

# Quality of Service Management for Real-Time Embedded Information Systems

Christopher D. Gill

Department of Computer Science

Washington University, St. Louis

[cdgill@cs.wustl.edu](mailto:cdgill@cs.wustl.edu)

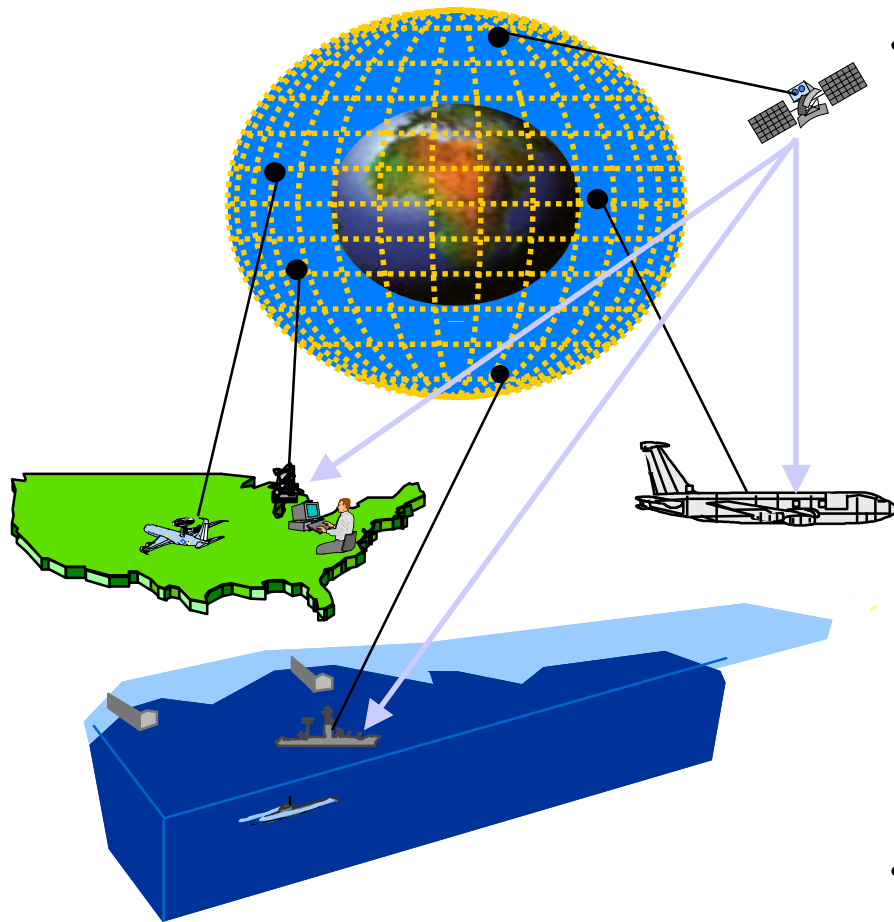
[www.cs.wustl.edu/~cdgill/PDF/DASC00\\_slides.pdf](http://www.cs.wustl.edu/~cdgill/PDF/DASC00_slides.pdf)

19<sup>th</sup> IEEE/AIAA DASC

Thursday, October 12, 2000

Research supported by: The Boeing Company/AFRL,  
DARPA contract 9701516, Sprint, and Siemens AG

## Research Motivation



Adapted from “The Future of AWACS”,  
by Lt. Col. Joe Chapa

- Meeting the QoS requirements of distributed real-time mission critical systems is **hard**
  - Both critical & non-critical timing constraints
  - Must adapt to rapidly changing environments
  - Possibly heterogeneous endsystems & networks
  - Must shield developers from QoS management complexity
  - Cost containment through portability & reuse
- Standards-based COTS solutions are essential, but many open research issues remain



## Research Context

### **Previous Work:**

- Generality and flexibility of resource scheduling strategies [Gill, Levine, Schmidt, "The Design and Performance of a Real-Time CORBA Scheduling Service", Real-Time Systems (special issue on middleware), to appear 2000]
- Decoupling QoS management from application logic [Doerr, Sharp, "Freeing Product Line Architectures from Execution Dependencies", Software Technology Conference, 1999]
- Support for performance analysis and visualization [Gill, Levine, O'Ryan, Schmidt, "Distributed Object Visualization for Sensor-Driven Systems", DASC, 1999]
- New challenges for system resource allocation: adaptation [Doerr, Venturella, Jha, Gill, Schmidt, "Adaptive Scheduling for Real-Time Embedded Information Systems", DASC 1999]

