Data Consistency with SPLICE Middleware

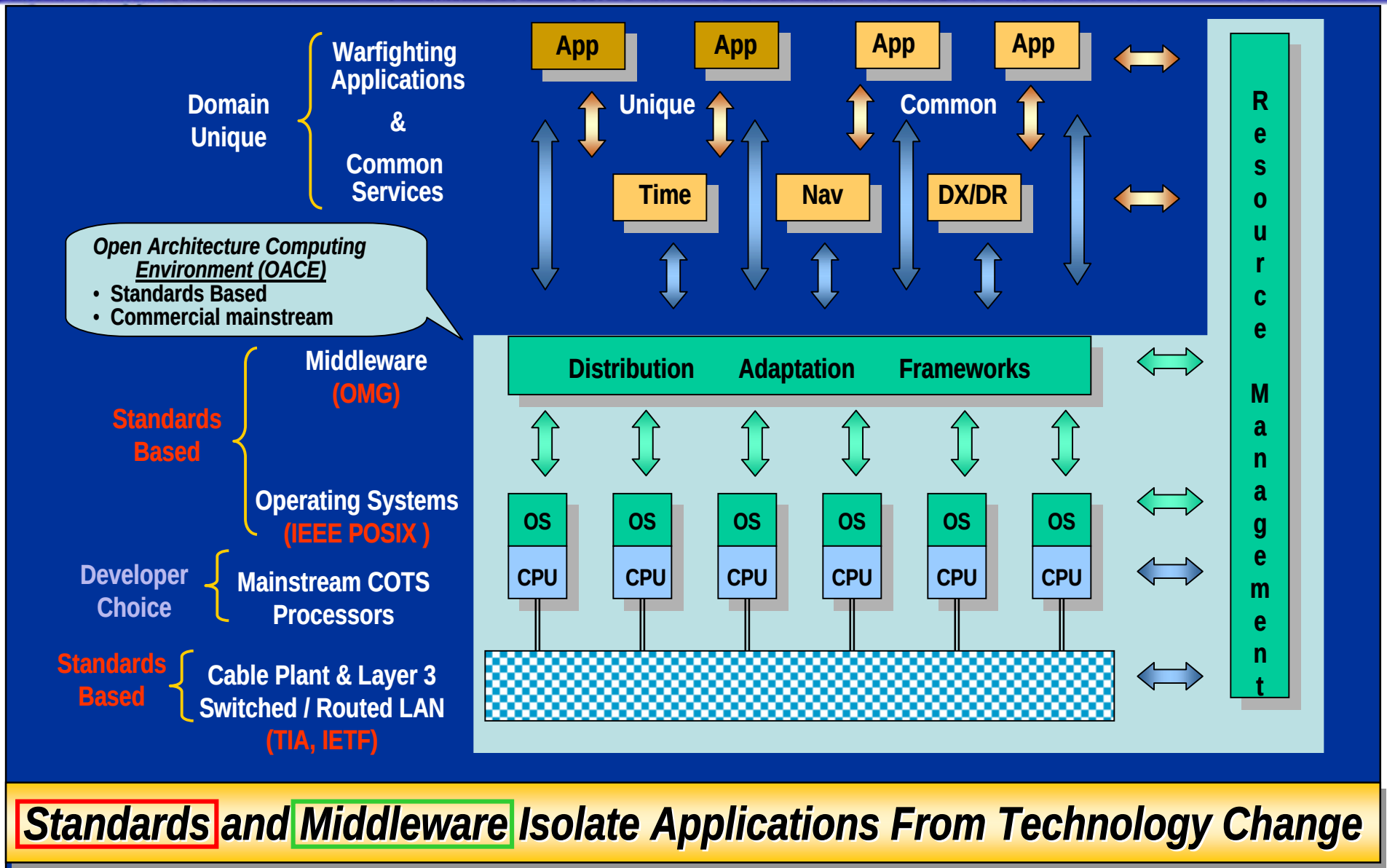
Leslie Madden
Chad Offenbacker

Naval Surface Warfare Center Dahlgren Division

Background

- US Navy Combat Systems are evolving to distributed systems and open standards
  - Leverage commodity technology
  - Enable technology refresh while reducing cost
  - More quickly add new warfighting capability and capacity to meet threats
- Combat systems need to distribute data across applications while maintaining a degree of data consistency sufficient to
  - Meet fault tolerance requirements
  - Meet failure recovery requirements
  - Support correct execution of distributed, real-time applications
- Historically, US Navy Programs of Record (PORs) have developed non-standards-based, custom solutions for maintaining distributed state data.
  - Full visibility into solution allows developers a full understanding of the consistency, fault tolerance, and failure recovery characteristics.
- Various US Navy and DoD thrusts encourage the use of standards-based solutions and COTS or open source products where feasible
  - DDS Persistence Profile potentially provides a standards-based solution to data consistency across distributed applications

