RUSS COLLEGE of Engineering and Technology

RMBench:
CORBA Benchmarking Services for Resource Management Middleware

Matthew Delaney { matthew.delaney.1@ohio.edu }

Lonnie Welch, David Juedes and Chang Liu
Center for Intelligent, Distributed & Dependable Systems
Ohio University
Athens, OH

OMG RTES 2004
Reston, VA
July 12-15, 2004

OHIO UNIVERSITY

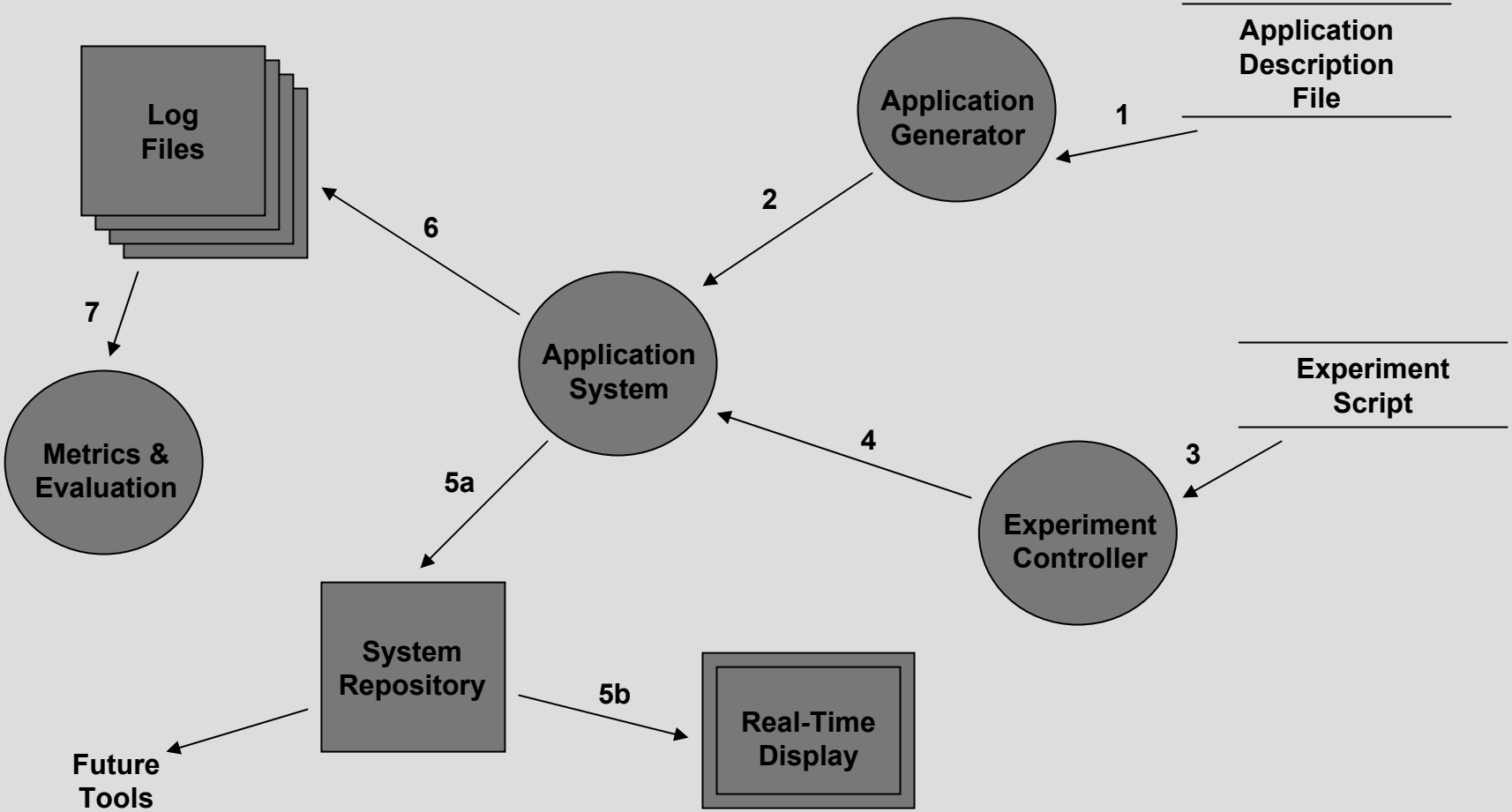
Evaluation of RM Middleware

- Previous benchmark approaches for RM middleware:
  - Hartstone Distributed Benchmark
  - Dynbench
  - SWSL/SWG
- Can current benchmarks emulate dynamic real-time QoS-based systems?

