



Federate Fault Tolerance in HLA-based Simulation

Zengxiang LI **Wentong CAI** Stephen J TURNER Ke PAN
Parallel and Distributed Computing Centre
Nanyang Technological University
Singapore

Outline

- Introduction
 - Why Is Fault Tolerance Needed?
 - Fault Tolerance in HLA-based Simulation
 - Active Replication Based
 - Checkpoint Based
 - Our Objective
- Decoupled Federate Architecture and SOHR
- Federate Fault Tolerance Mechanism
 - Basic State Recovery
 - Optimized State Recovery
- Experiments and Results
- Conclusions and Future Work

Why Is Fault Tolerance Needed?

- Failures in HLA-based simulation
 - Federates are subject to failures (caused by simulation program and/or platform)
 - The risk of a federation failure increases with the federation scale
 - Re-executing failed federation is time and resource consuming
- Crash-stop failures of federates
 - Federation execution is interrupted
 - Time advancement of federate is blocked
 - Simply resigning the crashed federate from the federation is not enough
 - Wrong simulation results

Fault Tolerance in HLA-based Simulation

- High Level Architecture (HLA) standard
 - HLA IEEE 1516 has no formal failure model
 - HLA Evolved provides fault tolerance interfaces
 - Data recovery protocols are absent
- Active Replication based mechanisms
 - Using a FT manager to manage federate replicas [Berchtold and Hezel 2001]
 - Allowing replicas to employ different synchronization approaches [Li et al 2010]
- Checkpoint based mechanisms
 - Global consistent checkpoint of the entire federation [e.g., Cucuzzo et al 2007]
 - Local uncoordinated checkpoint of each federate [e.g., Eklöf et al 2005]
 - All federates in the federation are involved to recover a single crashed federate

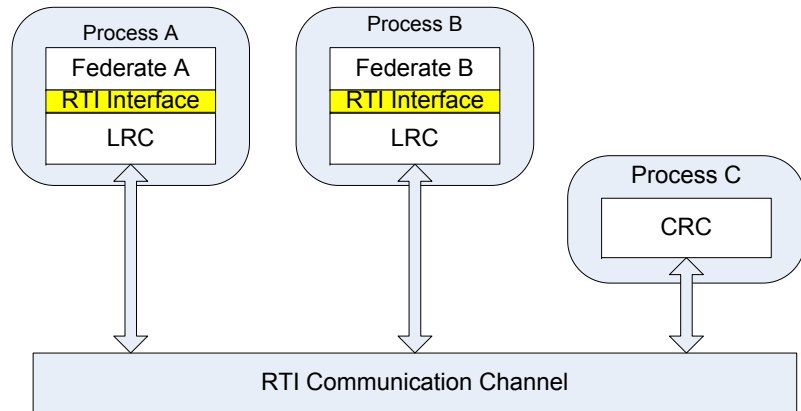
Our Objectives → A fault tolerance mechanism is proposed by exploiting the advantages of the decoupled federate architecture

- Rollback recovery protocol using checkpointing and message logging
- Transparent & Efficient

Outline

- Introduction
 - Why Is Fault Tolerance Needed?
 - Fault Tolerance in HLA-based Simulation
 - Active Replication Based
 - Checkpoint Based
 - Our Objective
- Decoupled Federate Architecture and SOHR
- Federate Fault Tolerance Mechanism
 - Basic State Recovery
 - Optimized State Recovery
- Experiments and Results
- Conclusions and Future Work

