Architectural Support for Mode-Driven Fault Tolerance in Distributed Applications

Deepti Srivastava and Priya Narasimhan

Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA, USA

Motivation

- Fault tolerance for Unmanned Aerial Vehicle (UAV) distributed application
  - Written in middleware to run across nodes of a distributed system
  - Has *modes:* multiple distinct behaviors that constitute distributed changes
    - Surveillance, Target recognition, Tracking, Feedback/Control
- Each mode has
  - Different critical components
  - Different latency requirements
  - Different resource usage profiles
- Many other distributed applications are multi-modal
  - x-by-wire cars – low rpm, high rpm, parked
  - Space Shuttles – take-off, on-orbit, landing
- "One-fault-tolerance-solution-for-all-modes" is not useful

Overview

- Mode-Driven Fault Tolerance Philosophy
- An Architecture for Mode-Driven Fault Tolerance
- Case Study
- Conclusion

Mode-Driven Fault Tolerance (MDFT)

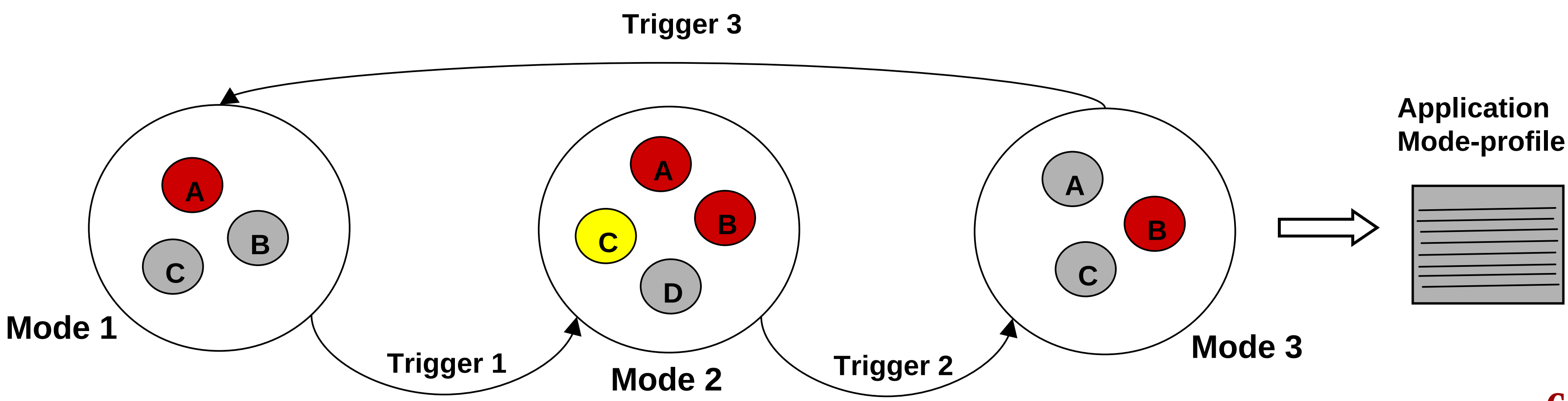
- Flexible, dynamically adaptive approach
  - Caters to the requirements of multi-modal applications
- Provides “appropriate” fault tolerance for each mode
  - Varies replication style, degree of replication, checkpointing frequency, etc.
- Tailors fault-tolerance properties
  - Uses application knowledge
- Utilizes system resources efficiently