### **Ongoing and Future Work:**

- New challenges for system resource allocation: distribution
- Canonical QoS management patterns

# Experimental Configuration

## ***Schedule Dispatching Approach:***

- Off-line generation of static scheduling and configuration info
- Static or hybrid static/dynamic run-time dispatching
- Option to cancel futile non-critical operation dispatches

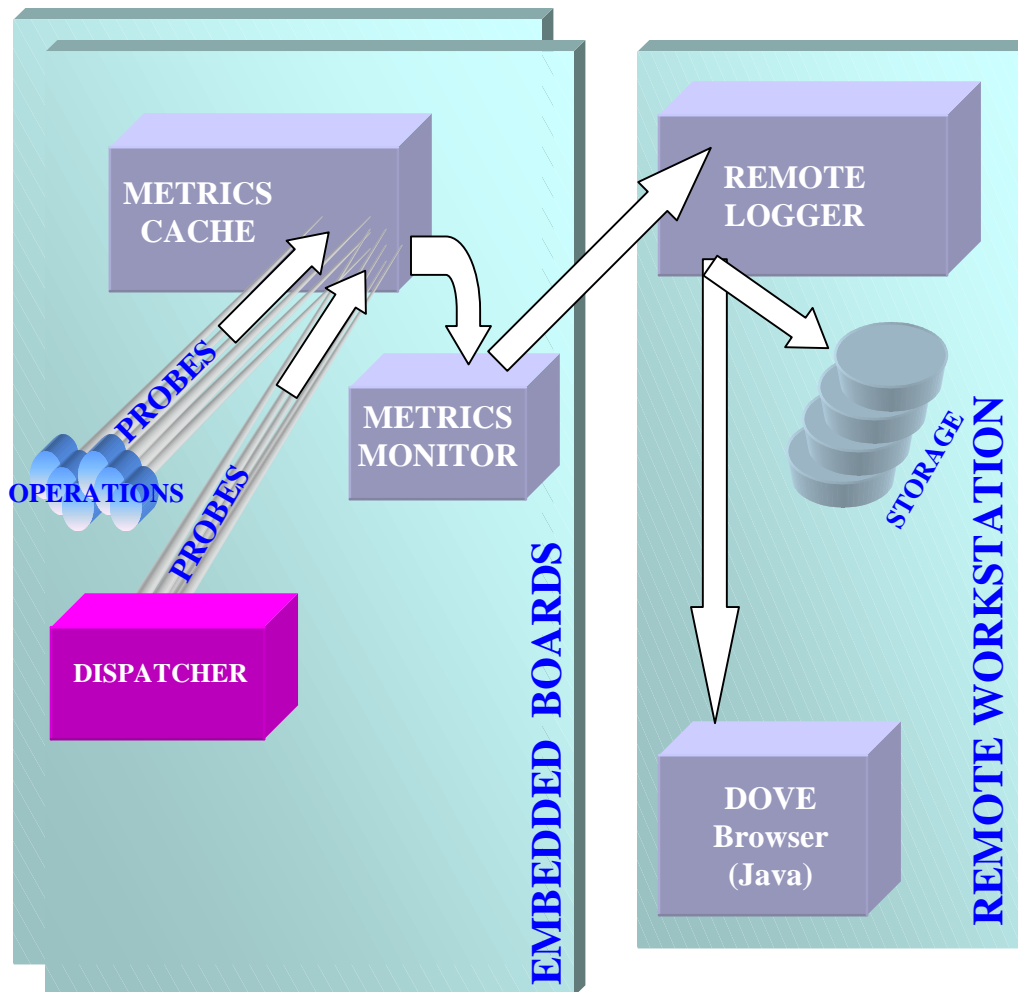
## ***Environment:***

- Realistic application (~30 schedulable dispatch points)
- TAO RT Event Channel and TAO ORB
- VxWorks 5.3 on 2 single-board 200Mhz PPC computers
- VME between boards, Ethernet to remote NT workstation

## ***Goals:***

- Quantify performance of static & hybrid scheduling strategies
- Demonstrate hybrid scheduling benefits in a realistic setting
- Visualize real-time behavior from a remote workstation