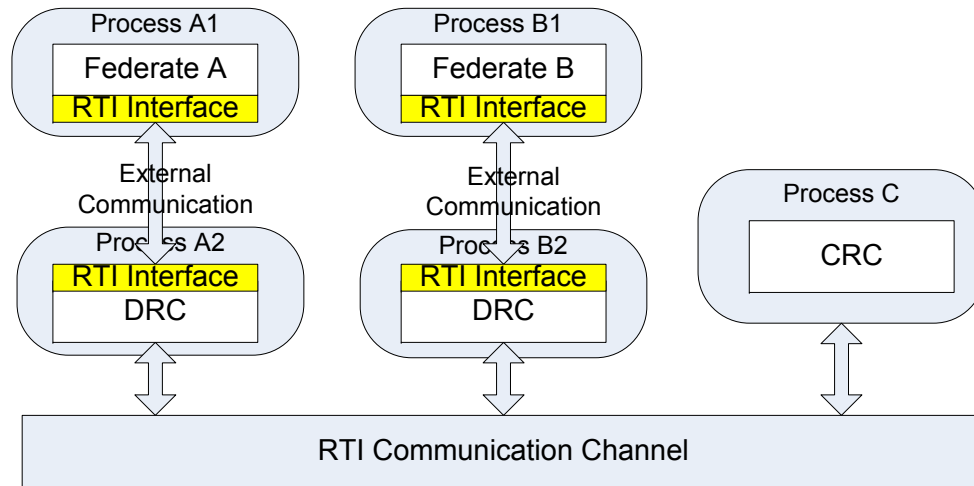
Traditional Federate Architecture



- ✓ LRC, a library, connects a federate to the federation
- ✓ A federate is compiled and linked with its LRC to form an execution process

- Federate and LRC become unavailable if either of them crashes
 - Federate state and LRC state are lost in the failure
- The crashed federate is lost from the federation
 - Exchanged messages are lost in the failure

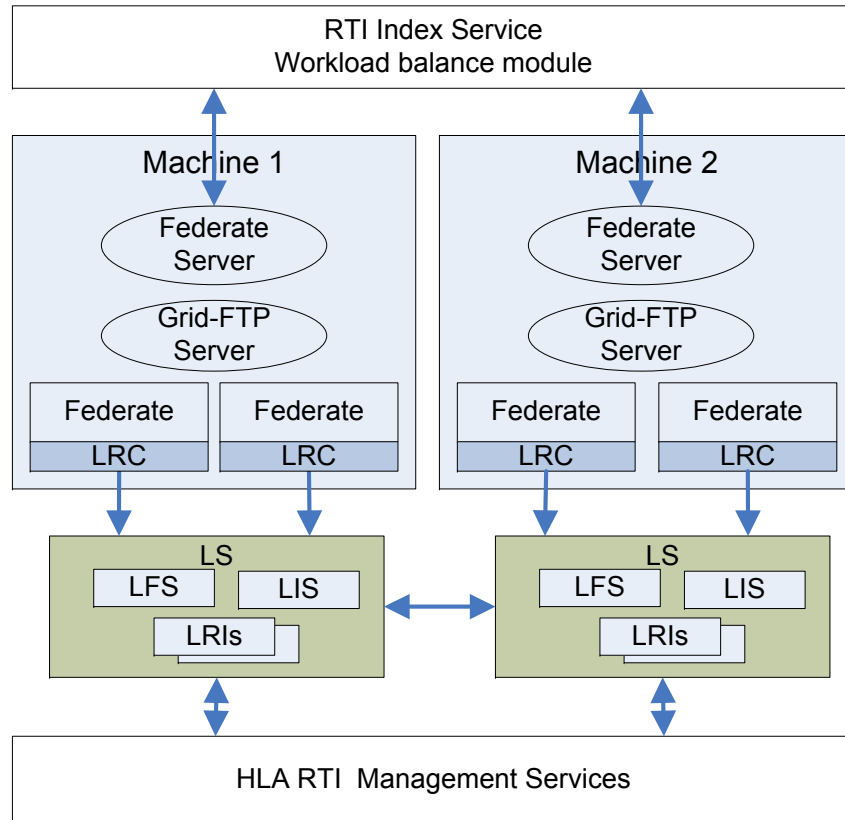
Decoupled Federate Architecture



- ✓ DRC plays the same role as LRC in the traditional federate architecture
- ✓ The federate and its DRC are executed in different processes

- **Failure isolation:** DRC survives from federate failures
- **Failure detection:** DRC monitors the status of the federate
- **Failure recovery:** DRC logs checkpoints and messages
 - DRC keeps connection to the federation for the federate
 - Failure recovery procedure is transparent to other federates