Approach and Contributions

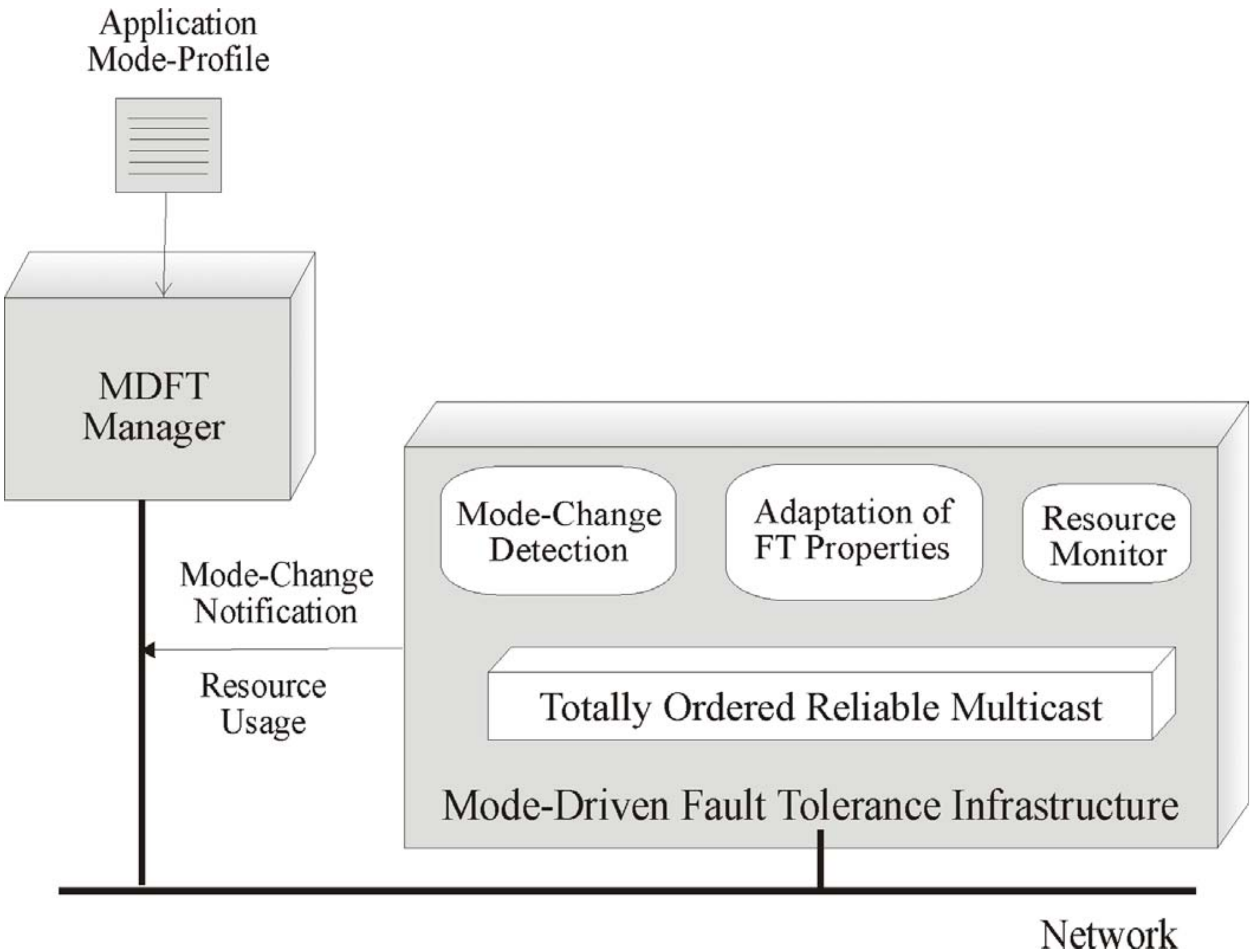
- Specification framework
  - Includes information about application modes relevant from a fault-tolerance viewpoint
- Architecture and infrastructure
  - Uses this information to adapt fault tolerance of application at runtime on a *per-mode* basis
- Key Features
  - *Transparent* to the application
    - The application should not need to be changed or re-coded
  - Can be done at design time or after deployment of application
    - MDFT can be retrofitted onto existing applications