US Navy Surface Domain OA Computing Environment (OACE)



Standards and Middleware Isolate Applications From Technology Change

DDS and Data Consistency

- **The DDS specification provides data *durability* QoS to control the middleware retention of distributed data for retransmission in the case of newly initiated data readers.**
  - **Volatile – Service does not retain any data on behalf of the data writer. Only existing data readers will receive data.**
  - **Transient-Local – Service will transmit data to new subscribers only if the original data writer is alive. ***
  - **Transient – Service will retain and transmit data to new subscribers, even if the original data writer is no longer alive, as long as the service has not terminated. ***
  - **Persistent – Data is kept on permanent media so it may outlive service.**
- **Transient durability, combined with reliable delivery and careful use of key fields appears to provide a standards-based mechanism for maintaining data consistency under application failure and restart conditions.**

*** In-memory maintenance of data based on *history* and *resource-limits* QoS**

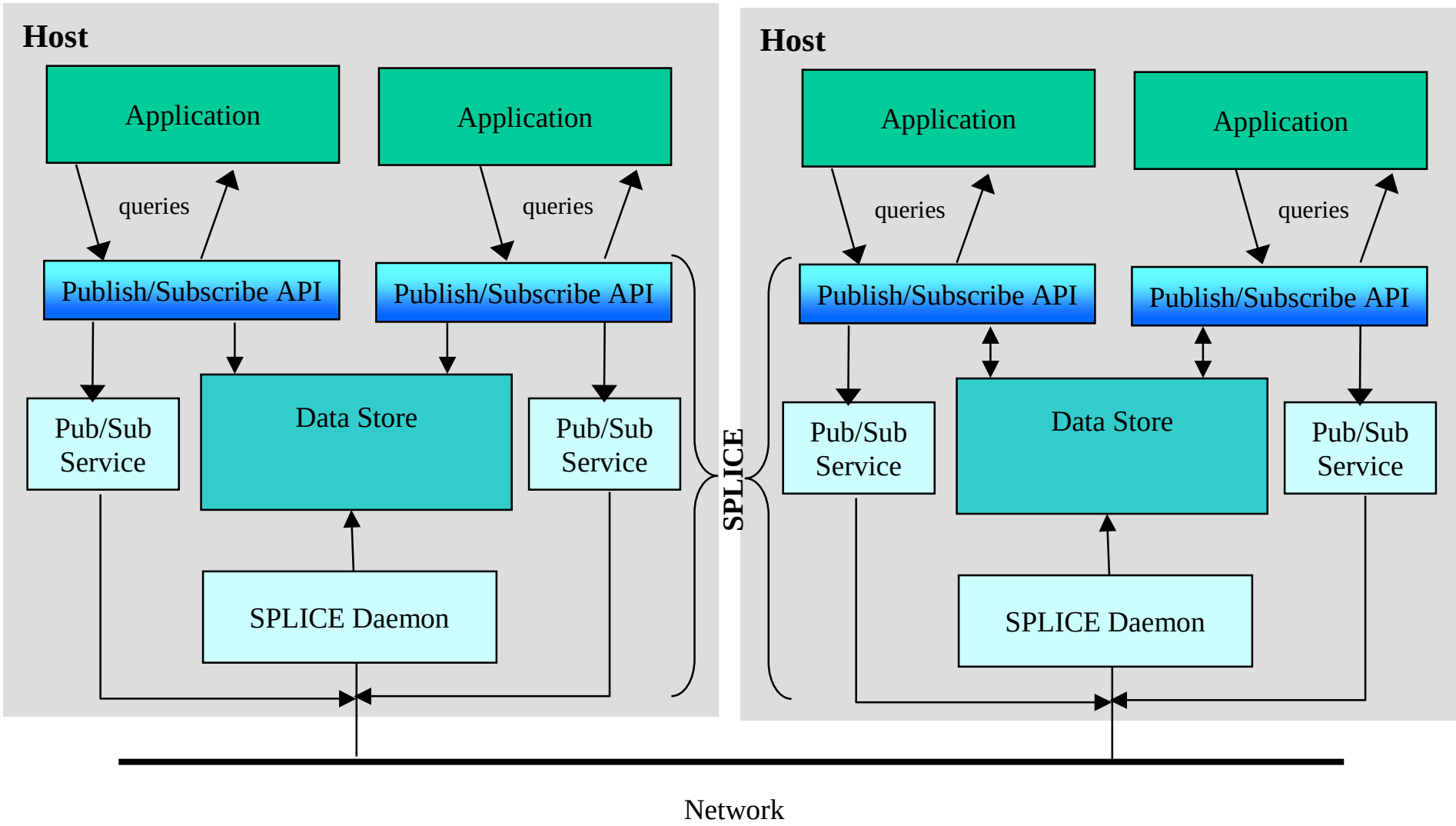
Benchmark Objectives

- Develop a mechanism to empirically evaluate “how consistent” distributed data is maintained by DDS implementations
  - Source code analysis may not be feasible. Commercial product vendors may not be willing to provide source code or may provide it at a high cost. Cost of analysis may be prohibitive.
  - Understanding the circumstances under which data consistency is & is not maintained will allow engineering of a system that meets fault tolerance and correctness requirements.
- Gain an understanding of the DDS Persistency Profile and how it might support US Navy Combat System data consistency needs
  - Need to understand differences between the DDS specification and the SPLICE implementation.
  - Characterize the performance & behavior of the SPLICE implementation.

Evaluation Goals

- Assess behavioral characteristics critical to highly reliable systems
  - Fault tolerance
  - Data consistency
- Determine performance & resource utilization characteristics - is it sufficient for all or some subset of Surface Navy Combat System applications
  - Latency
  - Performance scalability
  - CPU & memory
- Assess appropriate architecture patterns and usage strategies based on characteristics - significant differences between SPLICE and other publish/subscribe middleware products
  - SPLICE provides in memory data store that is refreshed by published data
  - SPLICE provides query-like mechanism for accessing data and “joining” stored data items – supported by the DDS *Content-Subscription* profile, but not currently implemented by other products