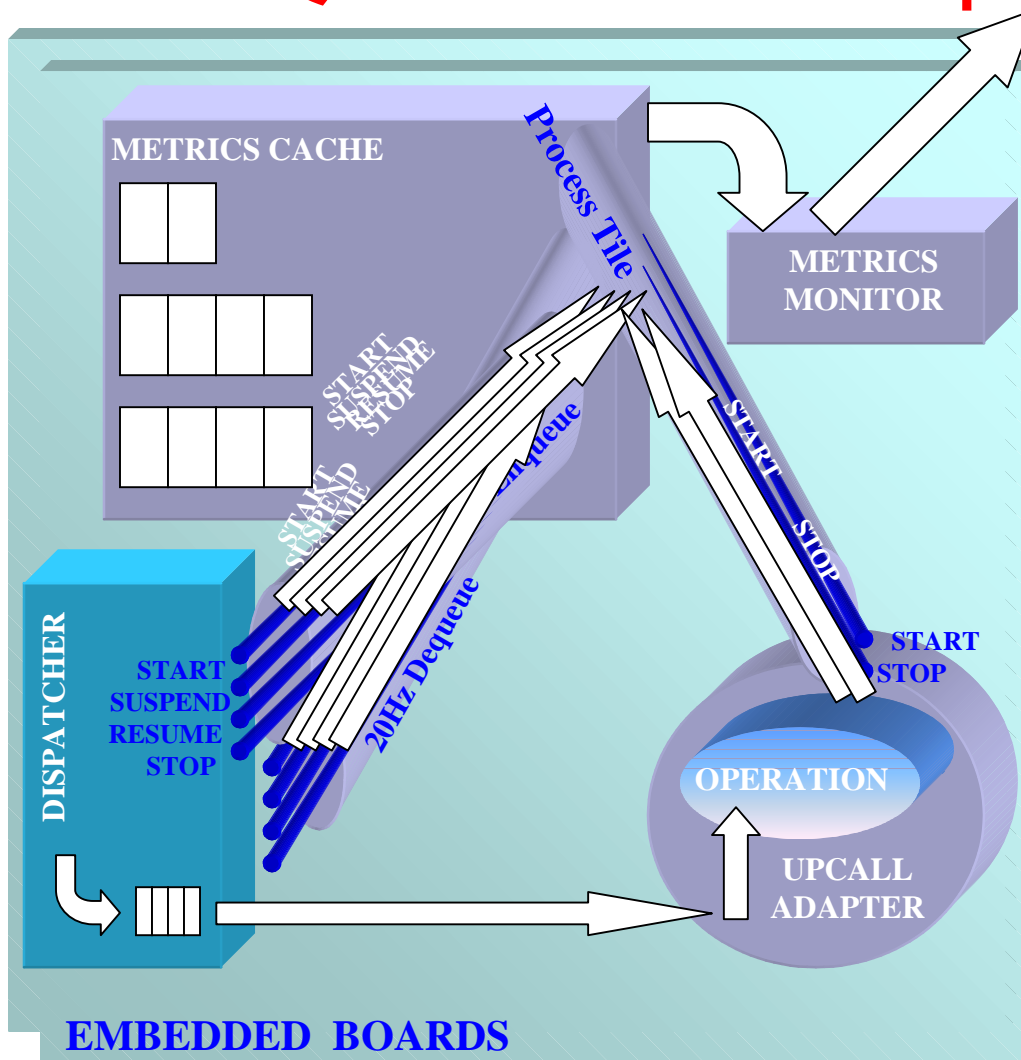
# QoS Metrics Framework Overview



## **Data Collection**

- Probes populate metrics cache
- Monitor periodically digests raw data, streams to logger
- Logger streams data to storage, viewers

