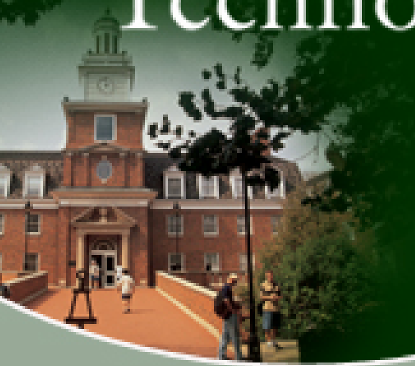
RUSS COLLEGE of Engineering and Technology



Adaptive Resource Management Services in CORBA:

Quality-based Adaptive Resource Management Architecture (QARMA)

David Fleeman { fleeman@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

Presentation Overview

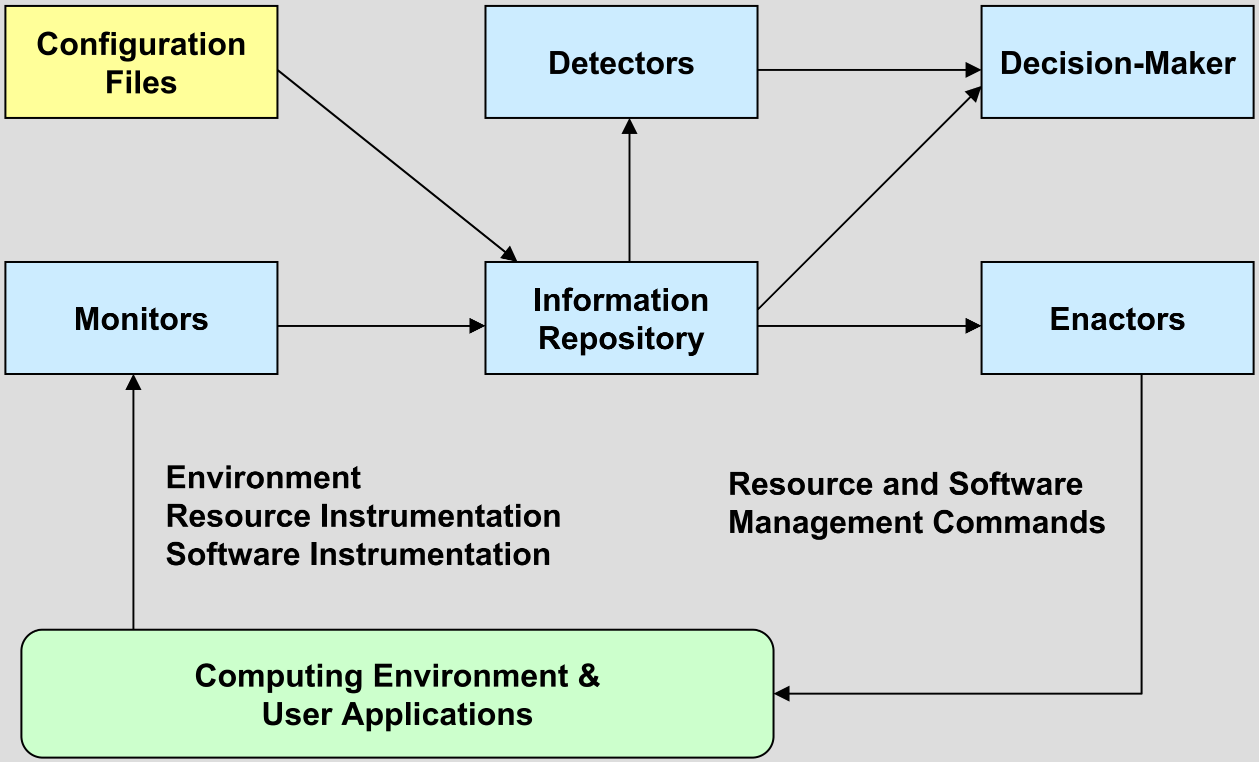
- QARMA
  - Overview of Architecture
  - System Repository Service
  - Resource Management Service
  - Enactor Service
  - Experimental Results

- Integration Efforts
  - RT CORBA Dynamic Scheduling Service (URI)
  - Utah CPU Broker (Utah)

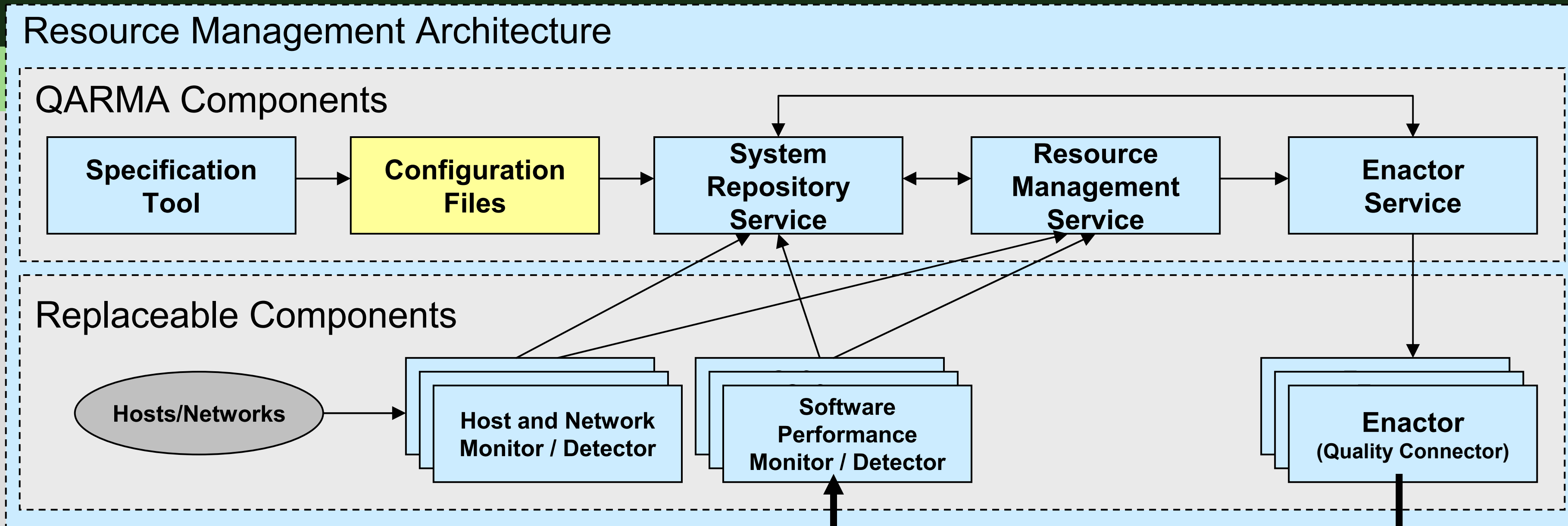
- Technology Transition
  - DARPA PCES BBN OEP (UAV)
  - DARPA ARMS MLRM component