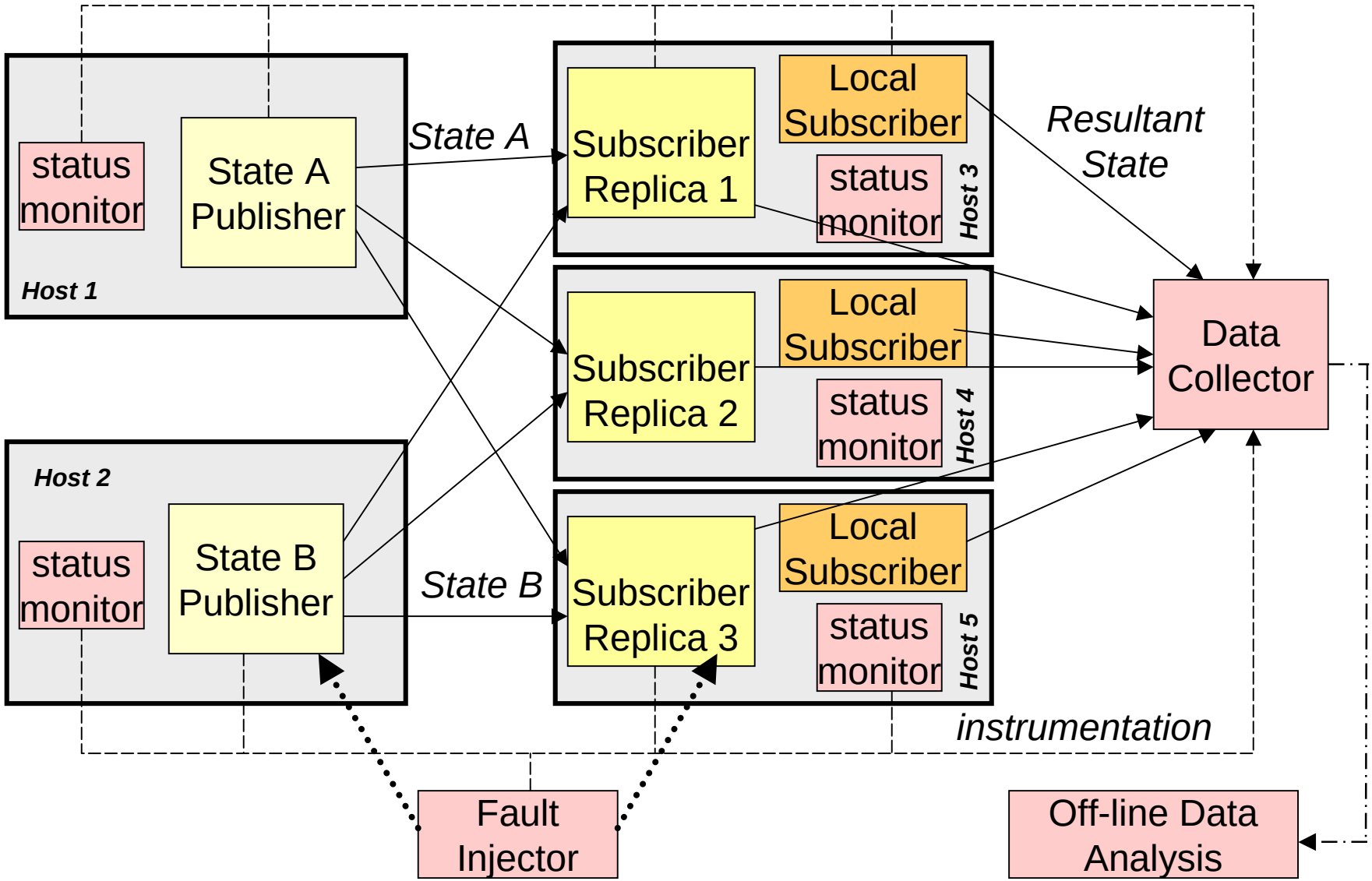
SPLICE Architecture



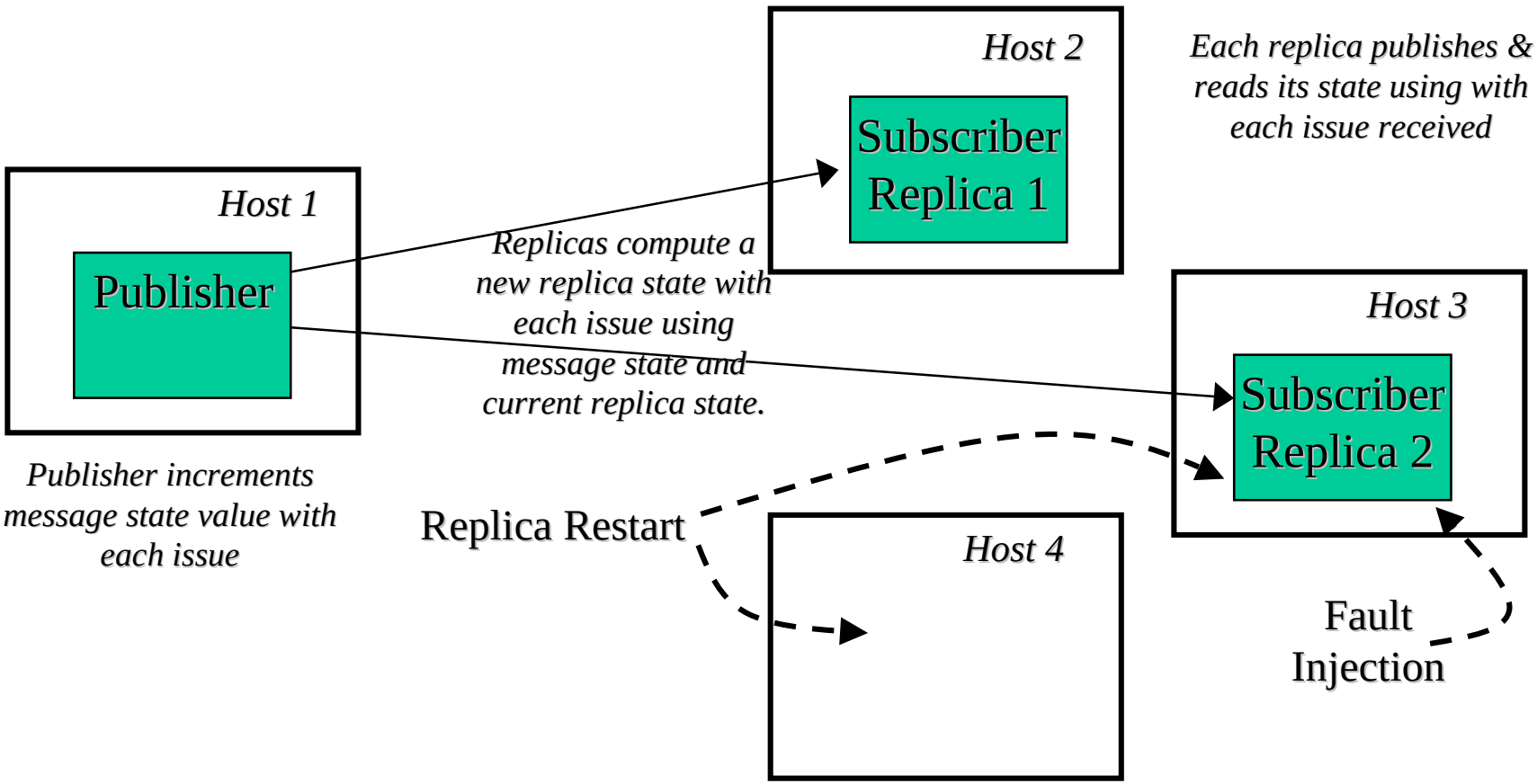
Challenges

- A test harness is required to capture externally observable behavior. Resulting data must be analyzable to draw conclusions about the degree of data consistency provided by the middleware.
- Test suite must be configurable to simulate a variety of system characteristics e.g.:
  - Different numbers of data producers and consumers.
  - Different configurations of replicated data producers and consumers.
  - Differences in data processing (algorithms) that may affect consistency in replicated consumers e.g. storing a value vs maintaining a sum of values received.
  - Ability to inject different types of faults into data producers and consumers.
- The health and status of each test application must be monitored continuously so the time fault occur can be accurately determined.
- Every change to state data for each data producer and consumer must be monitored on a continuous basis.
- Given that data inconsistency may occur infrequently, a mechanism is required to analyze large quantities of data from long periods of test execution.