# QoS Metrics Implementation



## **Operation Adapters**

- Intercept operation upcalls
- Allow dispatch cancellation

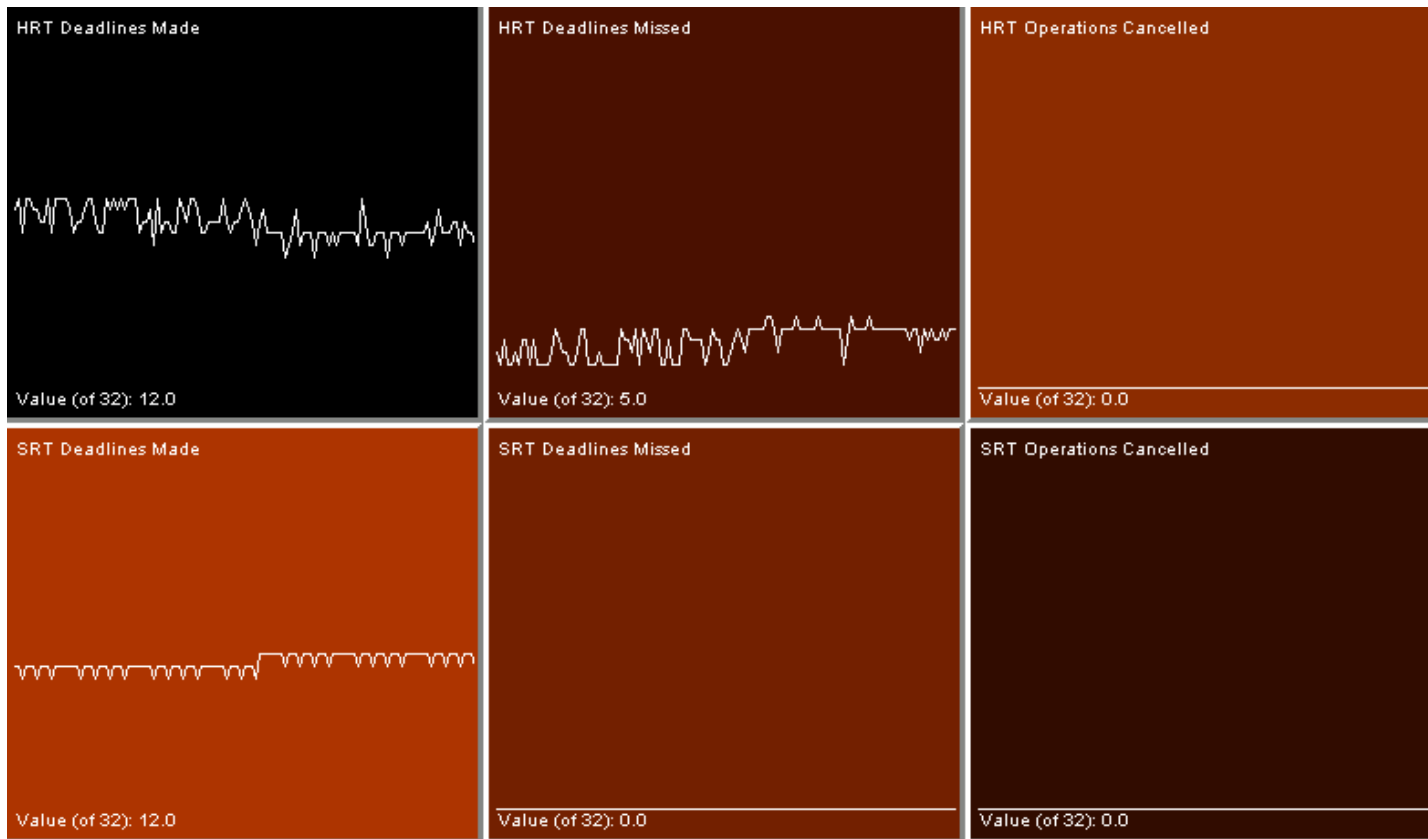
## **Efficient Metrics Cache**

- System is instrumented with C++ probe macros
- Each probe has a name/id
- Each probe macro writes to a specific location in a singleton metrics cache
- Each probe has one or more macros, i.e., start, suspend, resume, stop

