Design of a High-Performance Real-Time Message-Passing Interface (MPI/RT)

Anthony Skjellum
High Performance Computing Laboratory
NSF Engineering Research Center
Mississippi State University

Overview

- General History of MPI
- Technical Highlights of MPI-1, MPI-2, MPI/RT
- Scope of Specification and Functionality
- Philosophy
- Paradigms
- Clocks
- Summary/Conclusions
- Future Directions
- Online Access Information

General MPI History

- MPI is a de facto Message Passing Standard
- MPI-1 completed 5/94, revised 6/95 & 5/97
- MPI-2 involves extensions to MPI-1
- MPI/RT is a part of MPI-2 process but not part of standard to be released 5/97 - 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)

MPI/RT Forum Subcommittee

- Constituted informally in March/April, 1995
- Government, University, Commercial participants
- Has “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
- Meets at all MPI Forum meetings, and in between
- Like rest of MPI, an open forum

MPI-1 Technical Highlights, I.

- Reliable, large data point-to-point message passing and collective operations
- Group-oriented communication with safety/scope mechanisms
- Point-to-point operations (2-sided) scoped to ranks contained within groups contained in turn within communicators
- Non-blocking and blocking, synchronous and buffered point-to-point modes
- Blocking collective operations scoped to groups
- Dual-group (bi-partite) point-to-point
- Mapping of processes to graphs or Cartesian topologies

MPI-1 Technical Highlights, II.

- Environmental Inquiry
- Profiling Interface
- Supports SPMD/MPMD, but static process model
- Manageable (128 fns MPI-1.1, 129 fns MPI-1.2)
- Small number of concepts to master
- Language Independent Specification (LIS)
- Language Specific Bindings (LSB) for C, Fortran77
- Requirement for a thread-safe API

MPI-2 Technical Highlights

- Large superset of MPI-1.2
- Dynamic Process Management (Spawn, Client-Server, …)
- 1-sided communication (high-level put/get model to “windows”)
- Collective I/O for flat file structures
- Collective operations on bi-partite (2 group) structures
- External interfaces for 3rd party tools
- LSB for C, Fortran77, C++, and Fortran90

MPI Collective Operations

- Collective operations supported by MPI:
  - clique versions (MPI-1)
  - bi-partite versions (MPI-2) [local + remote group]
  - scoped to user-specified groups, not whole allocation
- Operations include:
  - MPI: gather, scatter, reduce, allreduce, alltoall, reducescatter, scan
  - MPI-2: alltoall generalization as well
  - MPI/RT: transpose-and-reshape to be defined

MPI/RT Technical Highlights

- Embraces MPI as base API and programming strategy, but modifies it for real-time
- Higher optimizability than MPI-1 and MPI-2
- Designed to support time-driven , priority-driven and event-driven 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

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 experts
  - Platform provides message passing API
- Step 2 - Application developers are responsible for achieving desired application performance and real-time guarantees
  - It is up to application developers to get desired QoS and performance using platform-specific API
- Inherently platform-dependent, non-portable

MPI/RT Philosophy - Rationale

- Designers and implementors of a platform are better platform experts than users
  - Operating systems
  - Communication layers
  - Middleware with MPI/RT
- Platform and middleware designers know better what it takes to provide QoS

Rationale - QoS Guarantees

- Providing QoS guarantee is a difficult schedulability problem:
  - Involves scheduling different resources:
    - CPUs where send/receiver applications are running
    - Data buses
    - DMA engines
    - Memory
    - Network interface chips (NICs)
    - Physical network (topology dependencies)
  - Different resources are scheduled using real-time paradigms and different scheduling algorithms

Rationale - MPI/RT Implementors' Viewpoint

- Platform designers are in the best position to know all the resources involved in satisfying data transfer QoS guarantees and resource's interdependencies
- Platform designers can adjust run-time system parameters during initialization
- Different solutions for satisfying QoS guarantees for different architectures (hidden from the users)