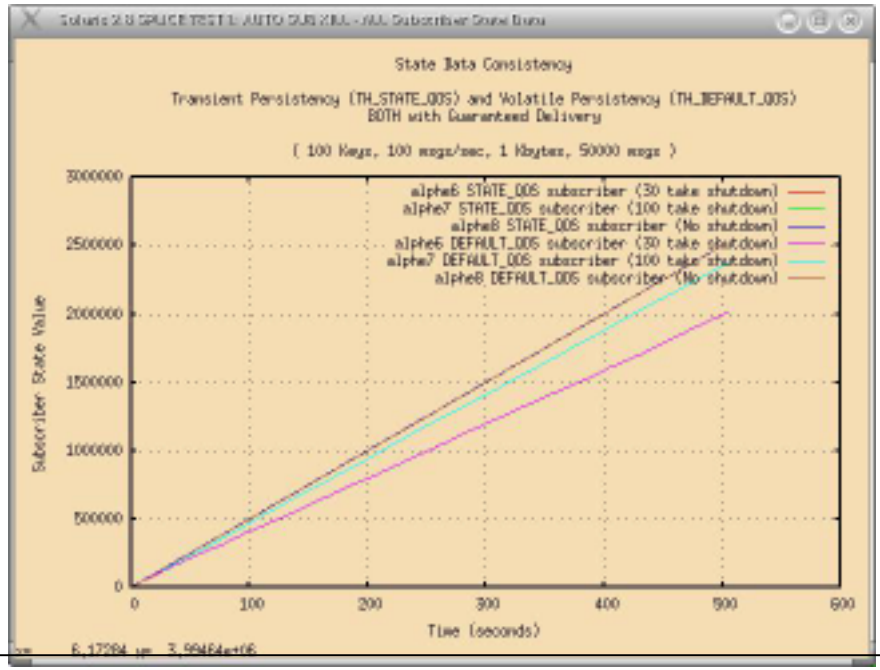
Fault Tolerance Experiment



Test Configuration



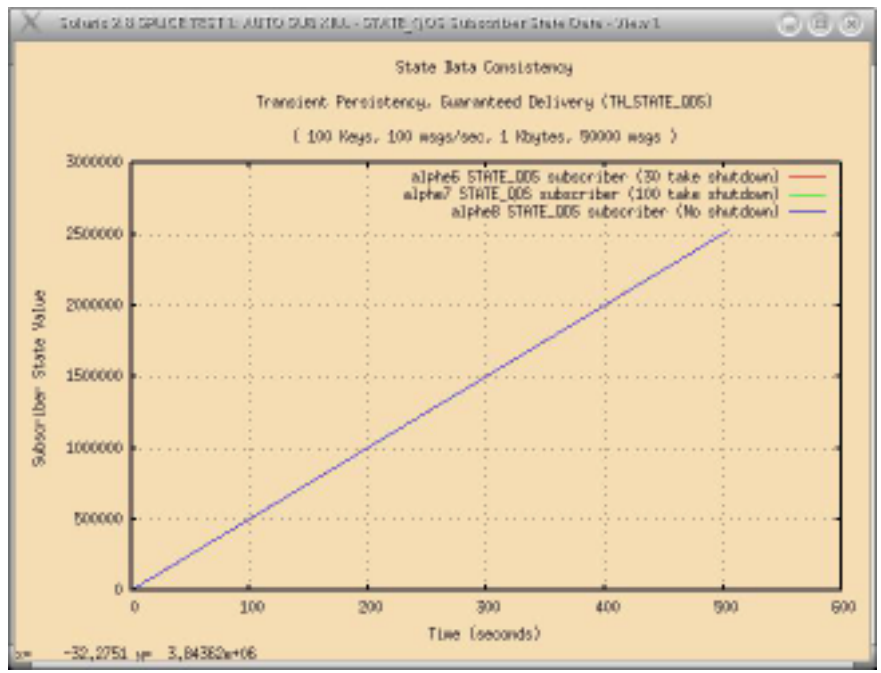
- **Default QoS:** Durability **= *Volatile*;** Reliability **= *Guaranteed***
- **State QoS:** Durability **= *Transient*;** Reliability **= *Guaranteed***

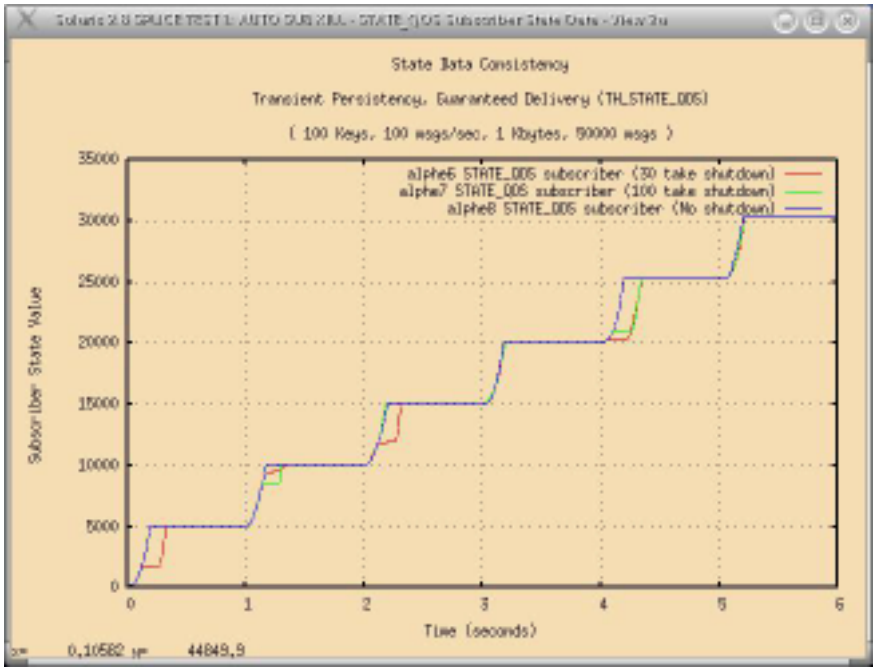


SPLICE Data consistency during periodic shutdown and restart of same node subscriber processes.

