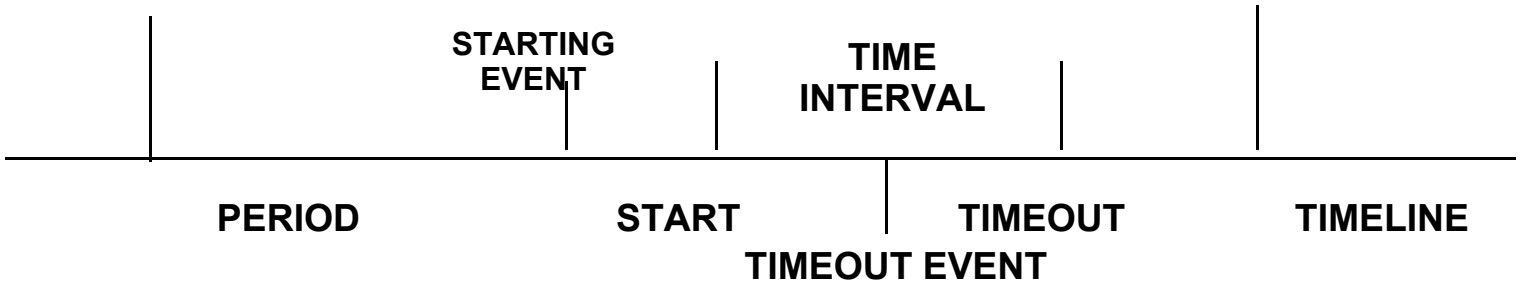
Time-Driven Real-Time Paradigm

- Application activities are satisfied within the requested time interval
- Quality of Service parameters:
  - Time interval for data transfer
  - Period of data transfer for periodic computations
  - Relationship between the time interval and the period