MDFT Specification

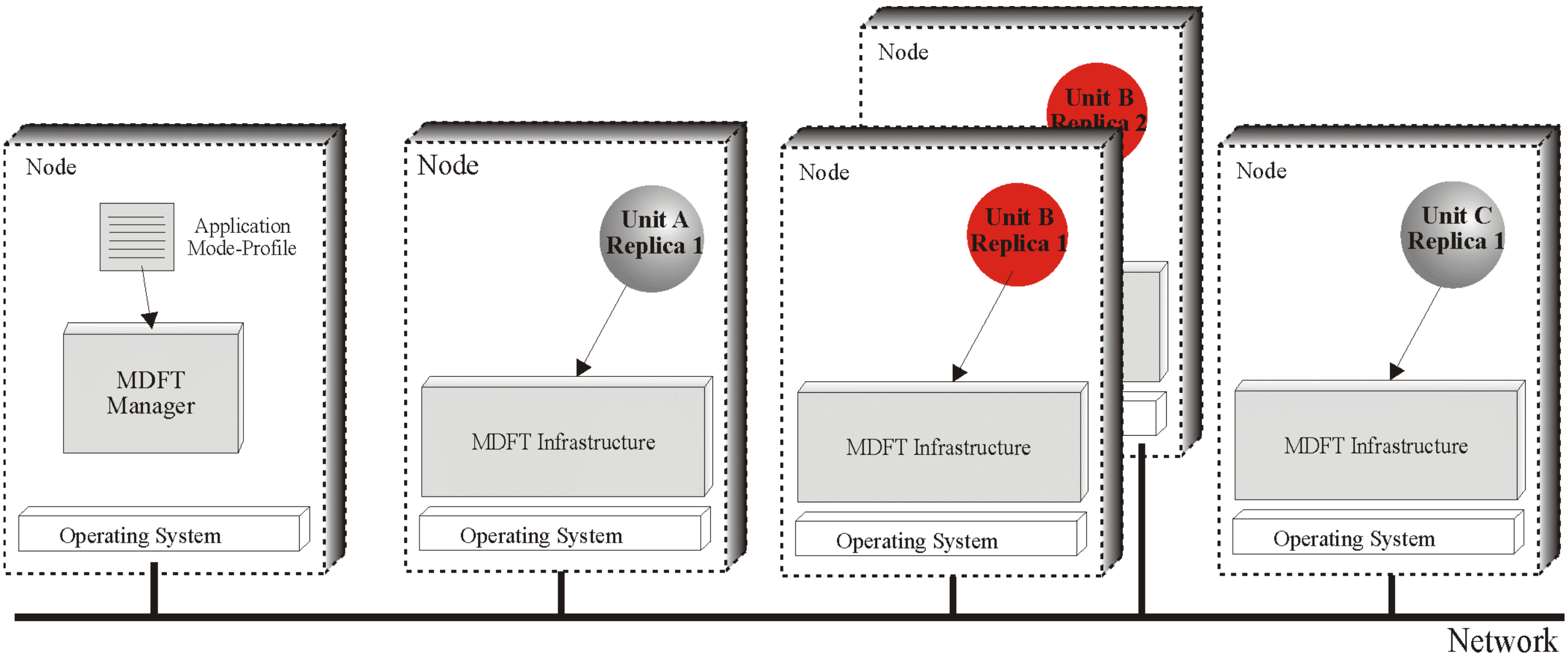
- Models the application to extract/define/identify characteristics relevant for fault tolerance
- Extends the software specification of the application by annotating it with fault-tolerance information
- Examples
  - Mode Transition, Transition Latency, Trigger
  - Unit of execution – task, process, object, component
  - Criticality of each unit



MDFT Infrastructure



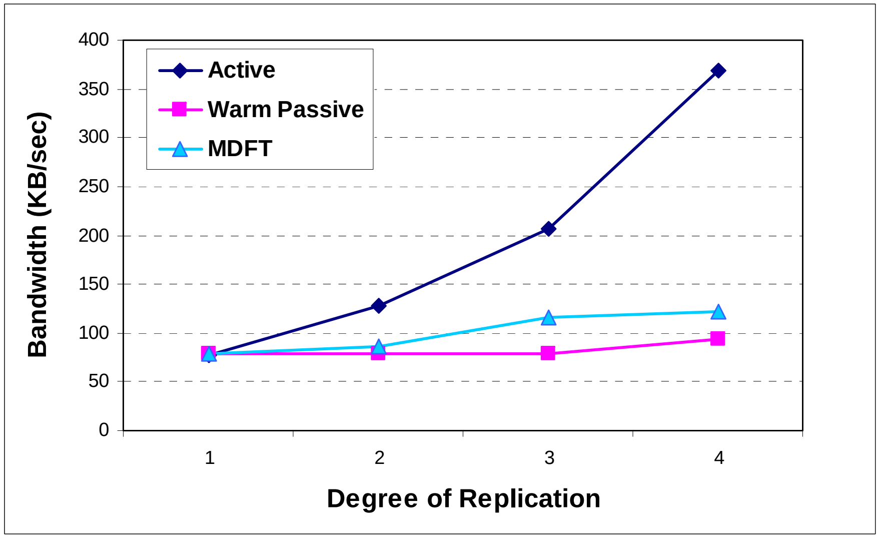
System Architecture



Case Study

- UAV-based test application – 4-tier multi-modal CORBA application
  - Sender, Distributor, Receiver, Mode Detector
  - 2 Modes
    - Target Tracking – Mode 1 (more critical, uses less b/w)
    - Surveillance – Mode 2 (less critical, uses more b/w)
- MDFT – Distributor is most critical component in both modes
  - Point of contact between senders and receivers
- MDFT – Adapt replication style of Distributor based on current mode
- Results
  - Demonstrates feasibility
  - More efficient use of bandwidth using MDFT than with static fault-tolerance
  - Most critical object "protected" in both modes

- Experimental Prototype
  - MDFT-enhanced MEAD (http://www.ece.cmu.edu/~mead)
    - MDFT work extends MEAD with dynamic adaptation capabilities
    - MDFT Manager
  - *Spread* Group Communication
  - TAO ORB – middleware



