THE Open GROUP

# Fault Management Based on Quality of Service Criteria

**First TAO Conference**
**6 August 2001**

Douglas Wells
The Open Group
d.wells@opengroup.org




## Overview

THE Open GROUP

- Premise

  There are many analysis techniques available from dependability and system critical disciplines that can be usefully extended for use in QoS-based, real-time distributed systems

- Outline
  - Description of the problem space
  - FFD—an initial experiment
  - Relationship to CORBA/TAO
  - Extrapolation to a research problem




## Distributed, Real-Time Systems

THE Open GROUP

- Real-time—timeliness is part of correctness
  - Requires expression of time constraints
- Reason for missing a “deadline” is irrelevant
  - Resources overloaded
  - Algorithmic overrun
  - Component failure
- Design must consider failure of components
- Special real-time computer discipline
  - The tighter the time constraints, the more stringent the design and coding restrictions
- QoS techniques regularly being applied here




## Fault Management

THE Open GROUP

- Concept includes (at least)
  - Failure detection
  - Failure isolation
  - Failure reporting
  - Failure recovery/masking
    - Including common use of term “fault tolerance”
  - Fault prediction
- SEI/IFIP WG10.4 taxonomy/definitions
  - Fault: adjudged or hypothetical cause of error
  - Failure: behavior different than was intended



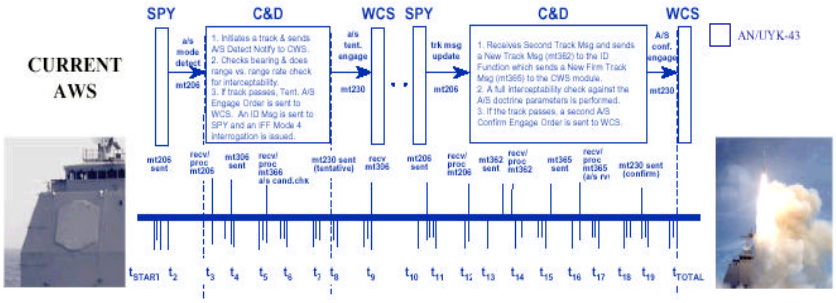

## HiPer-D: An Example Context

THE Open GROUP

- High-Performance Distributed (HiPer-D) Project (at NSWC in Dahlgen, VA) is applying COTS-based, distributed computing techniques to ship-board weapons systems
- AAW (ship self-defense) is “hard” real-time
  - Mandated timing requirements
  - Mandated failure recovery requirements
  - Program objective for capacity scalability
- Use of group communications in design




### Why DD-21 Needs Assured Response: SPY Radar Auto-Special Time-Line

## Group Communications

THE Open GROUP

- Reliable multicast technique based on atomic multicast—each message is reliably delivered to either (exclusive or)
  - —all designated recipients
  - —no recipients
- Popularized by Ken Birman at Cornell U.
  - Initial research/product was Isis
  - Current implementation is Ensemble




## (Simplified) Overview of Group Communication Operation

THE Open GROUP

- Start with known set of group members
- Message is sent (multicast) to agent on host node of each recipient
- Receipt acknowledgements are exchanged
- When all nodes have acknowledged, release message to each application group member
- Otherwise—after a time-out event occurs
  - Reform group by ejecting tardy members
  - Restart message delivery process with new group membership set

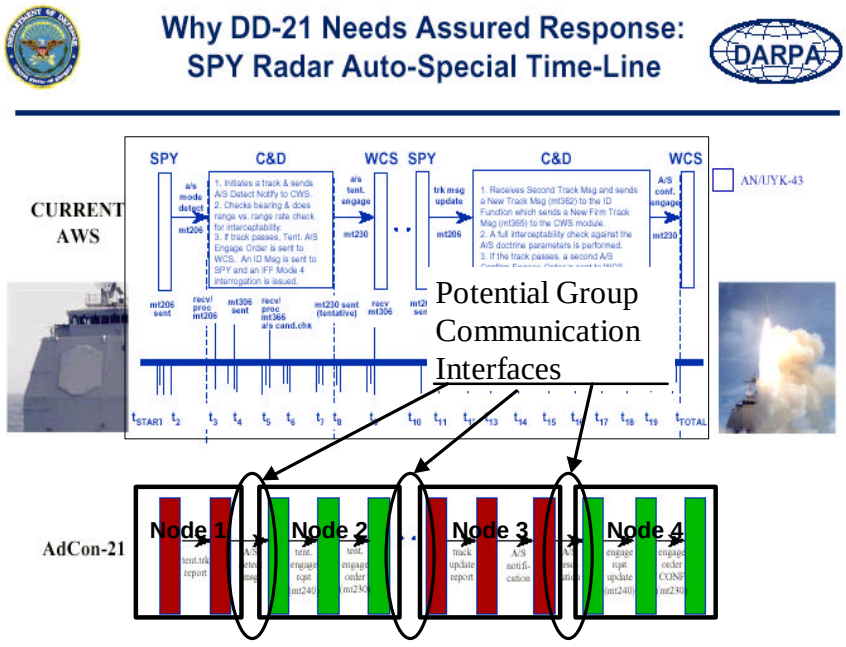



## Observations

THE Open GROUP

- Use of time-out is derived from requirement that timeliness is more important than tardy operation at full capacity
- Time-out event transforms timing fault into an (apparent) component failure
- Individual message delivery time-outs typically must operate an order of magnitude faster than overall system time constraint
- Example
  - End-to-end 1 second deadline might require 0.1 second time-out at each stage of group communications