Overview of SOHR

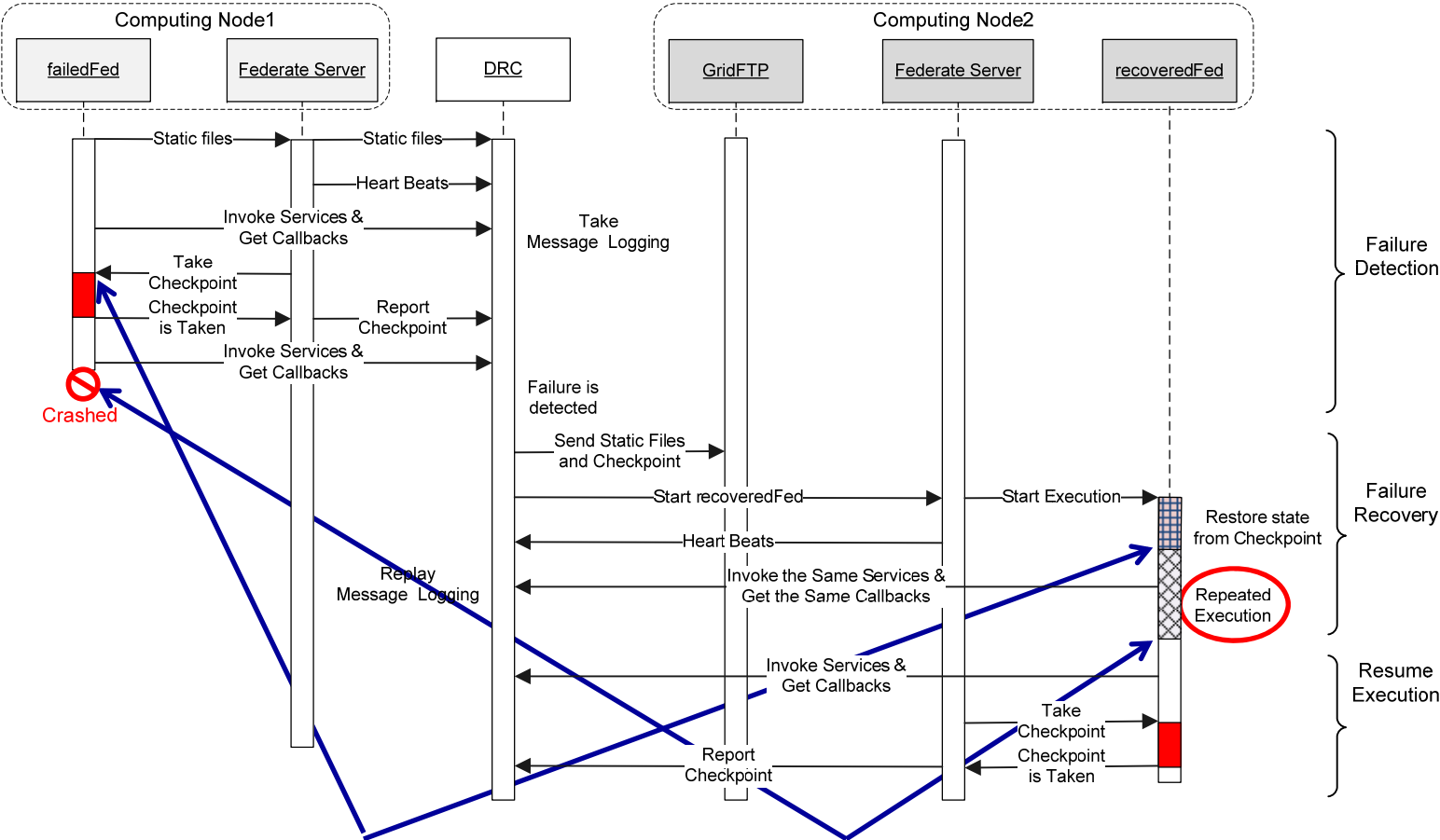


- **RTI Index service**
 - Provides system-level registry
- **HLA RTI Management services**
 - Provides HLA service groups
- **LS (local service):**
 - LFS creates LRI for each federate
 - LRI saves concerned RTI state
 - LIS provides Grid service RTI interface
- **LRC (local RTI component):**
 - Translates to Grid Service RTI interface
- **Federate server:**
 - Manages federates in the machine
- **Grid-FTP server:**
 - Enables file transfer
- **Workload balance module:**
 - Maps federates to available machines