Conclusion

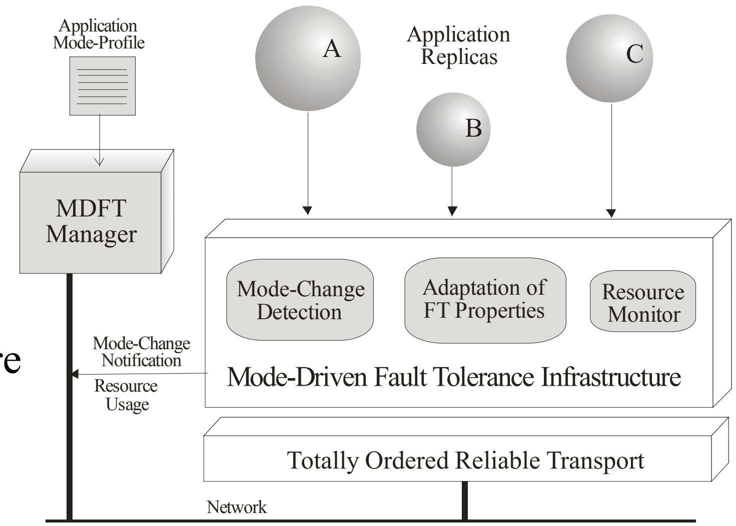
- Identified list of properties required to provision MDFT in distributed applications
  - MDFT Specification
- Created architecture that uses this information to dynamically adapt fault-tolerance properties of application on per-mode basis
  - MDFT Infrastructure
- Explored this in context of distributed middleware-based UAV application
  - Results and insights can be extended to other multi-modal applications
- Future Work
  - Availability of resources to implement MDFT
  - Identify fundamental building blocks of distributed change-management frameworks
  - Can we extend MDFT to handle other distributed changes?
    - Handle live upgrades of running distributed applications?

Questions … Comments …

Deepti Srivastava
Ph.D. Student
Carnegie Mellon University
Pittsburgh, PA 15213-3890
Tel: +1-412-268-5005
dsrivast@ece.cmu.edu

Backup Slides

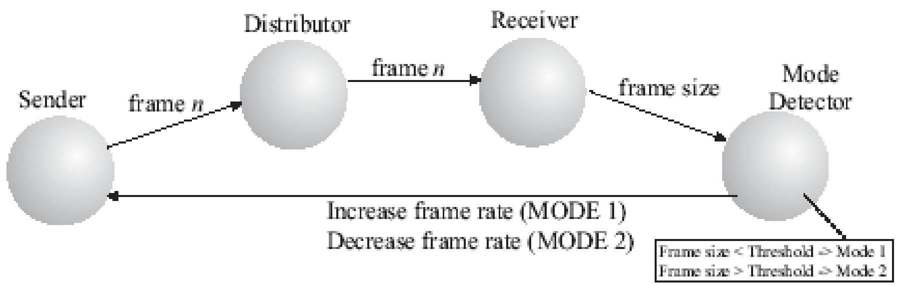
System Architecture (2)



- MDFT Manager
  - "Intelligence" of the MDFT System
  - Decides appropriate FT for each mode
  - User Interface to the MDFT framework; its inputs are
    - Mode profile
    - mode-change notifications
    - resource usage per mode
- Mode-Change Notifier
  - Detects mode change calls in the application at runtime
  - Informs MDFT-Manager of mode changes to trigger MDFT adaptation
- Mode Profile
  - Generated from the mode specification
  - Contains FT-related information for each mode of the application
- MDFT Adaptation Infrastructure
  - Change FT properties dynamically as directed by MDFT Manager
  - Exploit totally-ordered-reliable-multicast for disseminating MDFT changes consistently system-wide

Case Study

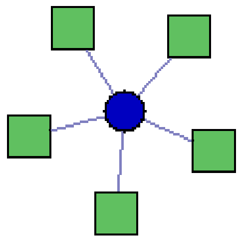
- Test Application
  - UAV: 4-tier multi-modal application
    - Sender – sends 2 types of frames (2 modes)
    - Distributor – point of contact between senders and receivers
    - Receiver – receives frames from distributor(s)
    - Mode Detector – signals mode change in application



- Experimental Prototype
  - MDFT Manager
  - Mode Profile for application
  - MDFT-enhanced MEAD (`http://www.ece.cmu.edu/~mead`)
    - MEAD originally designed to adapt to non-application-specific events
    - MDFT work extends adaptation capabilities to respond to application behaviors such as modes

Empirical Evaluation

- Setup:
  - Run on 5 Emulab nodes (Pentium III, 100 Mbps LAN)
  - Operating System: Linux (Redhat 9)
  - CORBA: TAO ORB
  - *Spread* group communication toolkit (for totally ordered reliable transport)
  - Distributor – replicated



- Results