MPI/RT Philosophy - Cornerstone

- Difficult for application developers to coordinate resource usage to guarantee application QoS
  - User solutions are 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 to best utilize platform data transfer primitives: two-sided, one-sided, and zero-sided communications
  - MPI/RT encourages this viewpoint but does not require it

MPI/RT Philosophy - 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

MPI/RT Paradigms

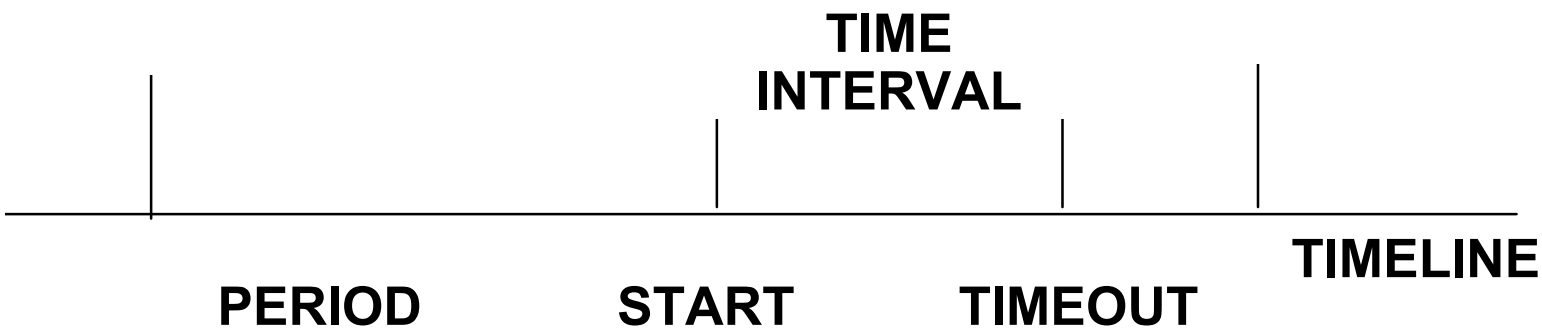
- Real-time paradigms - user perspective:
  - Time-Driven
  - Event Driven
  - Priority-Driven
  - Mixed Paradigms (Still being discussed/debated/designed)
- Each paradigm has specific QoS characteristics
- Each paradigm has unique additional functionality
- Common underlying principles

  `channels` - for early binding
- Lower level QoS parameters that are resource specific are not part of the API: (bandwidth, jitter, latency)

  Presented as advice to implementors

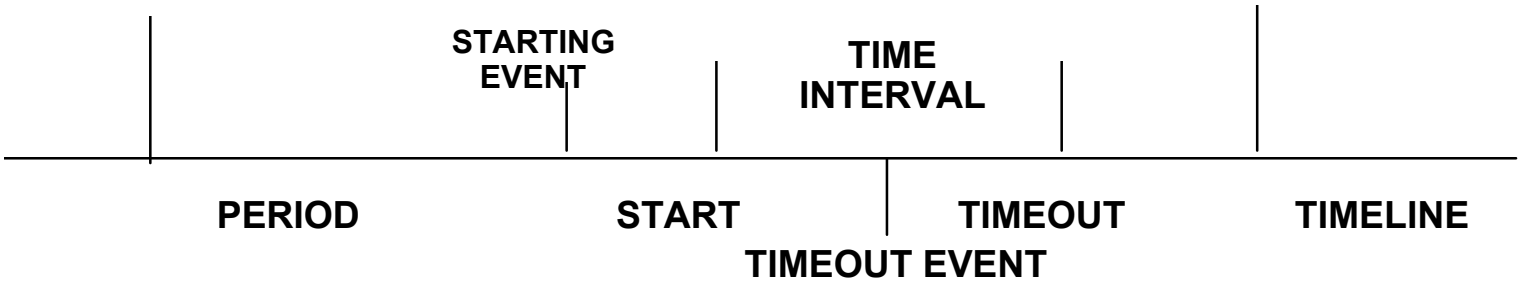
Time-Driven Real-Time Paradigm

- 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 Paradigm

- 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:
- Two models: low level and high level event-driven paradigms
- Events can be local (for low-level and high-level event driven paradigms) or group scoped (for high-level event-driven paradigm)