Outline

- Introduction
 - Why Is Fault Tolerance Needed?
 - Fault Tolerance in HLA-based Simulation
 - Active Replication Based
 - Checkpoint Based
 - Our Objective
- Decoupled Federate Architecture and SOHR
- Federate Fault Tolerance Mechanism
 - Basic State Recovery
 - Optimized State Recovery
- Experiments and Results
- Conclusions and Future Work

Fault Tolerance Mechanism

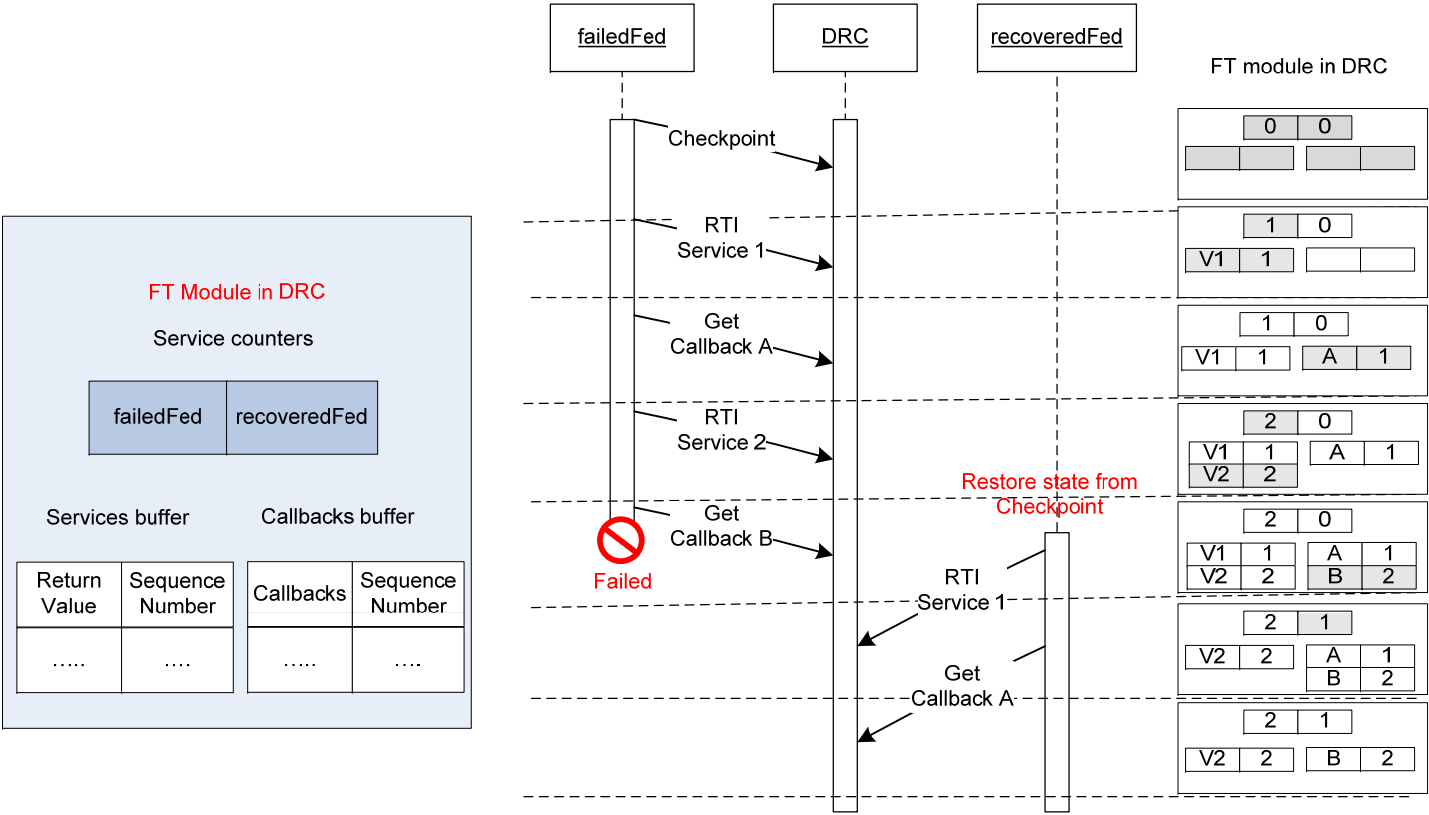


Checkpoint time ≤ Failure time

Basic State Recovery

- Federate checkpoint
 - Entire federate state
- Repeated execution
 - Federates are piece-wise deterministic (→ repeatable execution)
 - A sequence number-based method is used
 - DRC buffers the callbacks and delivers the same callbacks in the same sequence to the recovered federate
 - Recovered federate repeats the execution conducted by failed federate identically
 - Processing the same events in the same sequence
 - Invoking the same RTI services following the same sequence
 - DRC does not handle the duplicate RTI services
 - Repeating execution is finished when the sequence number of recovered federate is equal to that of failed federate
 - Recovered federate has the same state as the failed federate just before the failure

Basic State Recovery



(a) Data structures of FT module in DRC

(b) One example of basic mechanism

Optimized State Recovery

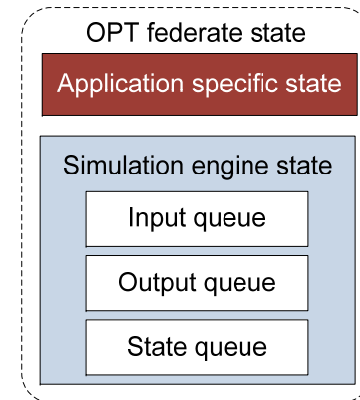
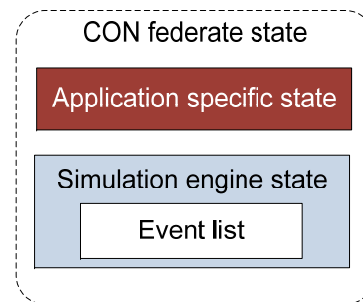
- Federate checkpoint

- CON federate: entire federate state
- OPT federate:

- Application specific state
 - Generally unsafe
- Input queue: Events
 - Conservative **VS** Optimistic
- Output queue: Outgoing messages
 - Correct **VS** Incorrect
- State queue: snapshots of application specific state
 - Safe **VS** Unsafe
- **OPT federate checkpoint includes**
 - Safe snapshot of application specific state
 - » The latest snapshot in state queue with $TS < \text{checkpoint time}$
 - Conservative events
 - » Conservative events in input queue with $TS \geq \text{checkpoint time}$

Event that will not be cancelled or retracted

Event that can be re-generated during repeat execution or buffered in DRC



Optimized State Recovery

- Repeated execution
 - CON Federate
 - Advances to failure time directly
 - Gets callbacks with TS between checkpoint time and failure time
 - Processes events between checkpoint time and failure time in TS order
 - Messages generated during the period are filtered out
 - OPT Federate
 - Advances to failure time directly
 - Gets **correct** callbacks with TS between checkpoint time and failure time
 - **Incorrect** callbacks must have been annihilated with the retractions
 - Processes events between checkpoint time and failure time in TS order
 - No rollbacks
 - Messages generated during the period are filtered out
 - Handling messages generated after failure time
 - Retract if generated by the failed federate but not recovered federate
 - Discard if generated by both the failed and recovered federates
 - Send if generated by only by the recovered federate