- **State value data shown for both STATE & DEFAULT QOS subscribers across all nodes.**
- **Both STATE & DEFAULT QOS subscriber processes execute concurrently on each node.**
- **Subscriber processes on each node shutdown and restart at different rates, or not at all.**

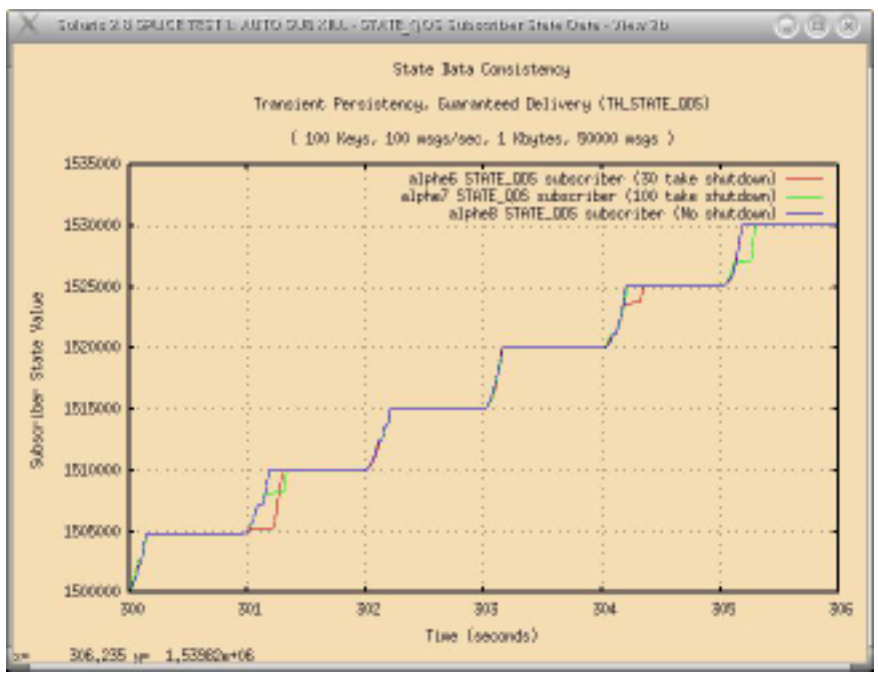
Despite frequent shutdowns and restarts, the STATE_QOS subscriber processes appear to maintain a close state value consistency across separate nodes over time.

