Interoperability Support in Distributed On-line Monitoring Systems

Roland Wismüller

Lehrstuhl für Rechnertechnik und Rechnerorganisation, LRR-TUM
Technische Universität München, Germany
e-mail: wismuell@in.tum.de
www: www.in.tum.de/∼wismuell

1. Introduction

Run-Time Tools (On-line Tools)

- *Monitor*, *analyze* and *control* the execution of a distributed target system (HW+SW)
- E.g. debuggers, performance analyzers, load balancers, ...
- Common properties:
  - event based scheme of operation
  - distributed structure:
    - distributed monitor/control components (*monitoring system*)
    - (usually) centralized analysis component
    - centralized user interface (for interactive tools)

1. Introduction (ctd.)

Interoperability

- Dazzling term with lots of different definitions:
  - polylingual software (e.g. C and Fortran)
  - communication of distributed programs (e.g. CORBA)
  - data bases (relational vs. OO)
  - ...
- An attempt of a more general definition:

  Interoperability is the ability of independent software components (not specifically designed for that purpose) to cooperate at a syntactical and semantical level.

1. Introduction (ctd.)

Interoperability of On-line Tools

- Multiple tools cooperate in the monitoring and control of the same target system

Motivation:

- For the user:
  - concurrent use of tools for different tasks
  - combined use can lead to additional benefits
- For the tool developer:
  - enhanced modularity
  - “factor out” target system dependencies

1. Introduction (ctd.)

Possible Scenarios

- Debugger + visualizer / message queue manager
- Debugger + tool for deterministic (re-)execution
- Debugger + checkpointing system
- Multiple performance analyzers
- Performance analyzer + resource management