Priority-Driven Real-Time Paradigm

- User specified priority for data transfer
  - Early binding - Priority of a persistent channel
- Quality of Service parameters:
  - Integer priority of the persistent channel for message transmission
- Priority is per channel, not per message
- Use multiple channels to achieve multiple priorities

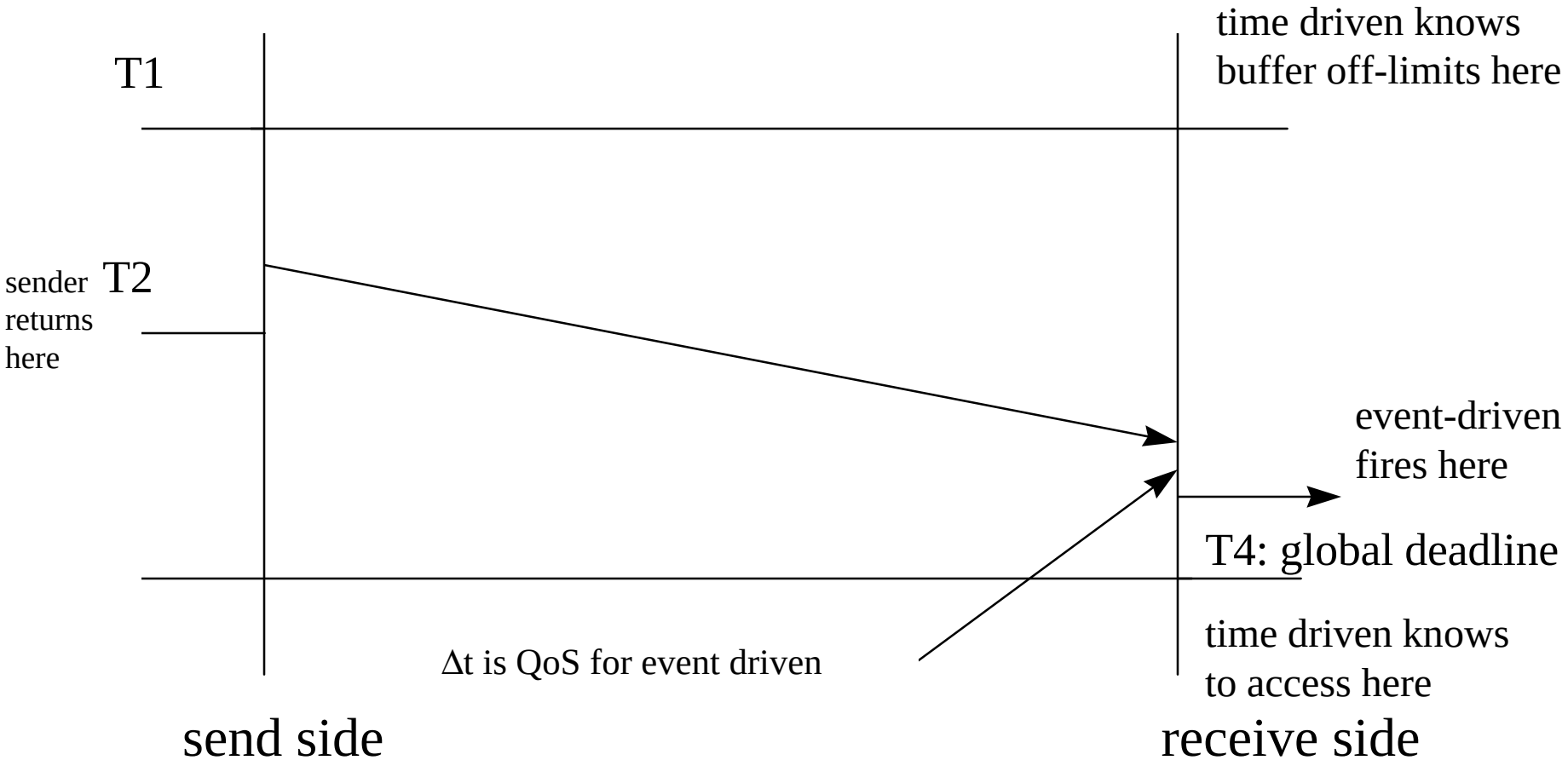
Real-Time Point-to-Point Channels

- 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

Communication “Sidedness”