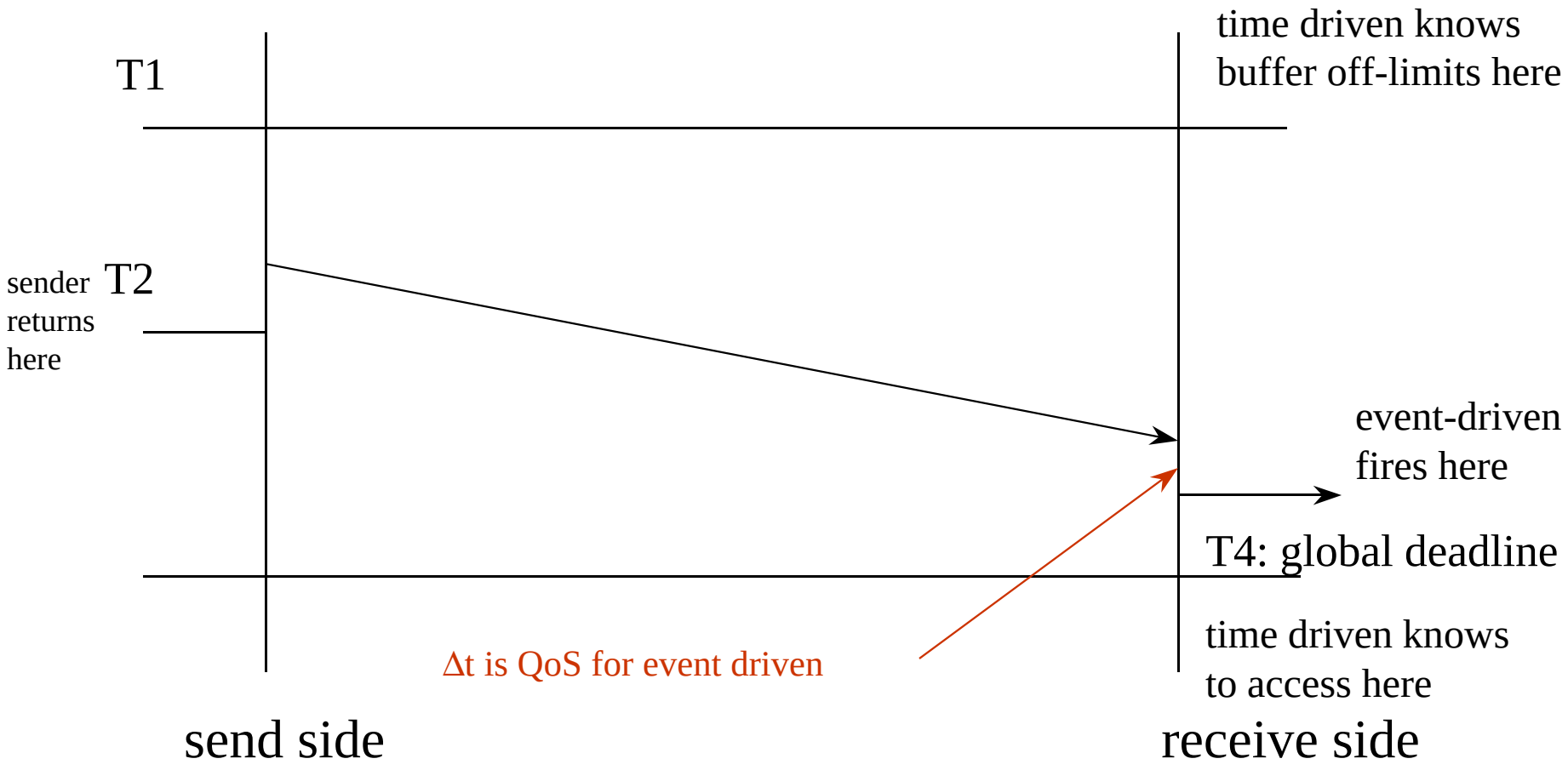


Event-Driven Real-Time Paradigms

- Application activities are bounded by time intervals, guarded by user specified events
- Quality of Service parameters:
  - Upper bound for an activation/deactivation time of an activity in response to an event:
- Events can be local (for low-level and high-level event driven paradigms) or group scoped (for high-level event-driven paradigm)



Comparing Time- and Event-Driven Transfers



Priority Models

- Channels may have a priority, not individual messages
- Early binding
- Optionally mix with event-driven paradigm (data transfers are event triggered)
- If there is no event-delivery QoS, this collapses to a priority-only model
- Channels with priority mean using multiple channels for multiple priority levels
- MPI/RT doesn’t give specific meaning to specific priority values
- Emphasizes static priority channels

Real-Time Point-to-Point Channels

- Communicators are used to describe groups and set up real-time communication
- Sets of channels, each with QoS are requested, and *collective admission tests* are performed by the implementation to define channels
- Point-to-point channels are uni-directional, reliable, packet passing mechanisms with QoS, buffering schemes, and fixed end-points

Buffers and Buffer Pools

- Buffer pools support abstraction that manages multiple buffers for an endpoint of a channel
- API specifies implementation buffer management strategy and application buffer management strategy
  - implementation: e.g., overwrite/non-overwrite of buffers when out of space
  - application: e.g., random access to buffers (not necessarily FIFO or LIFO)
- Concepts such as “double buffering” and “data fusion” are supported
- Buffer state transitions carefully specified to provide clear behavior
- Work on “cut-through”-like properties still under discussion

Admission Tests

- Admission tests over sets of channels is supported as a collective construction process. This is asserted to be a new way to present the real-time requirements to the message passing subsystem.
- A collective operation makes a set of point-to-point channels
- A collective operation defines a single collective channel, with a specific operation.
- Systems like Tenet and U. of Michigan group use admission tests on single channels, with hop-by-hop tests. MPI/RT specifies the admission test at a higher level.

Real-Time Collective Channels

- Collective channels use a fixed group of participants to define a collective operation as a single channel abstraction, offering the same type of quality of service as point-to-point channels
- For each collective operation, a collective channel with quality of service can be established. The MPI-type collective operation is executed on demand (or as scheduled) with the particular QOS.
- Collective channels are persistent and use “early binding” to negotiate once and use often a pattern of communication that is not binary, but rather involves the whole group of processes.

Collective Operation in RT

- Quality of service for clique and bi-partite collectives, in form of collective channels
- Early-binding semantics allows for use of poly-algorithms, and reduces per-use overheads
- Since existing MPI collectives (even MPI-2 extensions) have wrong abstraction level for transpose, we are developing specific transpose for RT, to be more easily used and optimized.