Requirements

- Tools *must* be usable at the same time
- Tools *should* cooperate

2. Structural Conflicts

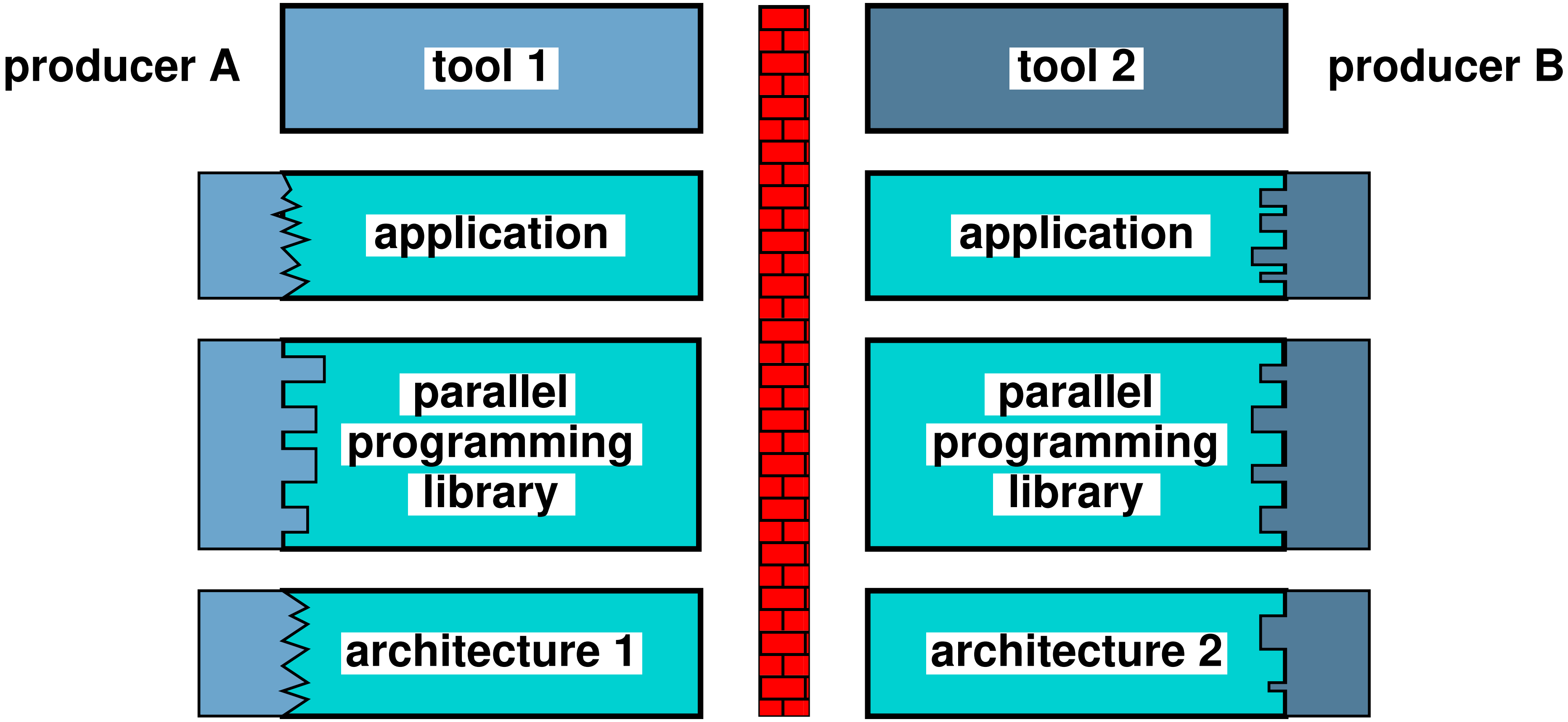
The Reality

- Independent tools usually cannot be used at the same time

Why?

- They require different (incompatible) preprocessing
- They require different pre-instrumented libraries
- They use exclusive interfaces (like `ptrace` or `/proc`)