## **Metrics Monitor**

- Harvests data periodically from the metrics cache

# Empirical Results: RMS Strategy

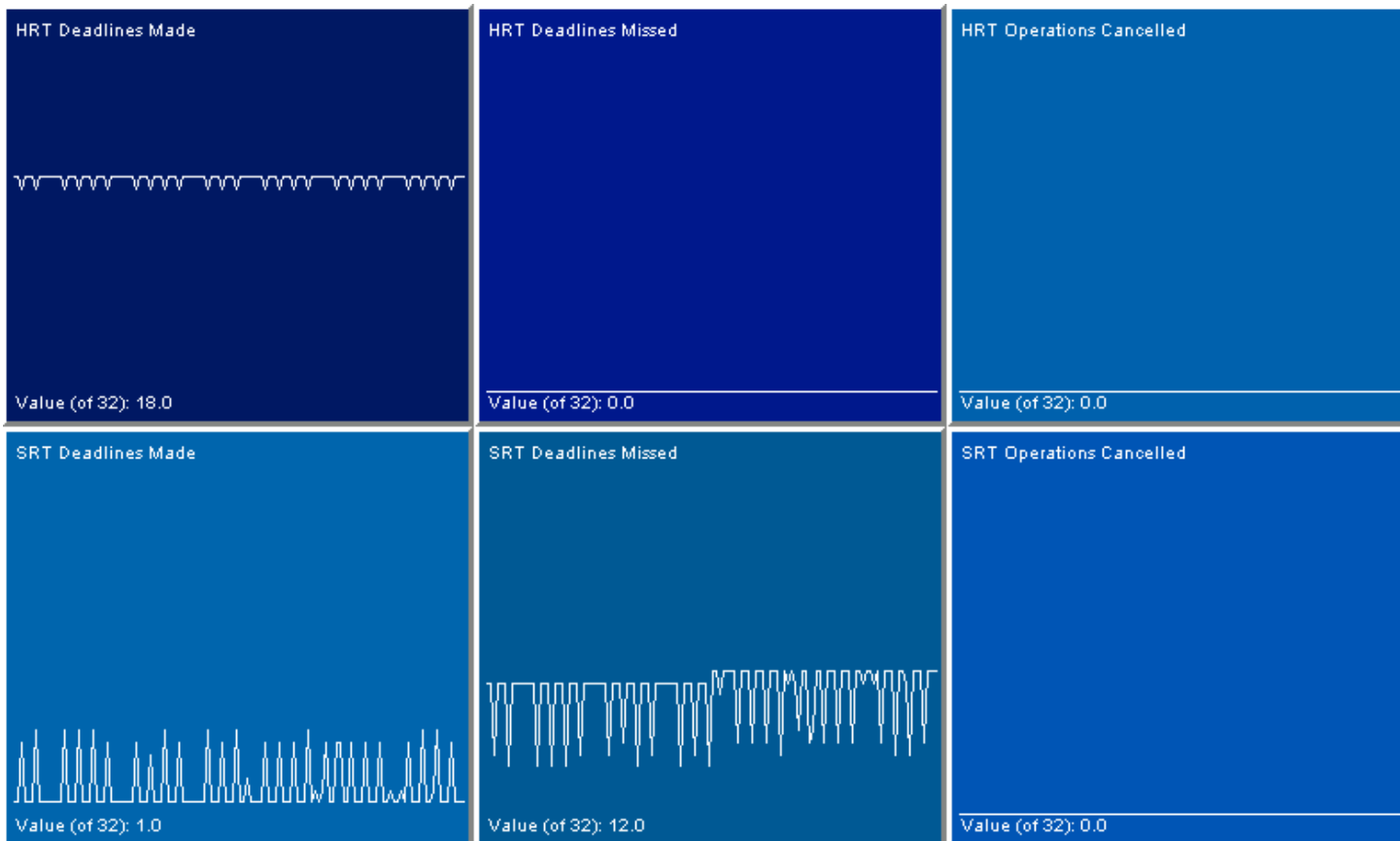


## The RMS Dilemma

- The set of critical operations must be feasibly schedulable
- Adding non-critical operations is to increase functionality and CPU utilization
- However, RMS does not protect critical operations under overload conditions
- As number of non-critical operations increases, impact on critical operations can worsen

*From experiments using a realistic application on realistic hardware. Transition is from an already overloaded state into another with even higher load, due to admission of even more short-duration non-critical operations into the schedule.*