Overview

- RMBench emulates dynamic RT QoS-based systems for evaluation of RM middleware
- Features include:
  - Workload as a function of input (dynamic workload functions)
  - QoS levels controlled by a CORBA interface
  - Specification-driven application generation
  - Specification-driven experiment control
  - Logging, offline analysis and metrics

RMBench Architecture

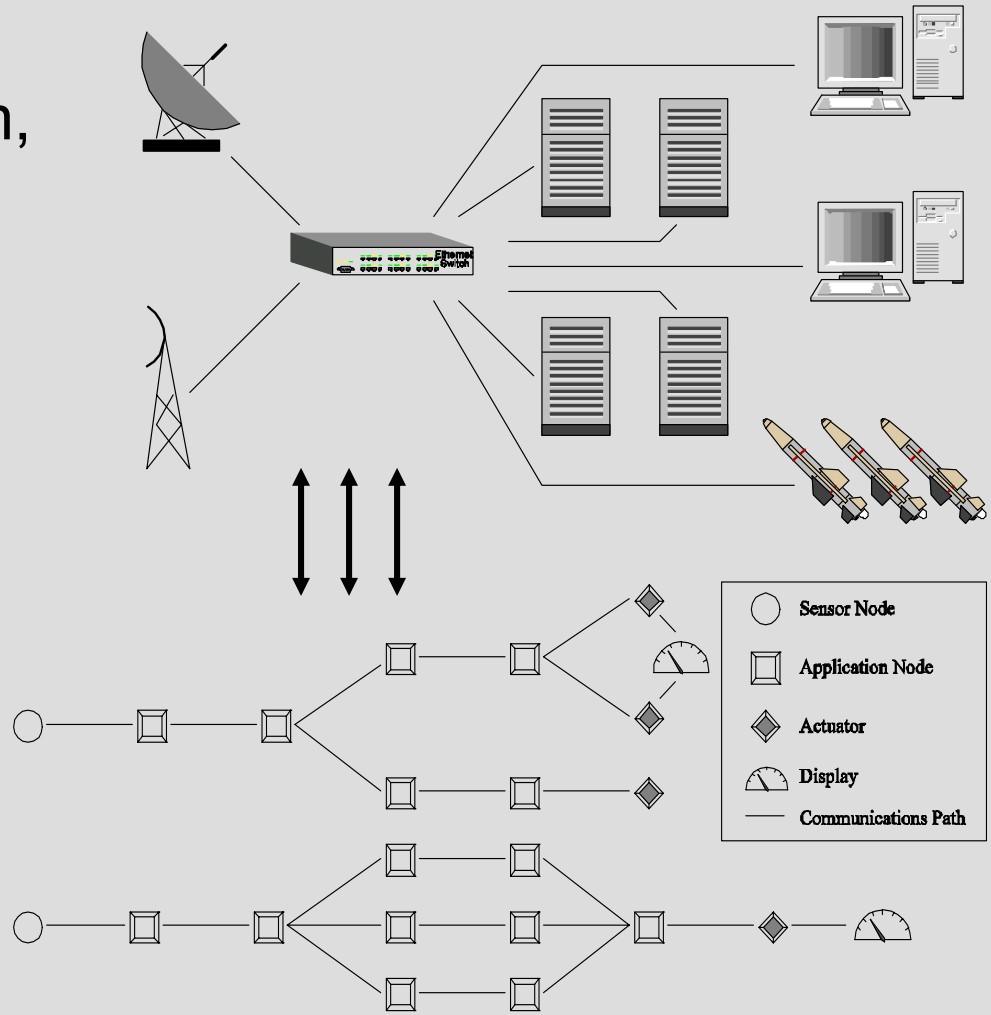


RMBench Application Description

- Applications are emulated by using a combination of Sensor and Path nodes
- Sensor node – *data producer*
  - Periodically create and distribute dynamic workloads
  - Simulate scenarios where workloads in the system change during runtime
  - CORBA interface is provided to change period and message size of sensor
- Path node – *data consumer*
  - Receives messages → performs work (CPU workload)
  - Passes on message

RMBench Application Description

Given a live software system, RMBench uses sensor/path nodes to generate the application for use in RM middleware evaluation



Workload Functions

- Often it is not meaningful (or is impossible) to characterize task CPU usage with worst case execution times
- Characterize as a function of input
- RMBench workload function input is incoming message size (in bytes)
- RMBench path nodes contain two workload functions:
  - CPU workload → square root calculation
  - Network workload → outgoing message size