- MPI-1 supports 2-sided point-to-point.
- MPI/RT supports 2-sided, non-realtime 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.

Comparing Time- and Event-Driven Transfers



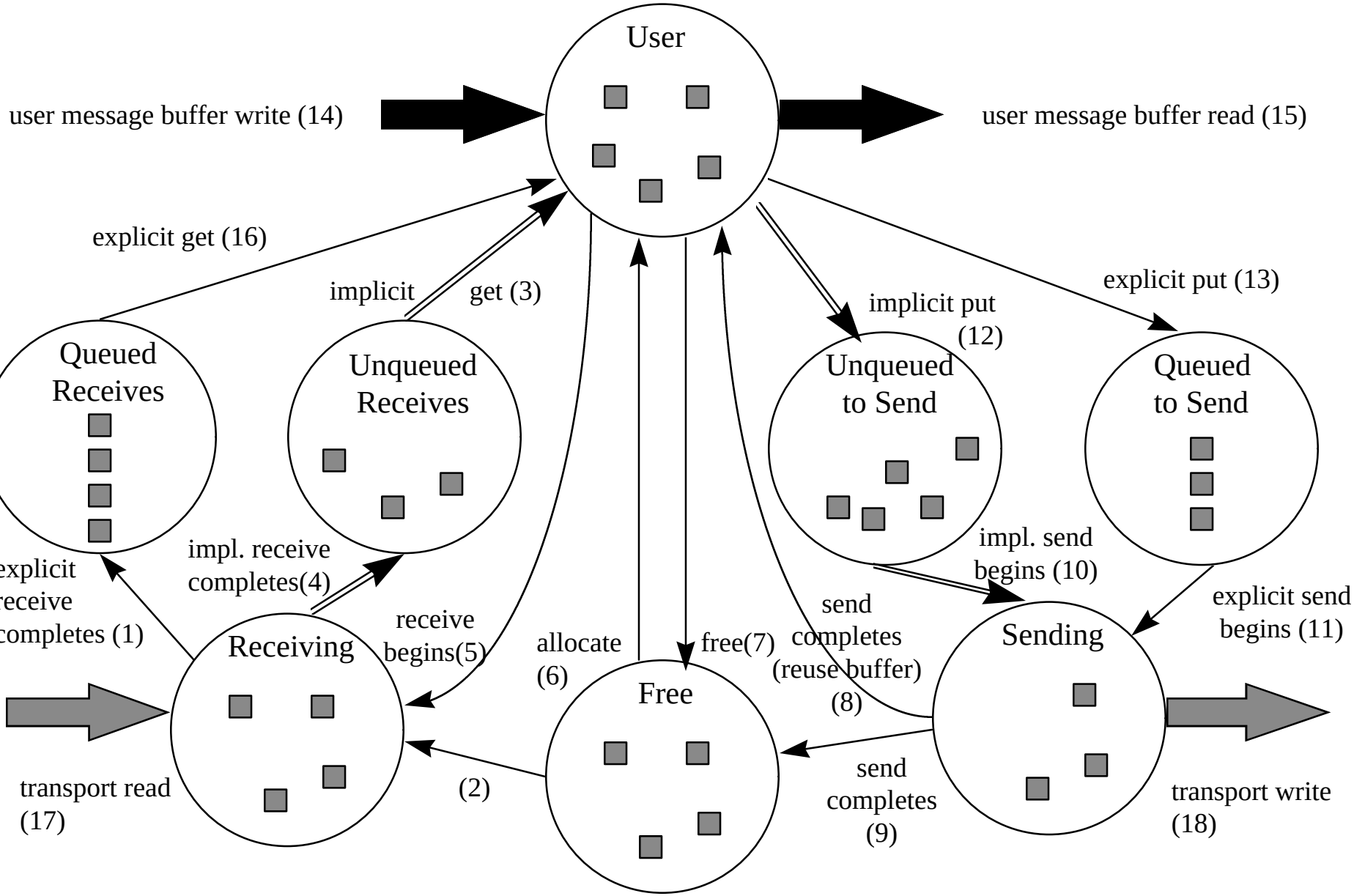