- Conclusions and Future Work

Generic Resource Management Architecture

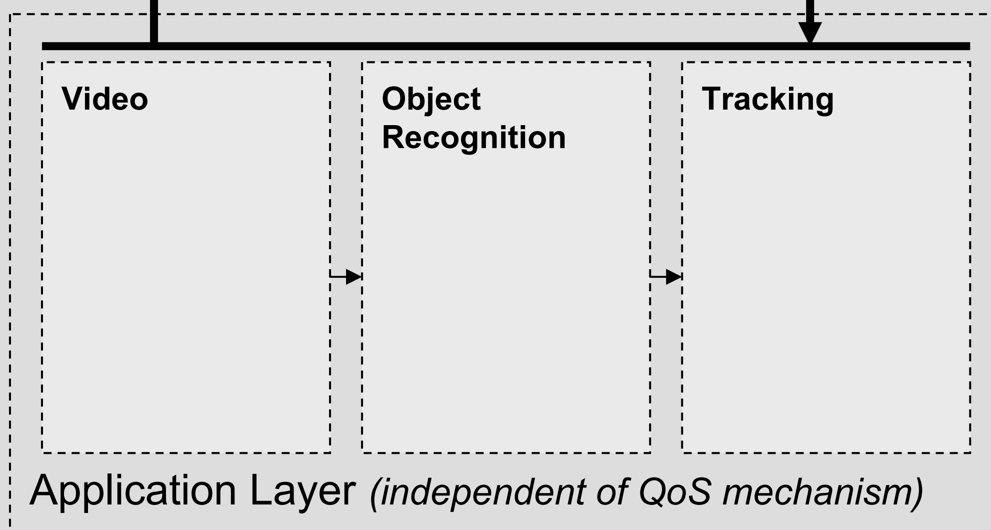


CORBA Resource Management Architecture



Resource Management Service Goal

- Each chain of applications achieves a "benefit" based on it's service attribute settings (e.g., frame rate) and extrinsic attribute settings (e.g., importance).
- The Resource Management Service changes service attributes in order to optimize total benefit, subject to the constraint that all tasks meet their real-time requirements.



QARMA System Repository Service

- Purpose
  - Central repository for communicating all specification and state data to management components
- Benefits
  - Reduced complexity of other components
  - Other components become stateless
  - Need for replication and fault tolerance of other components is decreased
  - Flexibility in replacing storage mechanism behind System Repository Service (e.g., memory, XML files, database)
  - Integration with components developed by other organizations hinges only on agreement of the data stored in the System Repository
- Drawbacks
  - Becomes a single point of failure
  - Not scalable for arbitrarily large systems
  - Some of these drawbacks can be handled with standard fault-tolerance and replication mechanisms.

QARMA Resource Management Service

- Purpose
  - Make intelligent allocation decisions.
- Benefits
  - Ability to perform global feasibility analysis for tasks that require multiple resources.
  - Ability to perform global optimization for service quality settings.
- Drawbacks
  - Not scalable for arbitrarily large systems.

QARMA Resource Management Service
The Key Problems

- Information Acquisition
- Information Modeling
- Resource Allocation Algorithms
  - Feasibility Testing
  - Optimization Objective
  - Control Mechanisms
  - Pluggable

Information Acquisition

- Obtained from System Repository Service
  - *Resource Specification:*
    - Describes hosts, network devices, and network connectivity.
  - *Software Specification:*
    - Describes the extrinsic attributes that affect performance of applications and the service attributes that can be used to control application resource usage and quality.
    - Describes the application connectivity and dependencies, performance requirements, and performance characteristics.
  - *Monitoring Data:*
    - Host and network resource usage state and statistics
    - Software performance state and statistics

Information Modeling

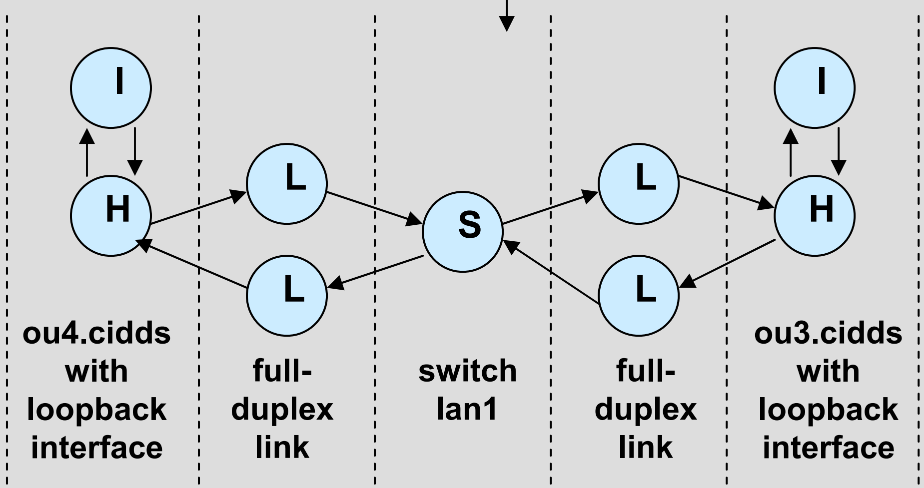
- Specification data is transformed into mathematical models.
  - Based on the models presented in Jane Liu's "Real-Time Systems" book.
  - Service and extrinsic attributes have been added to extend the model presented in the book.

- Four models are created:
  - *Resource Model*
  - *Task Model*
  - *Profile Model*
  - *Benefit Model*

Example: Specification => Data Structure => Model

```
Host ou4.cidds {
  Memory 256 MB;
  …
  Network {
    Interface eth0 {
      Name "10.0.0.11";
      …
} } }
Host ou3.cidds {
  …
  Network {
    Interface eth0 {
      …
} } }
Switch lan1 {
  Network {
    Interface eth0;
    Interface eth1;
} }
Network Topology {
    Link {
      Interface lan1 eth0;
      Interface ou4.cidds eth0;
      Speed 100 MB/sec;
      Latency 0.0 sec;
} … } …
```

```
struct NetworkInterface { string name; … }
struct Host {
  string name; long memory;
  sequence<NetworkInterface> interfaces;
}
typedef Host NetworkSwitch;
struct NetworkLink {
  sequence<StringPair> source_interfaces;
  sequence<StringPair> sink_interfaces;
  double link_speed;
  double link_latency;
}
```



QARMA Resource Management Algorithm Framework

- Optimization Problem
  - Allocate tasks to hosts
  - Set appropriate service levels (e.g., frame rate)
  - Optimize overall system benefit subject to real-time constraints.