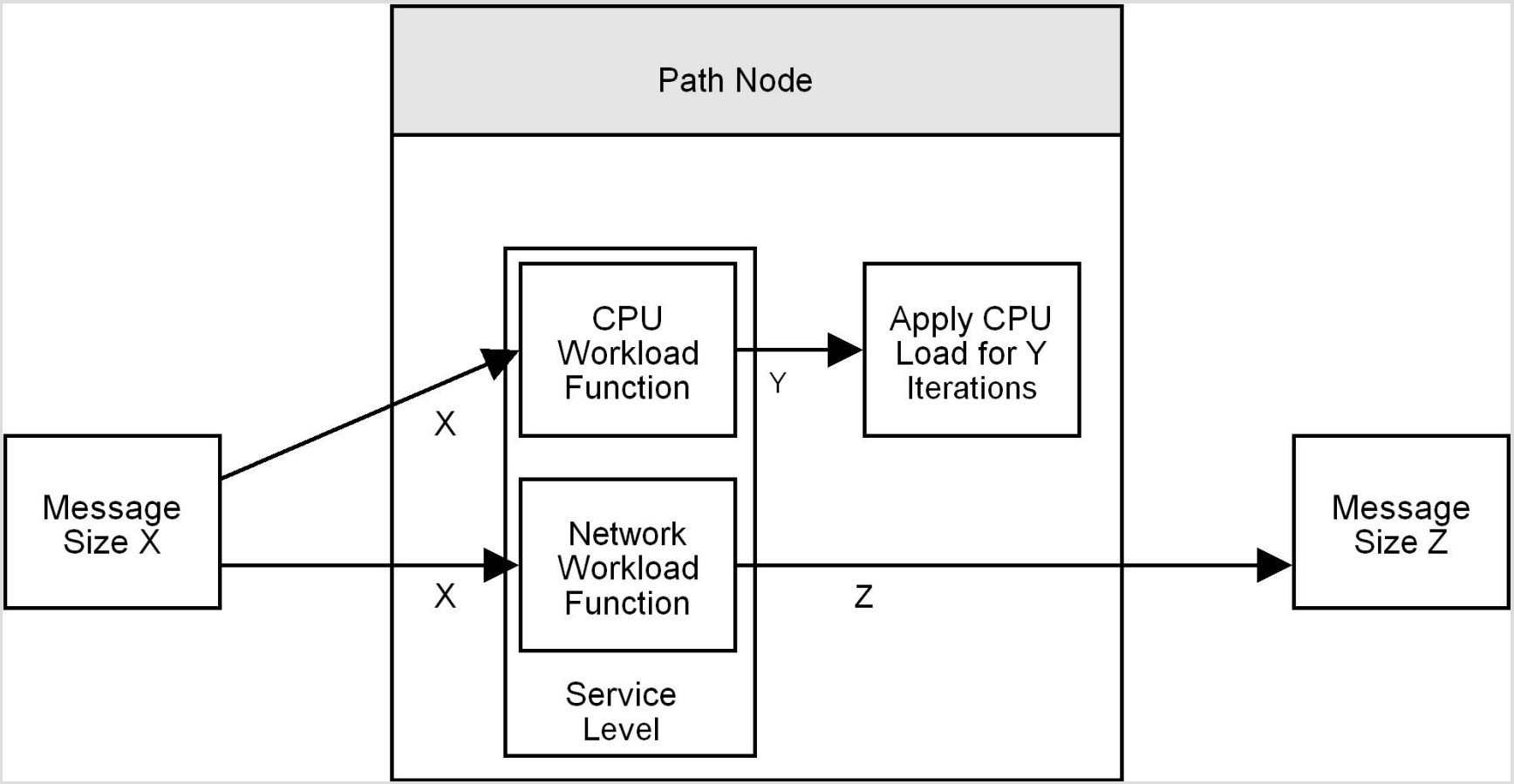
QoS Service Levels

- Workloads are combined into *service levels:*

```
SERVICE_LEVEL_LOW {
    CPU_WORKLOAD = 2 n
    NETWORK_WORKLOAD = 5 log n
}
```

- Only one service level is active at a given time (per path node)
- A CORBA interface is provided for controlling service levels

QoS Service Levels



Application Generation

- Users create application specification files
- Application generator process creates application topology from spec file
- Useful for rerunning / reusing experiments or distributed development

```
PathNode DISTRIBUTOR_1 {
    Host water light fire;
    ServiceLevel low {
      CPU_WorkloadFunction 2 n + 5 logn;
      Network_WorkloadFunction 2 n;
    }
    StartupLevel low;
    StartupHost light;
```