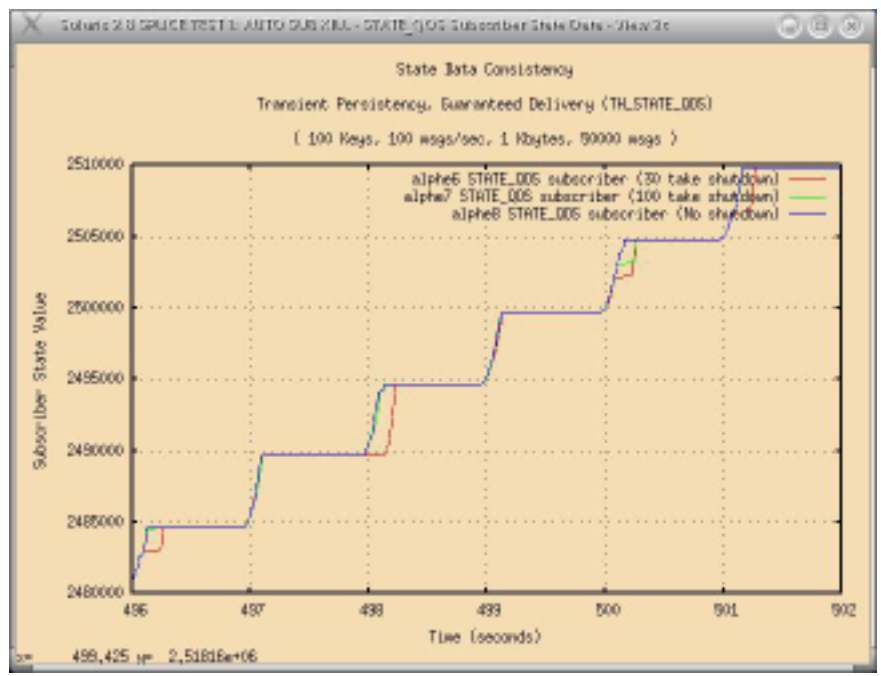


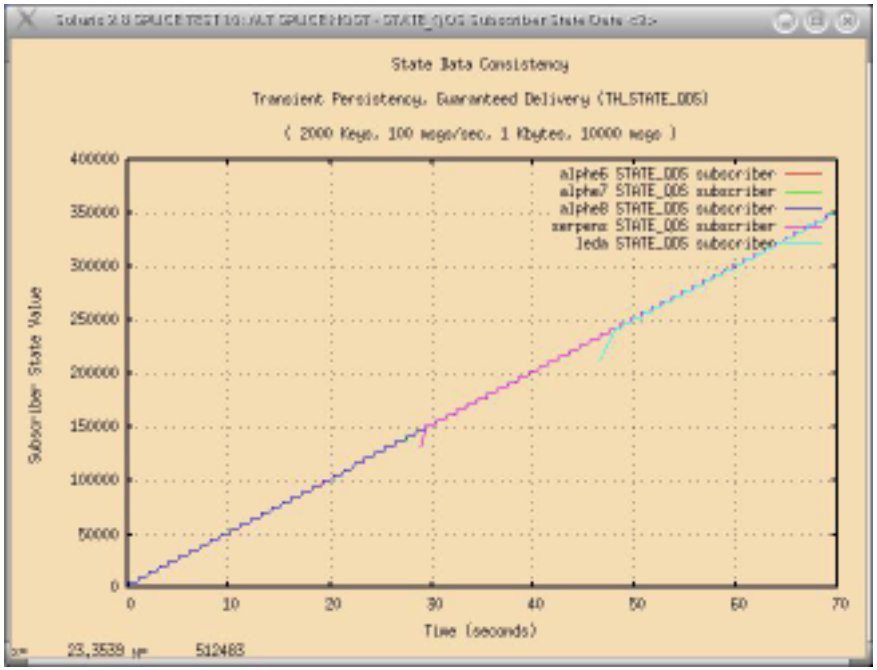


SPLICE Data consistency with periodic shutdown and restart of same-node subscriber processes.

- **Counter-clockwise inspection of graphs reveal a much closer view