- Approximation Approaches
  - Comparisons for various heuristics: greedy first-fit, best-fit, simulated annealing, genetic algorithms, branch-and-bound.
  - Pre-runtime analysis can use more advanced algorithms that take longer to run.
  - Runtime analysis must be fast and heuristics that incorporate strategies such as first-fit and best-fit show much promise.
  - Runtime analysis can utilize hints provided from *detectors* in the runtime system to reduce the complexity of algorithms.

QARMA Resource Management Algorithms

- Input
  - Software and resource specifications (System Repository)
  - Software and resource run-time state information (System Repository)
  - Hint sequences for run-time allocation invocation (interface argument)

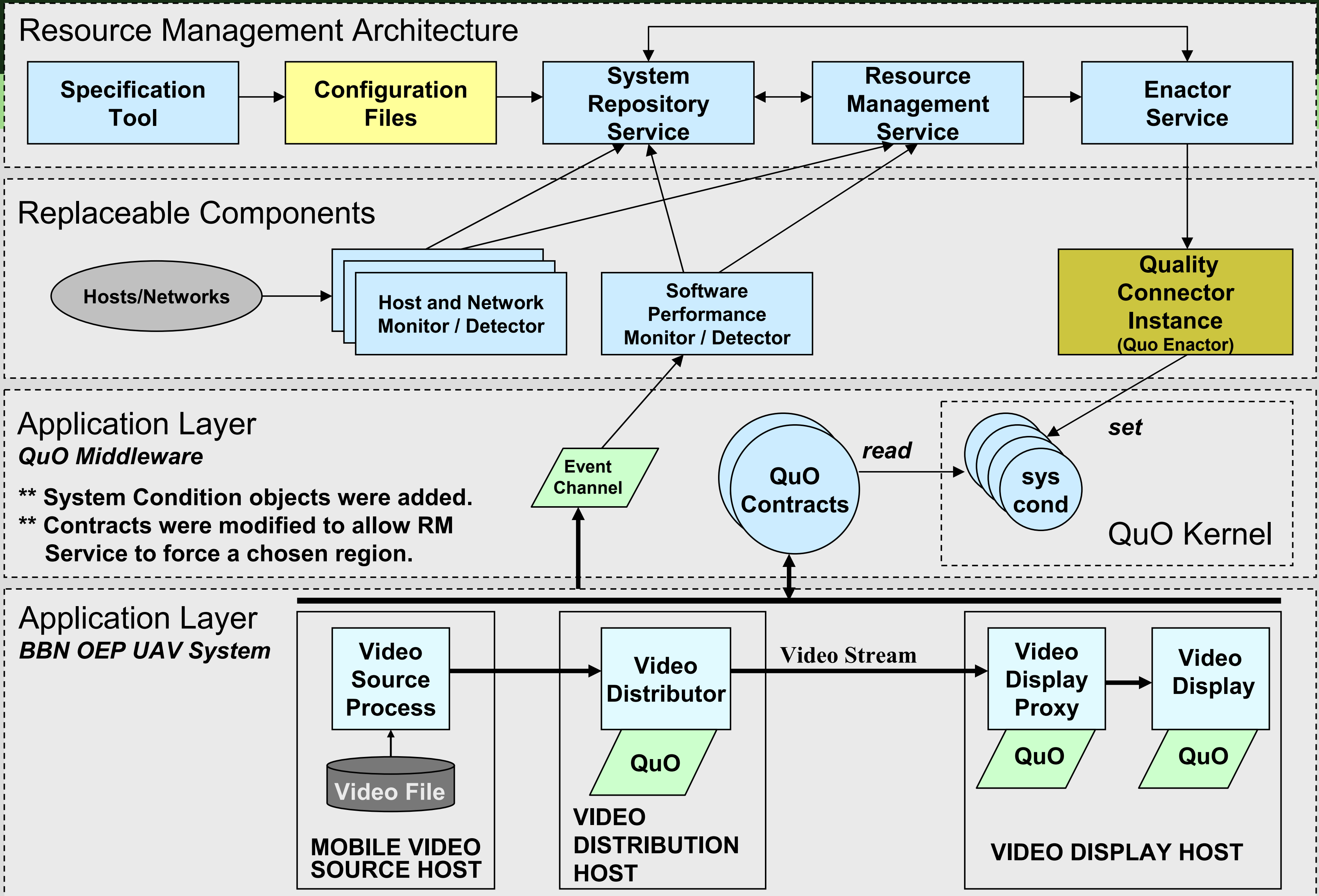
- Computation
  - Use monitoring data to update the efficient algorithmic models
  - Perform heuristic algorithm built on top of algorithmic models

- Output used to invoke control mechanisms
  - Service Level Adaptation
  - Process Allocation and Migration (Startup and relocation capabilities)
  - *Process Replication*
  - *CPU and Network Reservation and Prioritization*
  - *Dynamic Network Routing*