Real-Time Collective Channels

- 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.

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

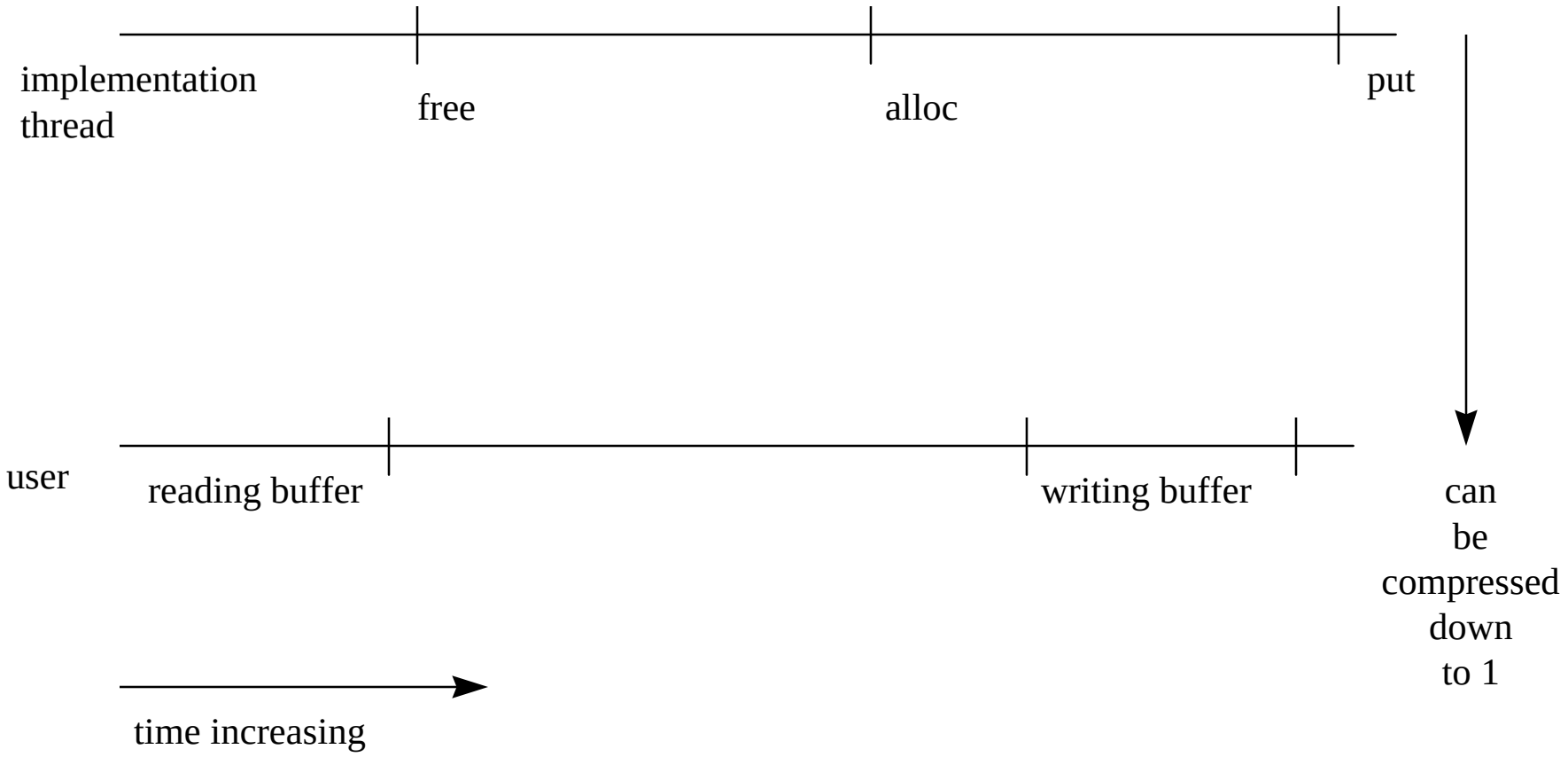
The Bufferpool/Buffer State Transition Diagram for MPI/RT