Comparison Between Basic and Optimized State Recovery Protocols

Advantages of optimized protocol	Failed Federate	
	Conservative	Optimistic
Advance to failure time directly	✓	✓
Receive buffered messages at one time	✓	✓
Avoid sending messages during repeated execution	✓	✓
Reduce overhead of taking checkpoint	N/A	✓
Avoid receiving incorrect messages with $TS < \text{failure time}$ and their retractions	N/A	✓
Receive messages with $TS < \text{failure time}$ in TS order	N/A	✓
Process events with $TS < \text{failure time}$ without any rollbacks	N/A	✓

Outline

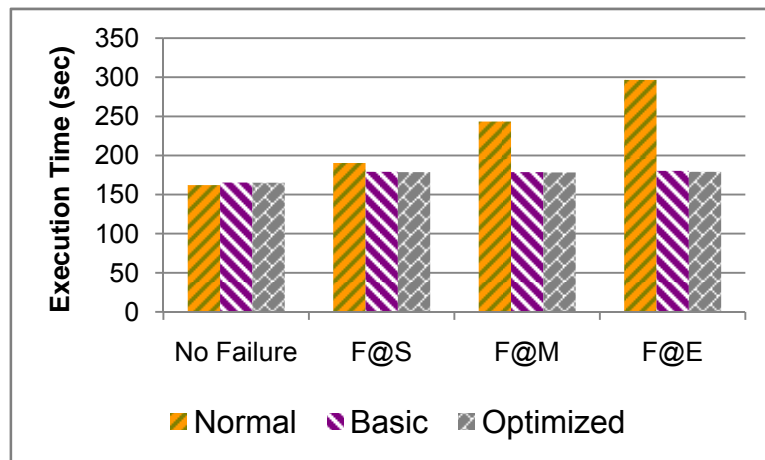
- Introduction
 - Why Is Fault Tolerance Needed?
 - Fault Tolerance in HLA-based Simulation
 - Active Replication Based
 - Checkpoint Based
 - Our Objective
- Decoupled Federate Architecture and SOHR
- Federate Fault Tolerance Mechanism
 - Basic State Recovery
 - Optimized State Recovery
- Experiments and Results
- Conclusions and Future Work

Experiments and Results

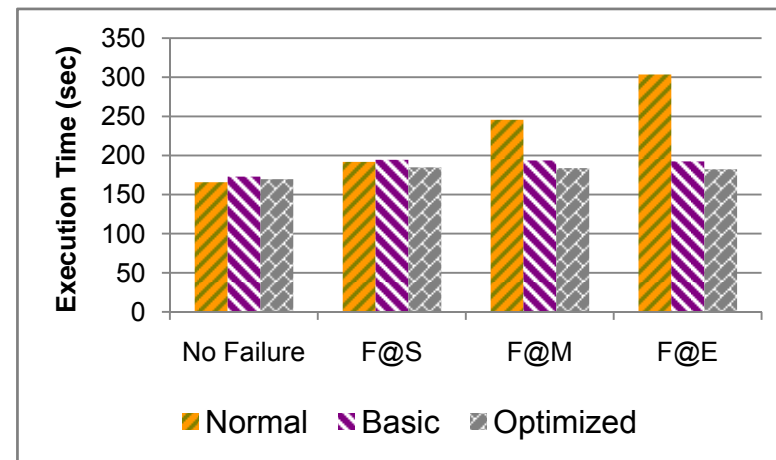
- P-hold simulation
 - Two symmetric federates in the federation
 - Federates may send an external event to the other federate while processing a local event (the probability of generating an external event is denoted as $P_{\text{ExternEvent}}$)
 - Spin-loops are added to emulate event processing and the cost of state saving and restoration in OPT
 - All machines used have the same workload
- Execution scenarios
 - Normal scenario: SOHR without a fault tolerance mechanism
 - Basic scenario: SOHR with the fault tolerance mechanism using **basic** protocol
 - Optimized scenario: SOHR with the fault tolerance mechanism using **optimized** protocol
- CON and OPT federation executions: federates in the federation employ conservative and optimistic synchronization respectively
- Length of repeated execution = failure time – checkpoint time