# Empirical Results: MUF Strategy

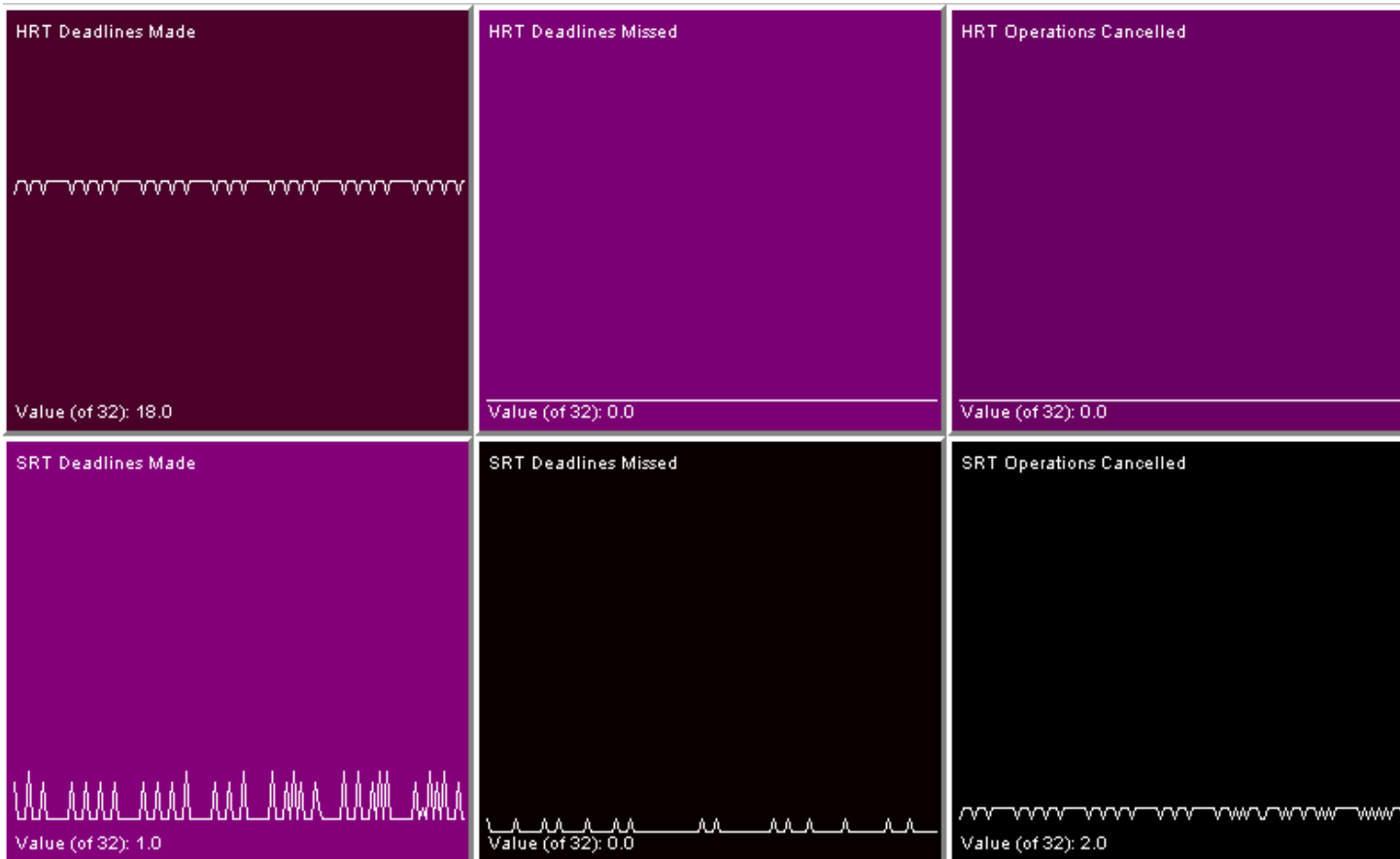


## A Hybrid Solution

- Maximum Urgency First: hybrid static/dynamic scheduling strategy
- Protects critical operations when overloaded
- Even as the number of non-critical operations increases, minimal impact on critical operations
- Operations are dispatched whether or not they will make their deadlines

*From the same experiment and same state transition shown previously for RMS. Here, all critical operations make their deadlines, and some non-critical operations make their deadlines, while others miss theirs.*