## Problems in Utilizing Real-Time Group Communications

THE Open GROUP

- Time constraints in RT systems must be met even in extreme conditions, not just in speed-of-light micro-benchmarks
- Group communication time-out periods are often of same order of magnitude as scheduling jitter in non-RT OS's
- False positives (tardy nodes declared dead), while handled correctly, are expensive
  - Node is forced "down," then allowed to rejoin
  - Requires reacquisition of application state
- COTS components (Isis, Ensemble) not designed using real-time techniques




## More Problems in Real-Time Group Communications

THE Open GROUP

- Different interfaces have different timing constraints. A node may be declared down in one context, but must remain "up" in another.
  - Notional interface time-out periods
    - HiPer-D AAW path: 0.5 second
    - Instrumentation: 3 seconds
    - Resource Management 10 seconds
  - Timing constraints (and time-outs) are usually associated with an interface to an external component—not an entire application
  - Note: this problem is not limited to group communication interfacess




## Fast Failure Detector (FFD) Objectives

THE Open GROUP

- General Goal of FFD:
  - Provide faster, more reliable detection of host node failure than other components
- Specific Goal of FFD Integration Effort:
  - Detect and report host failure within 250 msec
  - This should allow an application to recover from a host node failure within 1 second, even with a substantial state reacquisition cost




## FFD Design Considerations (i)

THE Open GROUP

- FFD (and Ensemble) utilize heartbeat (watch-dog/dead-man timer) pattern
  - Generation and monitoring of heartbeat messages (via time-outs) is a common method of detecting node crash failures
  - Reducing timeouts on missing heartbeat messages allows faster identification of failed nodes and thus supports shorter deadlines
  - Heavy loads cause queuing delays (jitter), which cause heartbeat messages to be tardy, which cause time-outs, which cause nodes to be erroneously declared down, which cause expensive, unnecessary reconfigurations




## FFD Design Considerations (ii)

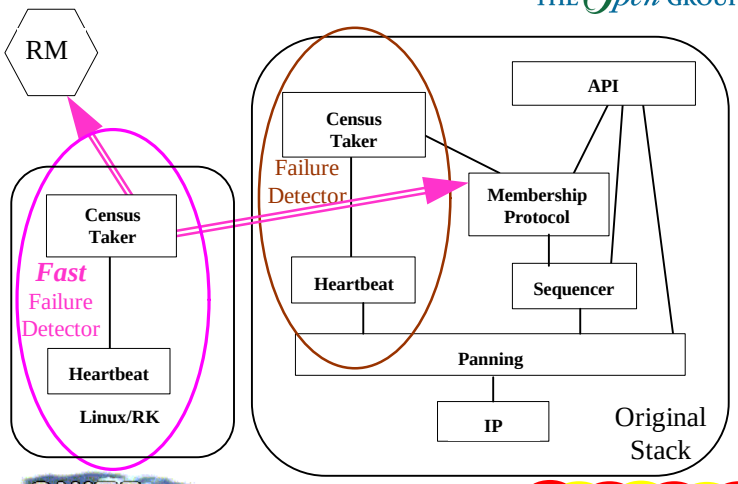
THE Open GROUP

- Assertions on Host Failure Detection
  - Providing dedicated resources for heartbeat generation and monitoring functions can reduce jitter, thus allowing use of shorter timeouts, thus improving real-time properties
  - Dedicated resources can best be provided in a separate host failure detector component that has been specifically designed to support real-time properties