MPI/RT Clock

- Provides precise timing correctness
  - Synchronized clocks to support quality of service (QoS) timing requirements
  - Platform portability achieved - performance tuning using QoS parameters.
- Support fine grained instrumentation
  - delicate tuning for optimal performance
- Predictable timing behavior
- The MPI/RT clock is a synchronized clock.
  - The design of the synchronized clock is left to the implementor. MPI/RT Clocks may be local or global.
- Require that clocks be monotonically non-decreasing

MPI/RT Clock Parameter Definitions

- *Drift* - clock rate error bound
- *Skew* - maximum bound on absolute value of difference between simultaneous values of synchronized clock in distinct nodes
- *Accuracy* - maximum bound on absolute value of the difference between simultaneous values of synchronized clock and ideal clock, started at the synchronized clock's hypothetical start time.
- *Resolution (Tick)* - time between two successive updates
- *Access Time* - maximum bound on time to execute a call of MPI_WTIME, the synchronized clock access function.
- Formats of clock attributes are POSIX 1.b compliant.

Other MPI/RT Areas of Effort

- Monitoring/Instrumentation (Medium term)
  - provide portable linkage to system behavior
  - still being specified
- Fault Handling (Long term)
  - early explorations into making an MPI specification with fault tolerance or just fault detection
  - timeouts are first step in this direction
  - will leave most work for future
  - different audience with different requirements

Implementation Efforts @ MSU

- *Inertia* - provides no QoS, but provides complete API for demonstrating programs. Supports program development.
- *Action* - provides simple QoS on specific systems, with single channels. Utilization may be low for many channel programs. Mainstream target is Linux/RT. Research target is Maruti.
- *Momentum* - robust prototype [with MPI-1.2 underpinnings] that includes significant admission test logic. Will run on Linux/RT, and embedded systems.
- Full-blown implementations will be left for vendors, who will leverage the prototypes.

Implementations, II.

- MPICH code base is valid only for the *Inertia* Stage
- *Action* and *Momentum* prototypes require software infrastructure that is friendlier to threads throughout, plus which have a device architecture suitable for channels, without imposing copies
- Also need gang-scheduling and other device abstractions not in MPICH for advanced prototypes
- Maruti is time-based, and may have to have some specially coded prototype.
- The implementations are starting now, so that feedback into the standard process will be timely

Summary/Conclusions, I.

- MPI-1 was a success in SPMD technical computing
- MPI/RT builds on that success, while adopting certain ideas from MPI-2, and incorporating ideas not finally accepted to MPI-2
- MPI/RT is not tied to the potential “dark corners” of MPI-2, such as huge size, 1-sided chapter, ...
- Profiling rules allow for incremental support of MPI-2 within MPI/RT context.
- MPI/RT is now its own de facto standards activity

Summary/Conclusions, II.

- MPI/RT models of real-time:
  - time-driven
  - event-driven [merged with priority-, or with time-driven]
  - high-level event driven (multicast events)
- Key unifying concepts drive MPI/RT:
  - channels (point-to-point and collective)
  - buffer pools with clear semantics for buffer access and reuse
  - synchronized clocks
  - early binding, highly optimized API semantics
  - admission tests on multiple channels simultaneously

Main Collaborators on MPI/RT

- Robert Babb, University of Denver
- Ron Brightwell, Sandia National Labs.
- Dennis Cottel, NRaD
- Zhenqian Cui, MSU
- Arkady Kanevsky, MITRE Corp. (co-chair)
- Harish Nag, Intel
- Anna Rounbehler, Raytheon
- Anthony Skjellum, MSU (co-chair)
- Jerrell Watts, Caltech/Syracuse
- Tom Robey, Khoral Research

Online Access to MPI/RT

- Reflector:
  - subscribe to *mpi-realtime-request@mcs.anl.gov* (make your message have the line “subscribe”)
  - send comments to *mpi-realtime@mcs.anl.gov*
- Latest MPI-2 specification, under http://www.mcs.anl.gov/mpi
- MPI/RT Page: http://www.cs.msstate.edu/mpirt
  Includes papers, presentations, drafts, tutorials, ...
- Reaching MPI/RT co-chairs:
  A. Skjellum, 601-325-8435, *tony@cs.msstate.edu*
  A. Kanevsky, 508-271-5352, *arkady@mitre.org*