State Transition Legend

(1) Explicit receive completes
(2) Allocate Buffer from Buffer Pool Free List for Receive operation
(3) Implicit get of buffer asynchronously from user activity
(4) Implicit receive completes
(5) Receive begins (buffered receive)
(6) Allocate Buffer from Buffer Pool Free List
(7) Free Buffer to in Buffer Pool
(8) Send completes (with buffer returned to user intact for reuse)
(9) Send completes (buffer returned to Buffer Pool Free List)
(10) Implicit send begins
(11) Explicit send begins
(12) Implicit put of Buffer
(13) Explicit put
(14) User message buffer write (user gathers in a buffer to program space)
(15) User message buffer read (user scatters out a buffer to program space)
(16) Explicit get of buffer by user
(17) Transport Read of Message into Buffer
(18) Transport Write of Message from Buffer

Two-Threaded Logical Implementation Model



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.

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 - for each synchronized clock a guaranteed upper bound on the error in the rate of clock. Dimensionless.
- 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.
- 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.

MPI/RT Clocks are the Backbone of QoS

- QoS parameters use clock attributes to accurately represent:
  - activation time for event handler (event-driven paradigm)
  - time intervals, timeouts, periods, start time (time driven paradigm)

Other MPI/RT Areas of Effort

- Monitoring/Instrumentation (Medium term)
  - provide portable linkage to system behavior
  - still being specified
- Fault Tolerance (Longer 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

Summary/Conclusions, I.

- MPI-1 was a success in SPMD technical computing
- MPI/RT builds on that success, while adopting 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
- Profiling rules allow for incremental support of MPI-2 within MPI/RT context.

Summary/Conclusions, II.

- MPI/RT supports three models of real-time as of now:
  - time-driven
  - event-driven
  - priority-driven
- Key unifying concepts drive MPI/RT:
  - channels
  - buffer pools
  - synchronized clocks
  - early binding

Some Future Directions

- Look at MPI/RT techniques as input to other activities:
  - Real-time IETF working groups (e.g., Packetway)
  - Real-time CORBA RFI
- Widen application base:
  - Traditional scientific/engineering users with no (current) explicit real-time requirements
- Study system utilization optimization

Main Collaborators on MPI/RT

- Robert Babb, University of Denver
- Ron Brightwell, Sandia National Labs.
- Zhenqian Cui, MSU
- Arkady Kanevsky, MITRE Corp. (co-chair)
- Jin Li, MSU
- Harish Nag, Intel
- Anna Rounbehler, SKY Computers, Inc.
- Anthony Skjellum, MSU (chair)
- Jerrell Watts, Caltech/Syracuse

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/RT: ftp://aurora.cs.msstate.edu/pub/mpi/rt-2-26feb97.ps
  (many older drafts are kept there too for completeness)
  Latest MPI-2 specification, under http://www.mcs.anl.gov/mpi
- Reaching MPI/RT draft core group

  A. Skjellum, 601-325-8435, *tony@cs.msstate.edu*
  A. Kanevsky, 508-271-5352, *arkady@mitre.org*
  A. Rounbehler, 508-250-1920, *anna@sky.com*
- New MPI/RT Page (this lecture will appear there as PS/PDF)
  http://www.cs.msstate.edu/mpirt [to be launched soon, 3/31/97]