## Group Membership Protocol Stack

THE Open GROUP

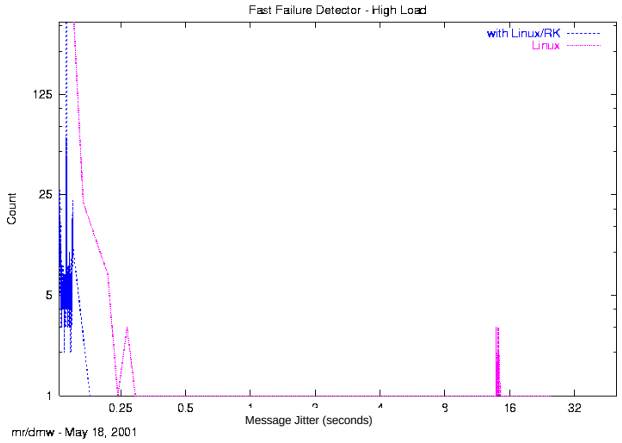






## FFD Message Latency (Jitter) Characterization

THE Open GROUP






## Note on Resource Consumption

THE Open GROUP

- Test-bed: 5 nodes, 10 Mbps Ethernet® LAN
- FFD parameters
  - Time-out period: 0.5 second
  - Replication factor: 5 (i.e., 100 msec heartbeat)
- FFD uses <1% of 100 Hz, 32 MB PC
  - Note: value is imprecise due to use of pseudo-Monte Carlo measuring technique in UNIX® and Linux®
- FFD uses <5% of network bandwidth
  - Note: value is minimum value reported on hub

## Some Simplifying Assumptions for First-Order Fault Analysis

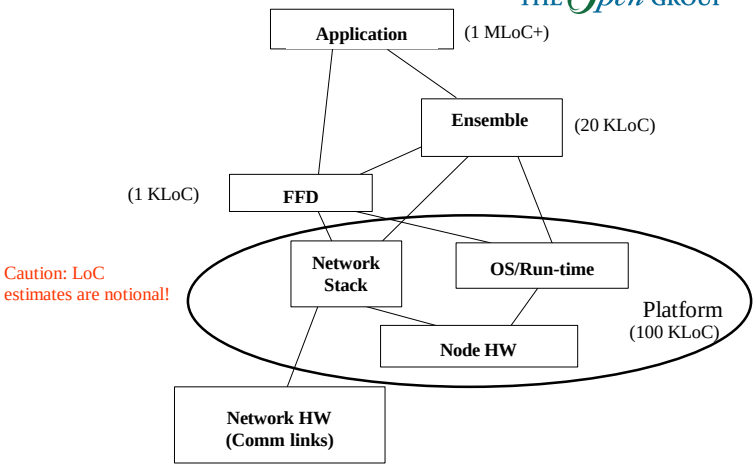
THE Open GROUP

- A component failure is due to either internal fault, environmental fault, or failure in other ("depends upon") component
- Internal component failure rate is proportional to number of errors (bugs) in it
- HW component bug count is proportional to transistor count
- SW component bug count is proportional to lines of code (LoC)




## (Simplified) Fault Dependency Graph of Node Failure Detection Function

THE Open GROUP







## First-Order Fault Analysis

THE Open GROUP

- Examine projected failure rates of fielded components based on bug rates (br)
- Example failure rates (fr) w/o FFD
  - fr(Ensemble) ≈ br(20K) + fr(platform) + fr(net)
  - fr(application) ≈ br(1M) + fr(Ensemble)
- Example failure rates w/ FFD
  - fr(FFD) ≈ br(2K) + fr(platform)+fr(net HW)
  - fr(Ensemble´) ≈ fr(Ensemble) + fr(FFD)
- Therefore
  - FFD should be more reliable than Ensemble or application




## Failure Detection Types and Failure Correlation

THE Open GROUP

- True negative: normal operation
- True positive: correctly detected failure
- False positive: erroneously asserted failure
  - Will wastefully perform system reconfiguration
- False negative: overlooked a failure condition
  - Unable to mask failure
  - May lead to overall system failure
- False positives can be tolerated as long as there aren't "too many" of them
- False negatives can potentially lead directly to system failure




## Reexamination of False Negatives

THE Open GROUP

- Multiple failures leading to false negatives in FFD are likely to be either highly correlated or highly non-correlated, not in between
- Highly non-correlated case (ordinary failures):
  - Timing constraint violation is likely to be "made up" by other portions of application chain
  - Result: no system failure
- Highly correlated case (e.g., battle damage):
  - All failures are likely to occur simultaneously
  - Failure recoveries will occur simultaneously
  - Result: similar to highly non-correlated case




## Relationship to CORBA/TAO

THE Open GROUP

- Why objects?
  - Objects are likely to exist at edges of failure domains, but are ***not*** likely to straddle them
    - Objects contain complex, interrelated state
    - Objects can utilize thin-wire communication
- Why CORBA?
  - CORBA objects have names (e.g., IOR); so, the dependency structure can be modelled
  - CORBA specifications are open and available
  - Several CORBA specifications are relevant
- Why TAO?
  - TAO is open and available




## Extrapolation to a Research Program

THE Open GROUP

- There must be some method to this madness
- More data are available than can be analyzed with existing techniques
- Need to extend QoS throughput techniques to fault management
  - Identify metrics that can be correlated
    - For example, reliability and availability
  - Develop FM resource management policies
  - Identify and develop more, reusable, adaptable components, similar to FFD
  - Develop and adapt techniques for failure correlation patterns in “ordinary” applications