Current Tool Design



2. Structural Conflicts (ctd.)

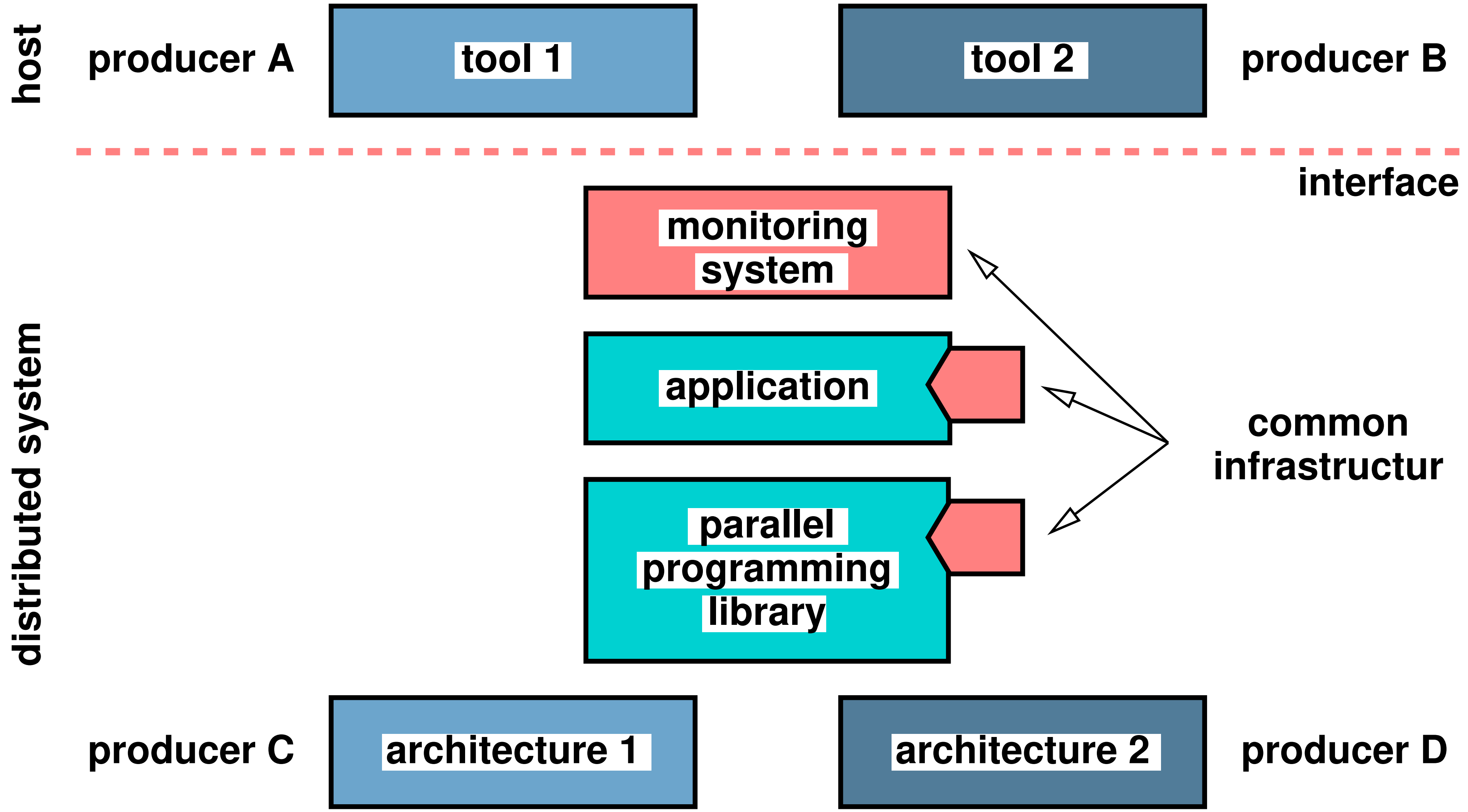
A Solution

- Use a common on-line monitoring interface

Approaches

- P2D2 (NASA Ames): specific to debugging
- UMA (X/Open): specific to performance measurement
- DAMS (Universidade Nova de Lisboa)
- DPCL (IBM, Univ. of Wisconsin, Madison)
- OMIS (Technische Universität München)

Tool Design Using a Common Monitoring Interface



3. Logical Conflicts

Situation

- Several tools access (i.e. read/write) the state of the target objects

Three types of problems

1. Consistency problems
2. Transparency problems
3. Write-write conflicts

3.1 Consistency Problems

Example

- Visualizer displays value of a matrix
- Debugger modifies a matrix element
- Visualizer should update its display

Support by monitoring system

- Monitoring system notifies visualizer
- Related to memory consistency in DSM systems

3.2 Transparency Problems

Example

- Sampling tool defines an alarm timer in a target process
- Debugger intercepts all signals
- Debugger should not see the alarm signal

Support by monitoring system

- Allow sampling tool to intercept event notification
- In general: intercept service requests and modify results

3.3 Write-Write Conflicts

General case

- Two tools modify the same part of a target object's state
- No specific support possible