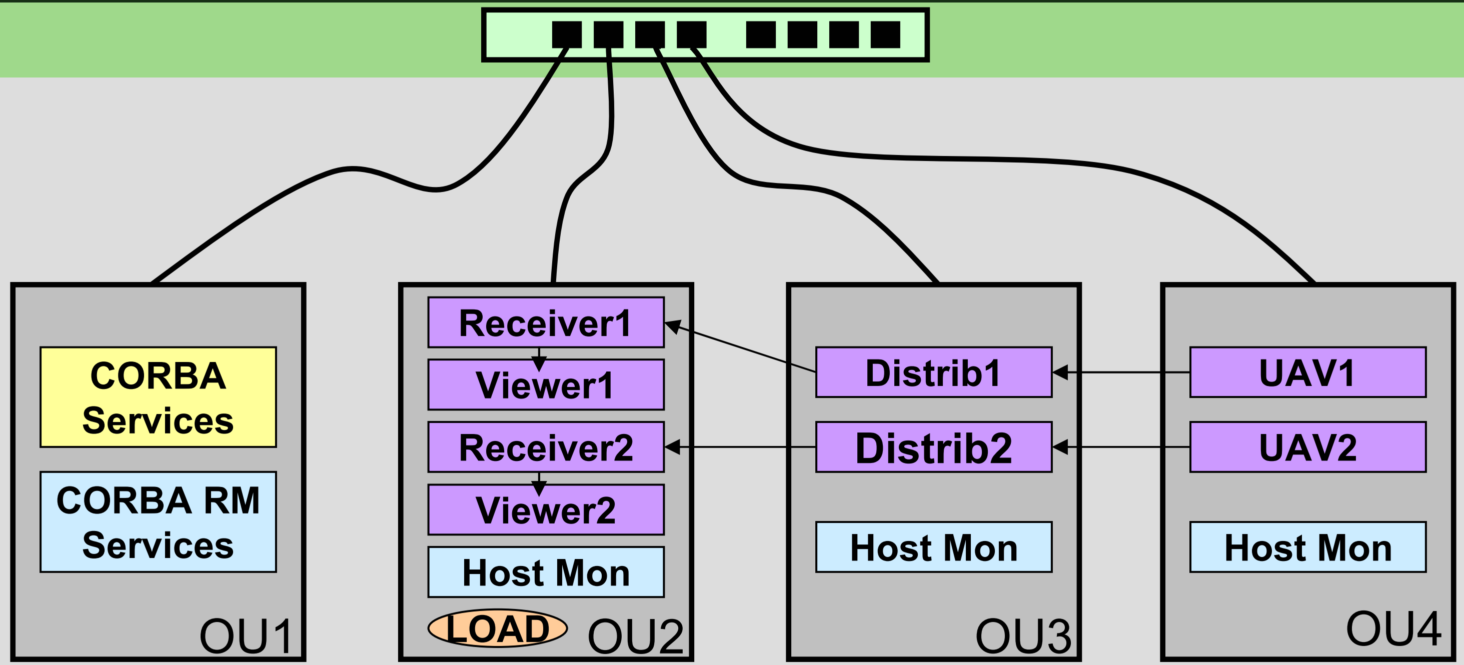
QARMA Enactor Service

- Purpose
  - Enact changes to the allocation of resources as directed by the Resource Management Service.
- Benefits
  - Delegates change directives to lower-level enactors.
  - Specification for enactors allows enactment mechanisms to be swapped out.

Integration With Video Distribution Software



Global Resource Management Demo Scenario

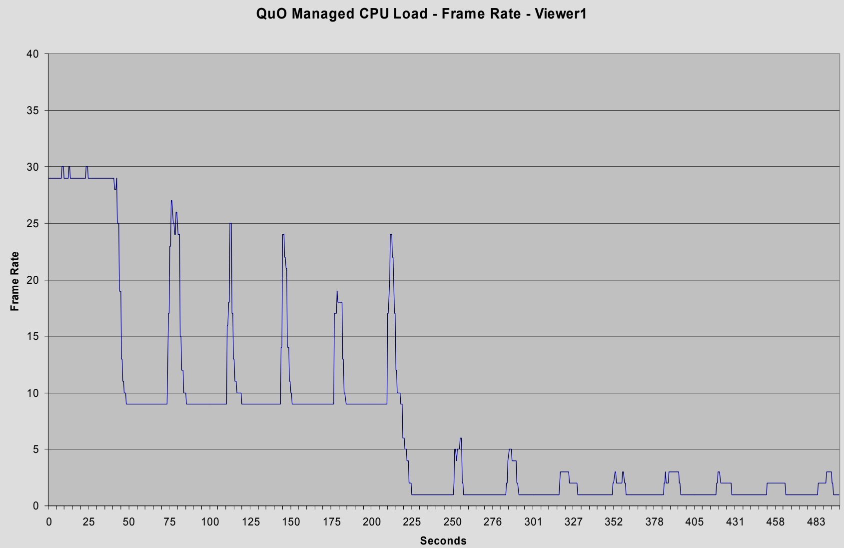


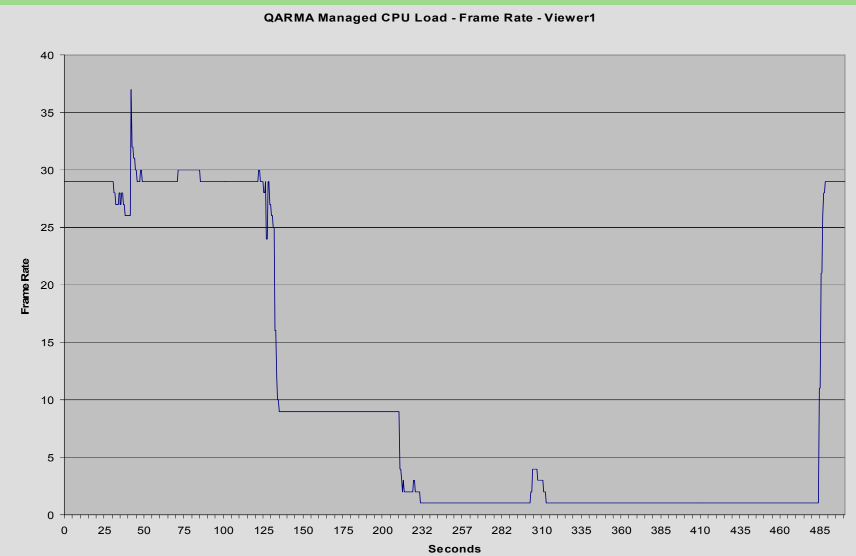
- There are two *Sender-Distributor-Receiver* streams.
- Receiver1 is set to the highest priority (because it is viewing a target).
- Use Hourglass to apply CPU load on the Receiver node.
- Load increases every 90 seconds as follows:
  - 40% at time 40
  - 60% at time 130
  - 90% at time 220
  - 98% at time 310
  - 100% at time 400