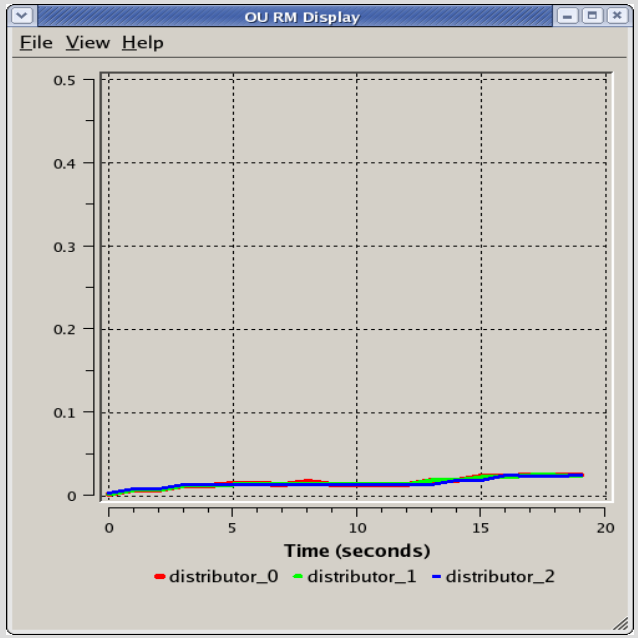
Experiment Control

- Users create experiment scripts for creating runtime changes in environment
- Sensors nodes can change period or workload
- Path nodes can change service levels

```
S  Sensor_A  S   1  200  500

S  Sensor_B  M   2  150  400

P  Path_A    S   4  LOW

P  Path_B    M   8  Sensor_B  HIGH
```

RMBench Support Processes

- Real-time performance display
  - Runtime latency information for path nodes
  - Gets data from system repository



RMBench Support Processes

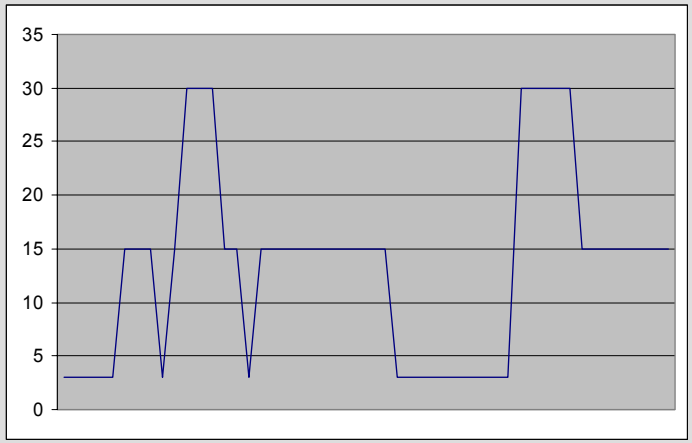
- System repository
  - Runtime data is stored and updated
  - Keeps history
  - Simple communication integration with outside components or future tools

Offline Analysis

- RMBench creates log files for offline analysis
- Files contain message number, message size, time received, network and CPU latency, and active service level
- Analysis tools are provided to determine metrics such as:
  - End to end latency
  - Stability
  - QoS violations
  - QoS violation rate
  - Utility

Experimental Results

- RMBench was used to detect RMS instability
- Experiments showed erratic service level changes with no change in environment
- Service levels were simulating MPEG frame rates of 3, 15, and 30



Future Work

- Event-driven tasks (message generation)
- Stochastic workloads
- QARMA evaluation

Conclusions

- RMBench extends current benchmarking methodologies by enabling dynamic workload functions in application modeling
- RM middleware can control QoS service levels within path nodes
- The experiment controller creates a repeatable and dynamic environment for middleware evaluation
- RMBench has been used to show instability in a developmental RMS

2004 Publications

- D. Fleeman et al. "Quality-based Adaptive Resource Management Architecture (QARMA): A CORBA Resource Management Service," *The 12th IPDPS Workshop on Parallel and Distributed Real-Time Systems* (WPDRTS '04), Santa Fe, New Mexico, USA, April 2004.

- D. Juedes et al. "Heuristic Resource Allocation Algorithms for Maximizing Allowable Workload in Dynamic, Distributed, Real-Time Systems," *The 12th IPDPS Workshop on Parallel and Distributed Real-Time Systems* (WPDRTS '04), Santa Fe, New Mexico, USA, April 2004.

- F. Drews et al. "Utility-Function based Resource Allocation for Adaptable Applications in Dynamic, Distributed, Real-Time Systems," *The 12th IPDPS Workshop on Parallel and Distributed Real-Time Systems* (WPDRTS '04), Santa Fe, New Mexico, USA, April 2004.

- E. Aber et al. "Experimental Comparison of Heuristic and Optimal Resource Allocation Algorithms for Maximizing Allowable Workload in Dynamic, Distributed Real-Time Systems," *The 6th Brazilian Workshop on Real-Time Systems* (WTR '04), Brazil, May 2004.

- C. Liu et al. "Model-Driven Resource Management for Distributed Real-Time and Embedded Systems," *The 2nd RTAS Workshop on Model-Driven Embedded Systems* (MoDES '04), 2004.

- F. Drews et al. “Workload Functions: A New Pardigm for Real-time Computing,” *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS’04) Work In-Progress Session*, Le Royal Meridien, King Edward, Toronto, Canada, May 25-28, 2004.