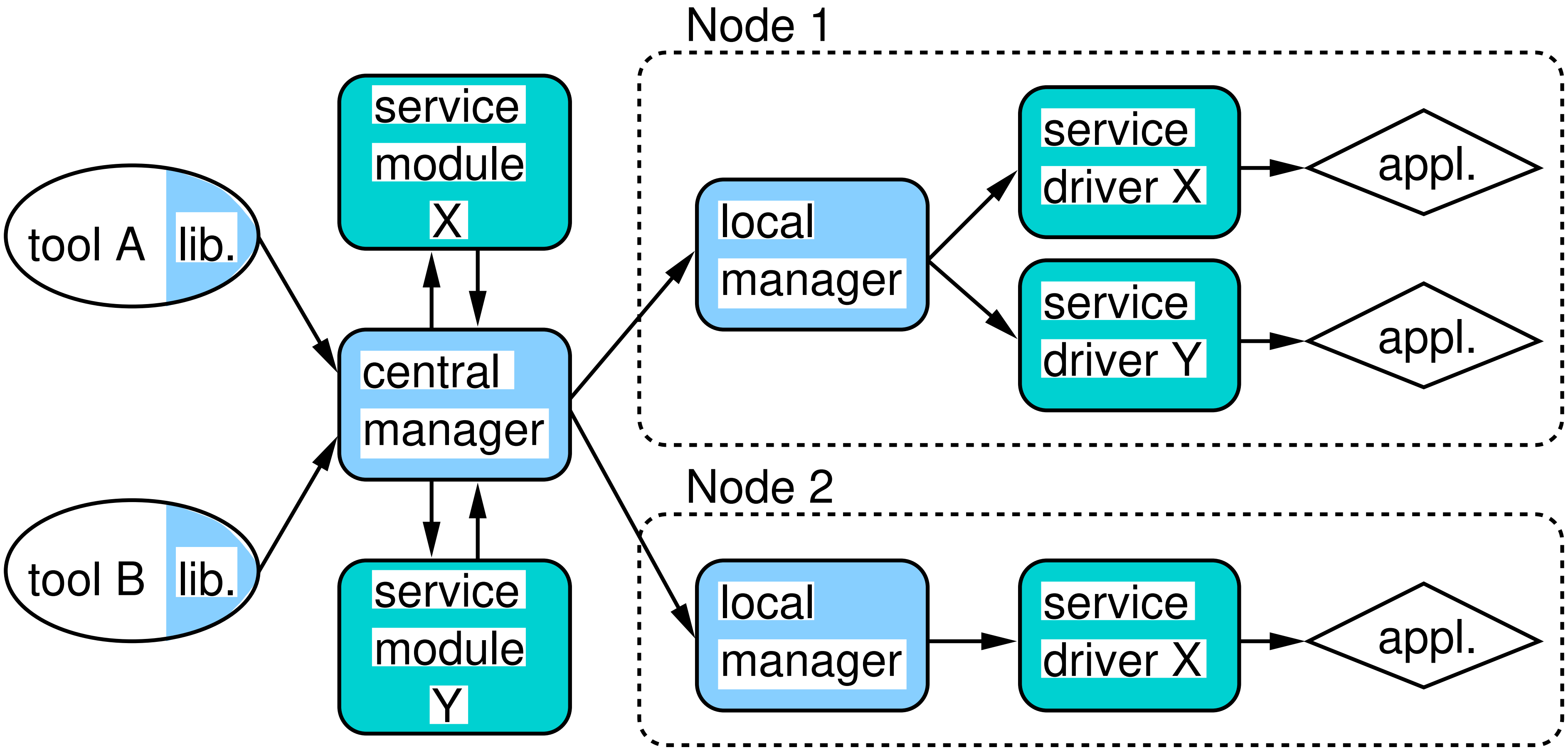
Special case

- Two tools instrument the same event / program location
- Example: visualizer and replay tool instrument end of receive function
  - visualizer displays information on received message
  - replay tool changes the message received
- Problem: determine the correct order of instrumentations

DAMS (Distributed Applications Monitoring System

- Infrastructure for distributed monitoring systems
- Collection of distributed daemons
- Implementation of monitoring services as extensions
- PDBG: debugging service for DAMS, based on gdb
- No structural conflicts: only one driver per target process
- Support for consistency problems: notification services
  - change of process state (running, stopped, terminated ...)
  - end of a monitoring action, including result
- Problem: interoperability only within one extension