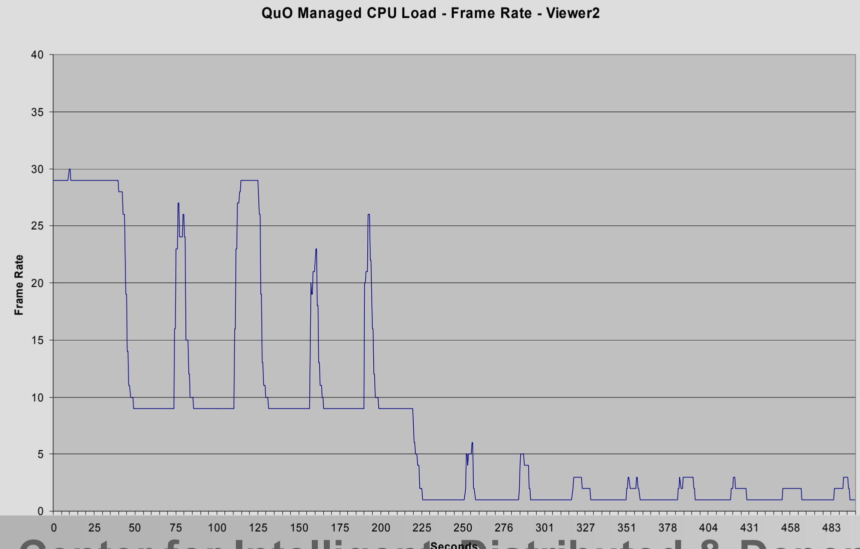
Global RM vs Local RM CPU Load Experiment Video Frame Rate

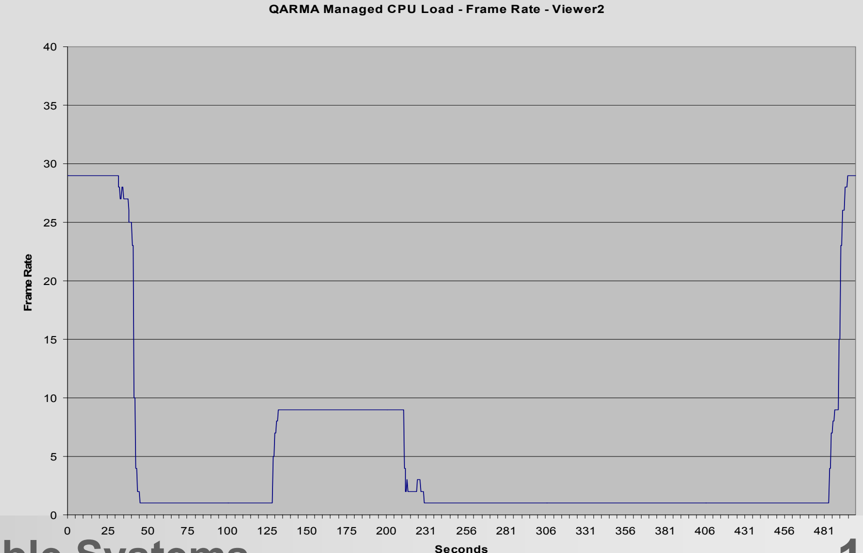
Local RM (No QARMA)

Global RM (QARMA)

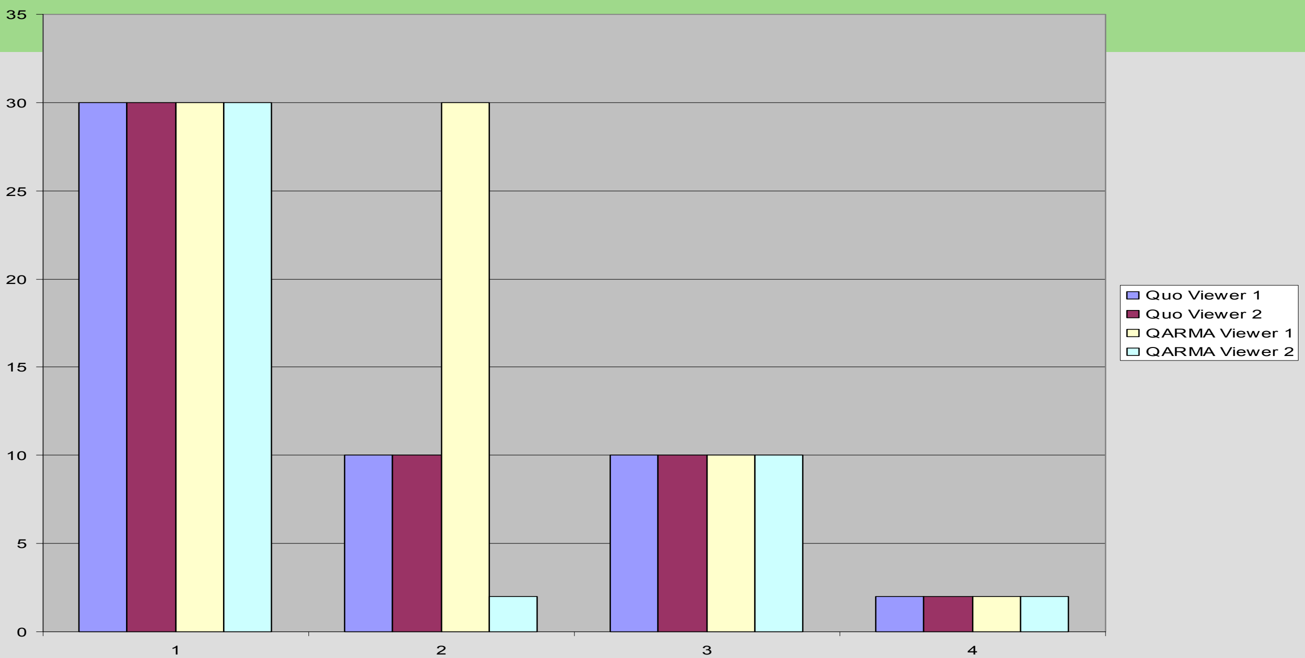








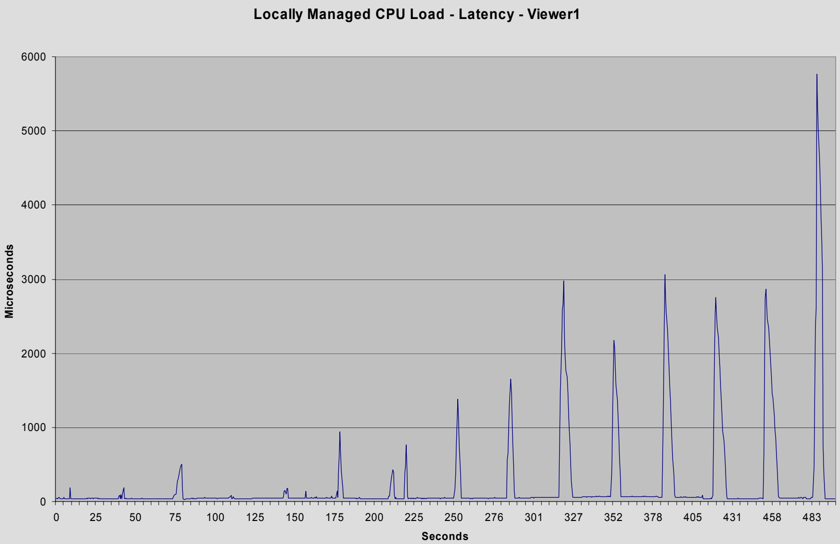
Global RM vs Local RM CPU Load Experiment Video Frame Rate