Experiments and Results

- Benefit of the proposed fault tolerance mechanism



(a) CON federation

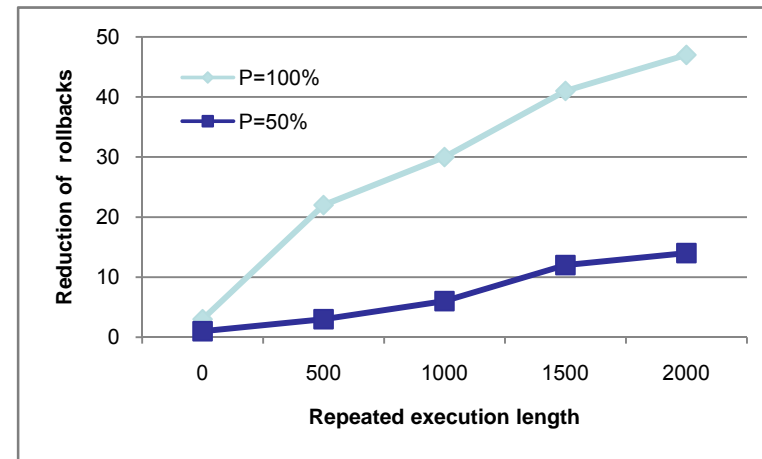
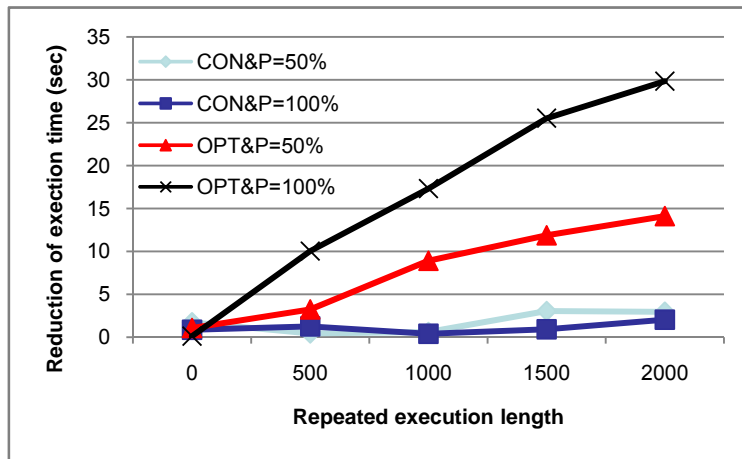


(b) OPT federation

- $P_{\text{ExternEvent}} = 50\%$; Length of repeated execution = 1000 simulation unit
- Acceptable overhead is introduced in Basic and Optimized scenarios
- Failures are not tolerated in Normal scenarios
 - The later the failure happens, the more time is wasted on the failed federation execution
- Failures are tolerated in Basic and Optimized scenarios
 - Performance degrades slightly in the presence of failures

Experiments and Results

- Advantages of the optimized state recovery protocol over the basic one



- CON federation: Reduction is minimal
- OPT federation: Reduction is considerable
 - Increases linearly respect to the length of repeated execution
 - Increases more significantly when $P_{\text{ExternEvent}}$ is larger
- Increases linearly respect to the length of repeated execution
- Increases more significantly when $P_{\text{ExternEvent}}$ is larger

Outline

- Introduction
 - Why Is Fault Tolerance Needed?
 - Fault Tolerance in HLA-based Simulation
 - Active Replication Based
 - Checkpoint Based
 - Our Objective
- Decoupled Federate Architecture and SOHR
- Federate Fault Tolerance Mechanism
 - Basic State Recovery
 - Optimized State Recovery
- Experiments and Results
- Conclusions and Future Work

Conclusions and Future Work

- Decoupled Federate Architecture
 - Benefits for federate fault tolerance
- Federate Fault Tolerance Mechanism
 - Heartbeat-based failure detection
 - Checkpointing and message logging
 - Basic State Recovery **VS** Optimized State Recovery
- Future Work
 - DRC crashes
 - Federate and DRC crash
 - Byzantine failures



Thanks for your attention!

Questions & Answers