DAMS Architecture

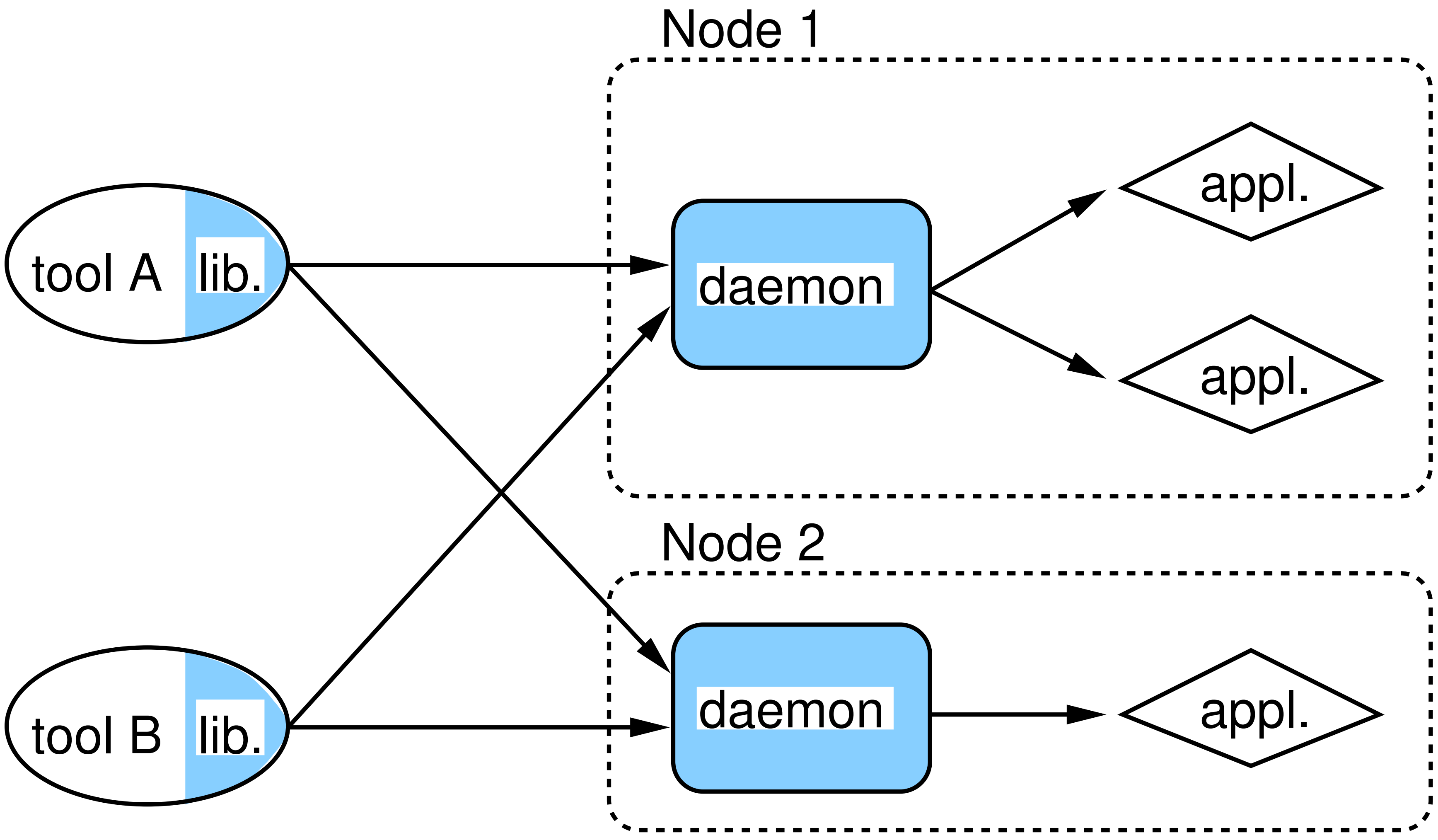


4 Support in Monitoring Systems (ctd.)

DPCL (Dynamic Probe Class Library)

- Infrastructure for dynamic instrumentation of distributed systems
  - inserts (nearly) arbitrary code fragments (*probes*) into the code of a running process
  - probes can return results to tools
- Implemented as a C++-library and a network of daemons
- Daemons are capable to connect to multiple tools
- No structural conflicts: only one daemon instruments a target process
- Currently no support to solve logical conflicts

DPCL Architecture



4 Support in Monitoring Systems (ctd.)

OMIS (On-line Monitoring Interface Specification)