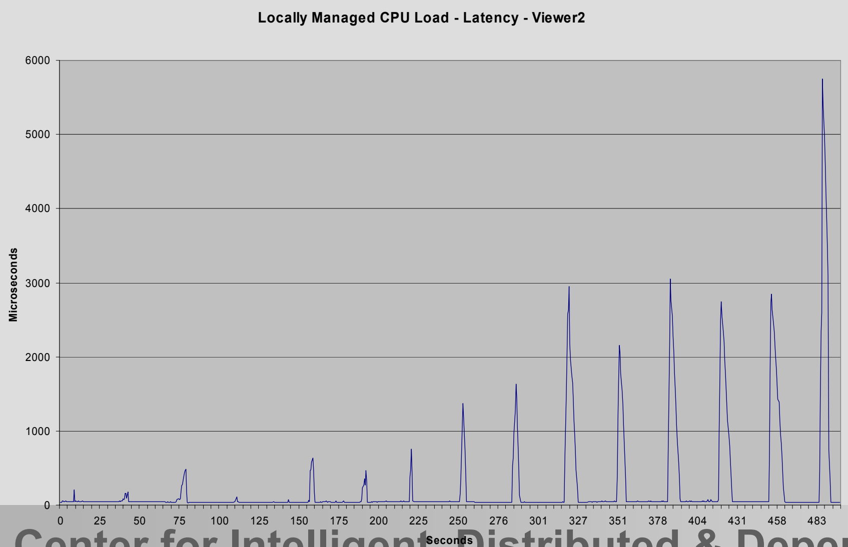


	Frame Rate Average (frames per second)
Baseline Viewer1	**5.48**
Baseline Viewer2	**6.01**
QARMA Viewer1	**8.73**
QARMA Viewer2	**2.67**

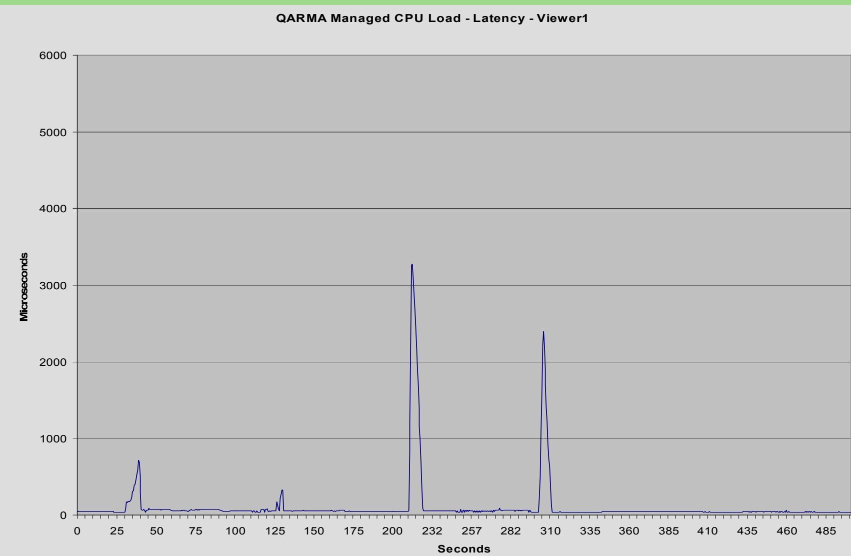
Global RM vs Local RM CPU Load Experiment Video Latency

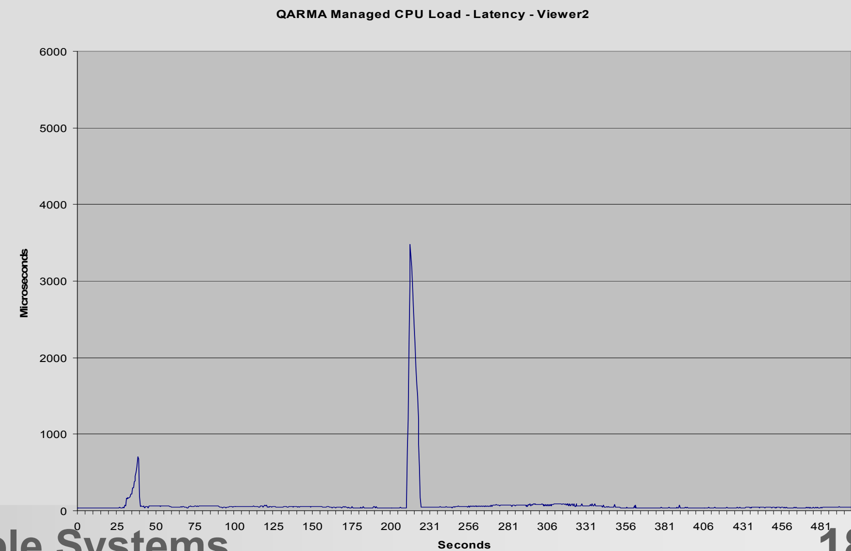
Local RM (no QARMA)





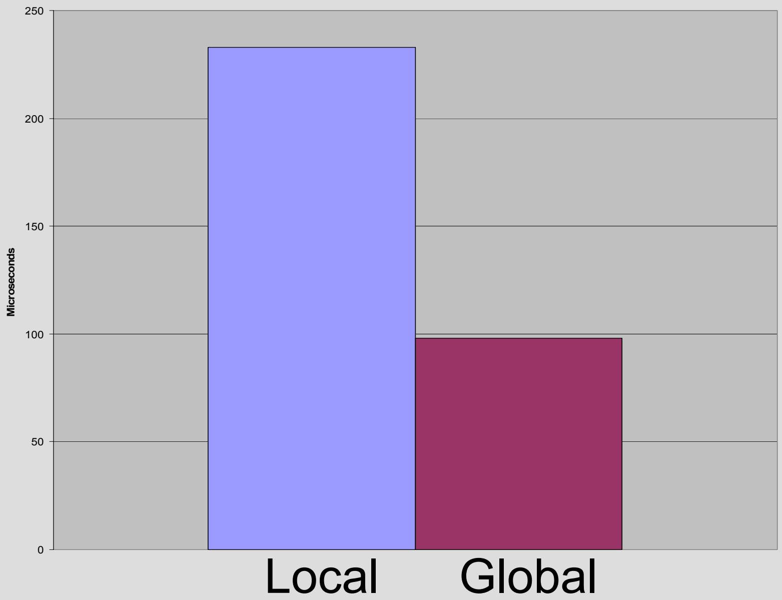
Global RM (QARMA)