# Empirical Results: MUF + Cancellation

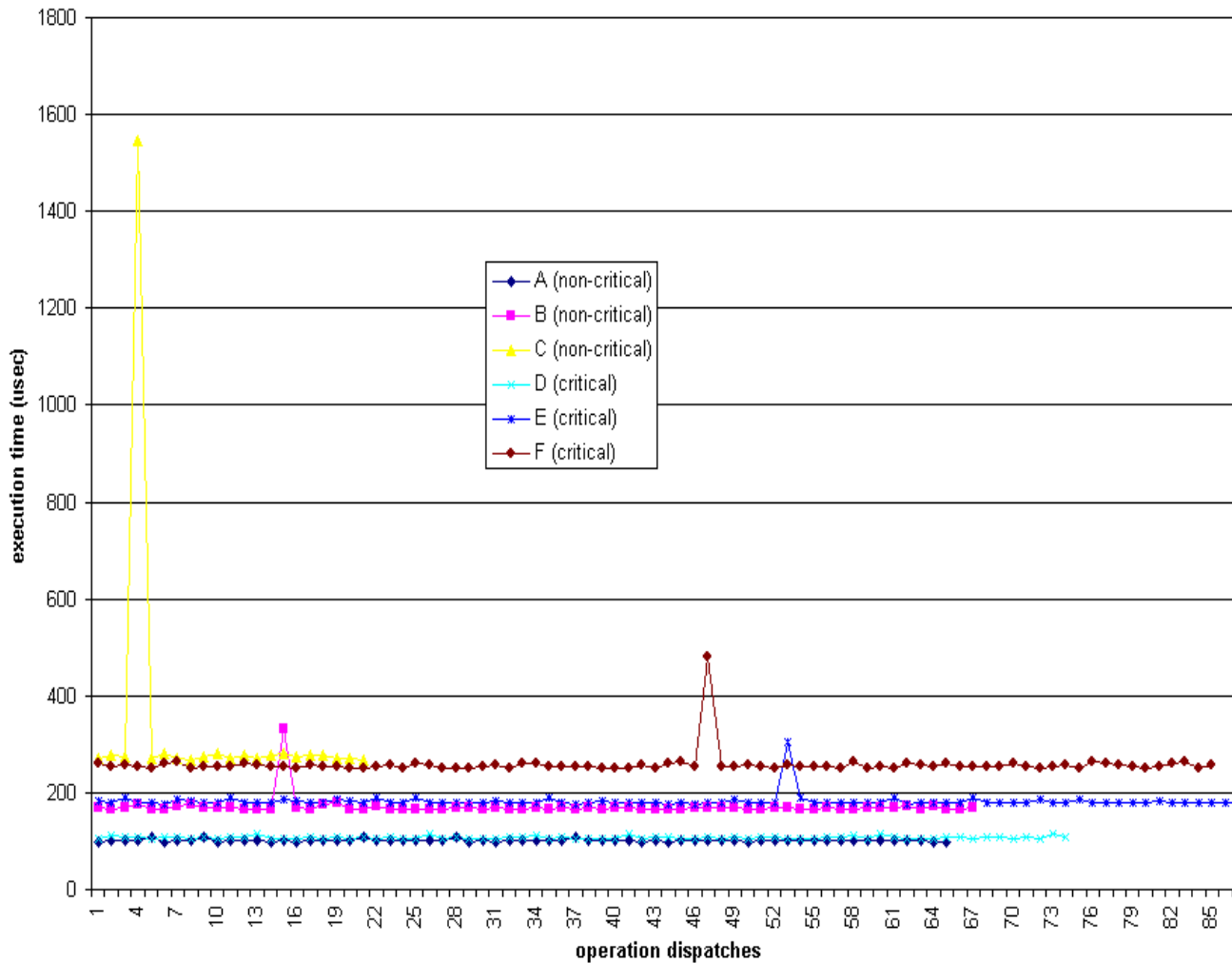


## Refining The Solution

- Operations that miss deadlines have lower (e.g., zero) value
- Canceling futile operations reduces wasted CPU usage
- Cancellation decision made at the point of application component dispatch, via an Adapter
- Trade-off between cancellation decision overhead and additional control over wasted work

*From the same experiment and same state transition shown previously. Reduced number of missed non-critical deadlines, though the number of non-critical deadlines made was also lower due to pessimistic (WCET) cancellation strategy.*

# Empirical Results: Execution Stability



## MUF Results

- Time probes measured time spent in dispatched upcalls
- Reasonably well-bounded execution times for both non-critical and critical operations
- Anomalies for critical operations attributed to non-determinism in experimental setup (network interrupts)
- Heavier queue loading may occur with more diverse non-critical utilization
- However, operation cancellation should help mitigate loading effect

## Concluding Remarks

- Empirical results show hybrid QoS management techniques can be employed effectively and beneficially in a realistic real-time embedded information systems environment
- Fine-grain timing data capture and visualization techniques are useful to examine the benefits and trade-offs of various QoS management approaches
- Further studies are underway for coordinated adaptive and distributed resource management in a realistic real-time embedded information systems environment