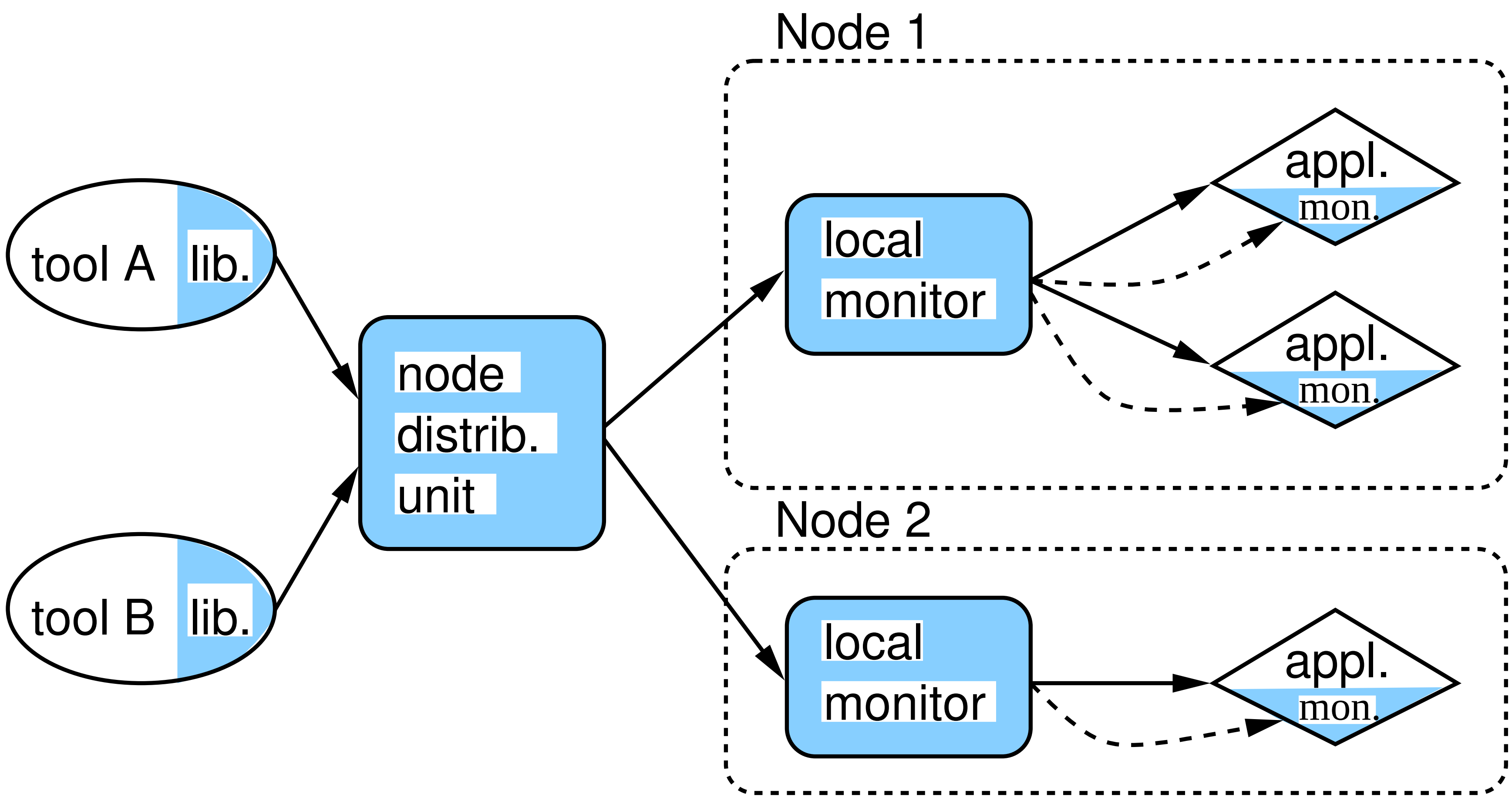
- Specification of a monitoring interface
  - basis: event/action paradigm
- OCM: Implementation of OMIS for PVM
  - distributed server infrastructure
  - services implemented in extensions
  - infrastructure has some (explicit) knowledge on services
  - „pre-instrumented“ libraries

4 Support in Monitoring Systems (ctd.)

OMIS (ctd.)

- No structural conflicts:
  - only one server per target process
  - instrumentation of all extensions can be inserted in „pre-instrumented“ libraries
- Support for consistency problems: notification services
  - certain manipulation services have associated events
  - problems:
    - incompleteness
    - ambiguity: different services may modify the same part of a target object’s state

OCM Architecture



5. Conclusions and Future Work

Conclusions

- Interoperability of run-time tools is a promising way of implementing advanced tool environments
- Interoperable tools need a supporting monitoring system
- Several projects: DAMS, DPCL, OMIS, FIRST (CSCS), ...
- Support for logical conflicts still imposes problems

Future Work

- Test / implement ideas to solve logical conflicts in the OCM
- Monitoring system support for direct tool interactions
- Implement a set of interoperable tools based on OCM