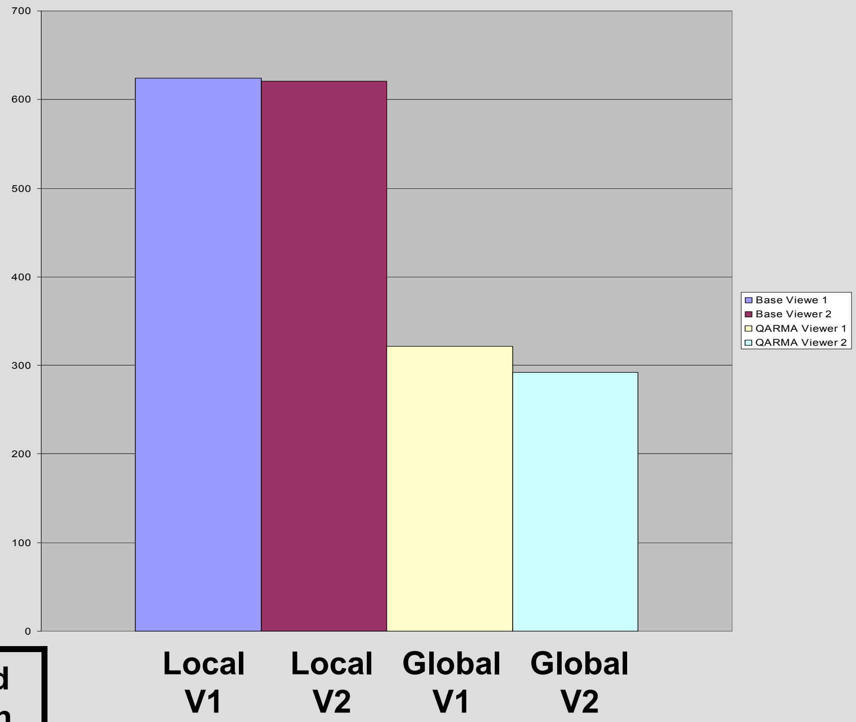


Global RM vs Local RM CPU Load Experiment Video Latency

Average Latency



Standard Deviation

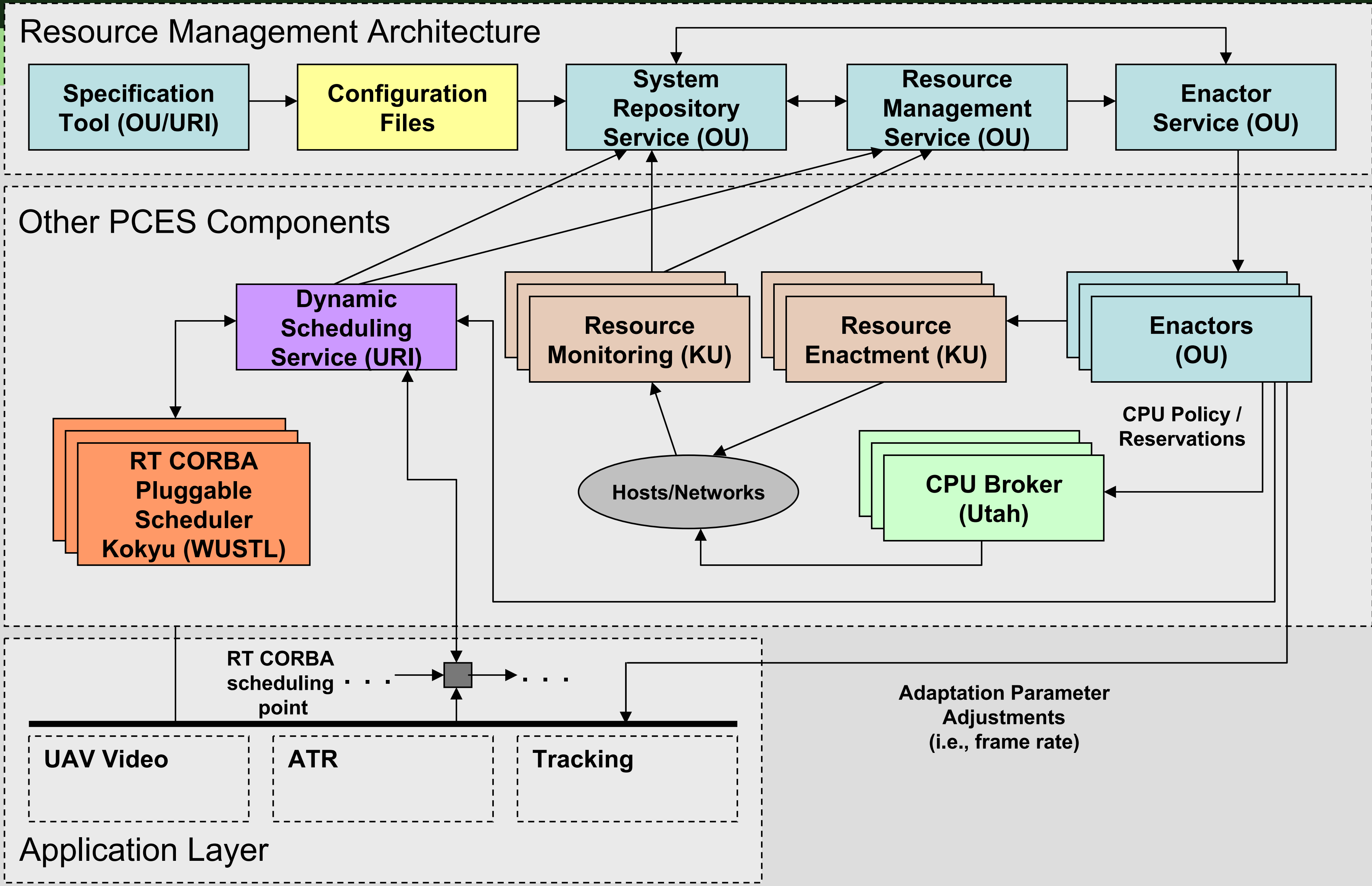


	Latency Average	Latency Max	Standard Deviation
Baseline Viewer1	236 ms	5770 ms	624 ms
Baseline Viewer2	229 ms	5751 ms	621 ms
QARMA Viewer1	106 ms	3270 ms	321 ms
QARMA Viewer2	89 ms	3479 ms	292 ms

Global RM vs Local RM CPU Load Experiment Conclusions

- Applications controlled by QARMA
  - Demonstrated greater stability
  - Exhibited greater determinism
  - Had lower average latency
  - Delivered higher frame rate for more important streams.
  - Delivered higher average frame rate for more important streams.

Envisioned PCES Component Integration



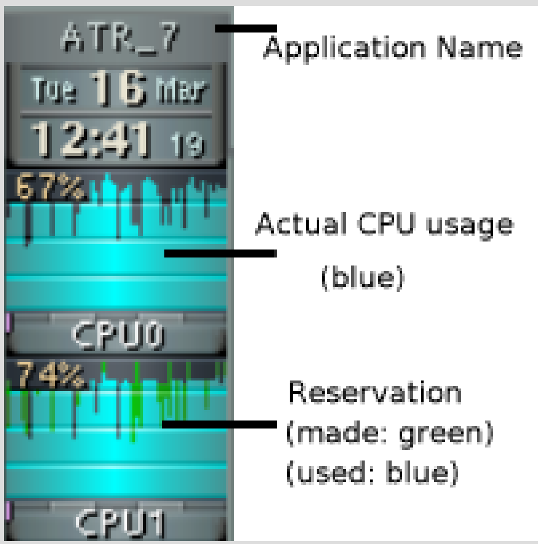
CPU Broker Integration

- PCES Phase II Experiments
  - The CPU Broker managed the applications on the single host that the receiver and ATR program were running on.

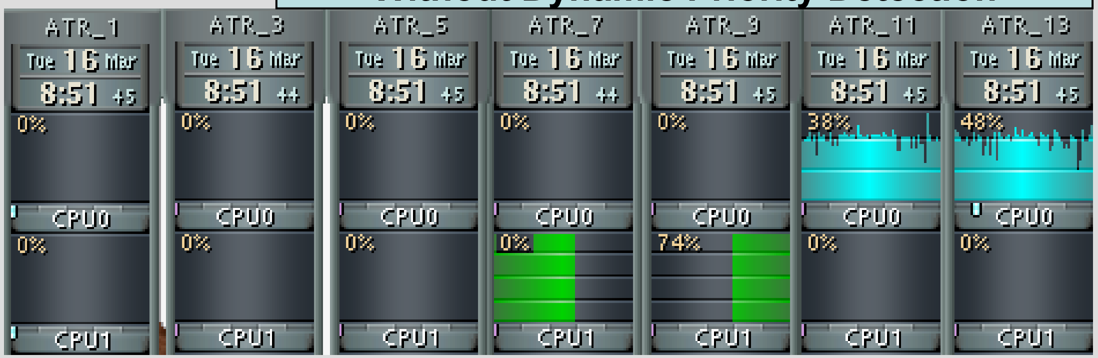
- Enhancement using QARMA Dynamic Priority Detection
  - Our coordination mechanism will allow CPU Brokers to be running on the distributor and sender hosts as well.
  - Importance can now be defined for a system (group or string of applications).
  - When the importance for a given UAV stream is modified, the Broker Coordinator determines which applications are in that system and the hosts on which those applications are running.
  - The Coordinator then invokes a CORBA interface to the CPU Brokers running on those hosts to change the priority of the applications in that system.

CPU Broker Experimental Results

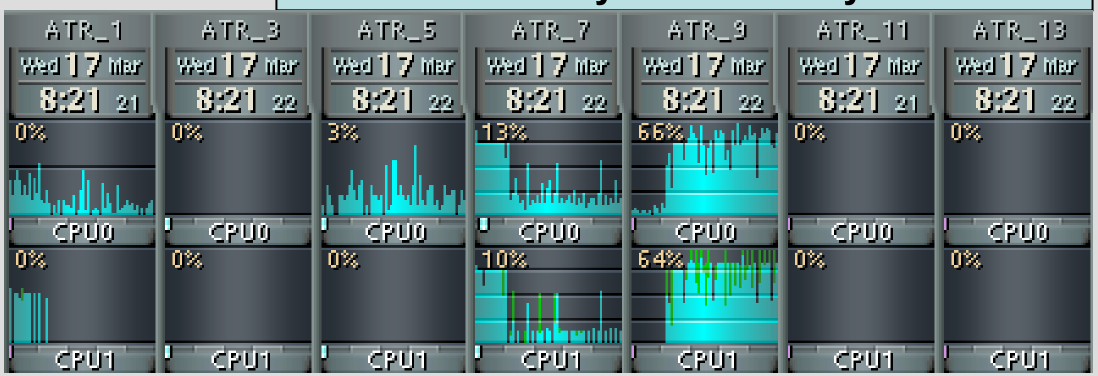
- Start the experiment with all streams at equal (lowest) priority.
- Increase stream 7 to the highest priority.
- Halfway through the experiment, set stream 7 to the lower priority and increase stream 9 to be the highest priority.



Without Dynamic Priority Detection



With QARMA Dynamic Priority Detection



Thanks to Matthew Gillen, now at BBN, for performing these experiments.

RMBench: Resource Management Benchmarking Services

- Facilitates the benchmarking and evaluation of resource management technology and middleware
  - QARMA Resource Management Architecture
  - URI Dynamic Scheduling Service
- Features
  - Provides language to generate arbitrarily large software systems that simulate real-world systems.
  - Provides language to define flexible and reusable experiments.
  - Each generated application provides a CORBA interface for adjusting runtime characteristics such as fidelity and workload.
  - Workload functions are used to provide variable CPU and Message size parameters.
  - An *experiment controller* process reads experiment files in order to produce a repeatable *dynamic* environment in which sender and receiver parameters are changed during runtime.
- Latest Enhancements
  - Variable service level added to the applications generated by RMBench.
  - Enactor to control service level of RMBench applications added to QARMA.

Future Work

- Integrate other resource management control mechanisms
- Investigate and develop more advanced allocation algorithms
- Provide fault tolerance and stabilization of the Resource Management Service as well as reflexive management.
- Continue integration, transition and validation efforts
  - Validation with RMBench
  - Validation with PCES OEP (Boeing and BBN)
  - Integration with PCES technology
    - Scheduling Service, Kokyu, CPU Broker
  - Transition into DARPA ARMS MLRM infrastructure
  - Transition into TAO as CORBA Services

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.