of state value consistency at the beginning, middle, and end of the aforementioned test run.**
- **The rate of change of each STATE_QOS subscriber state value REMAINS CONSISTENT across the SPLICE nodes.**

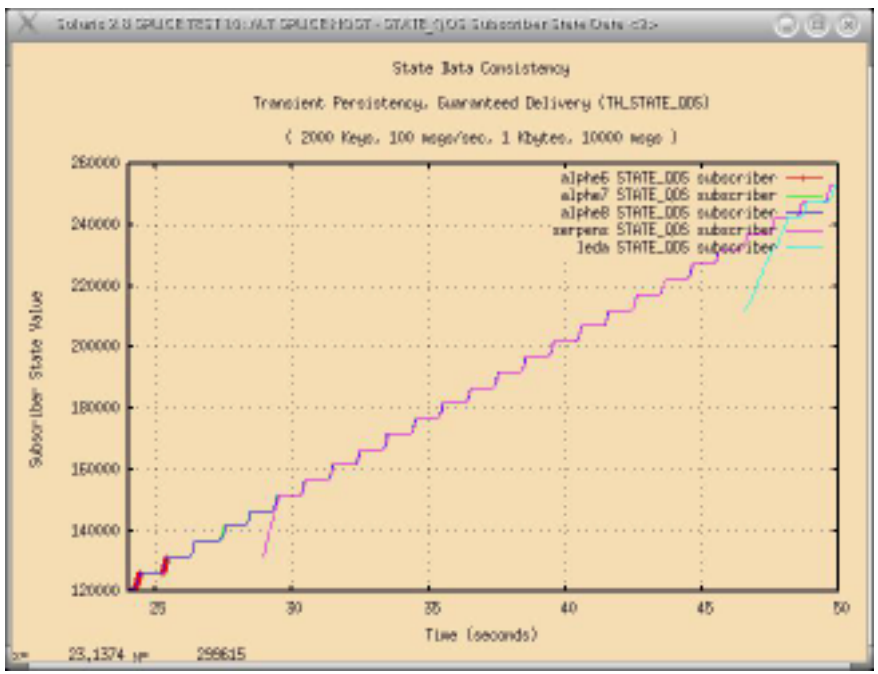


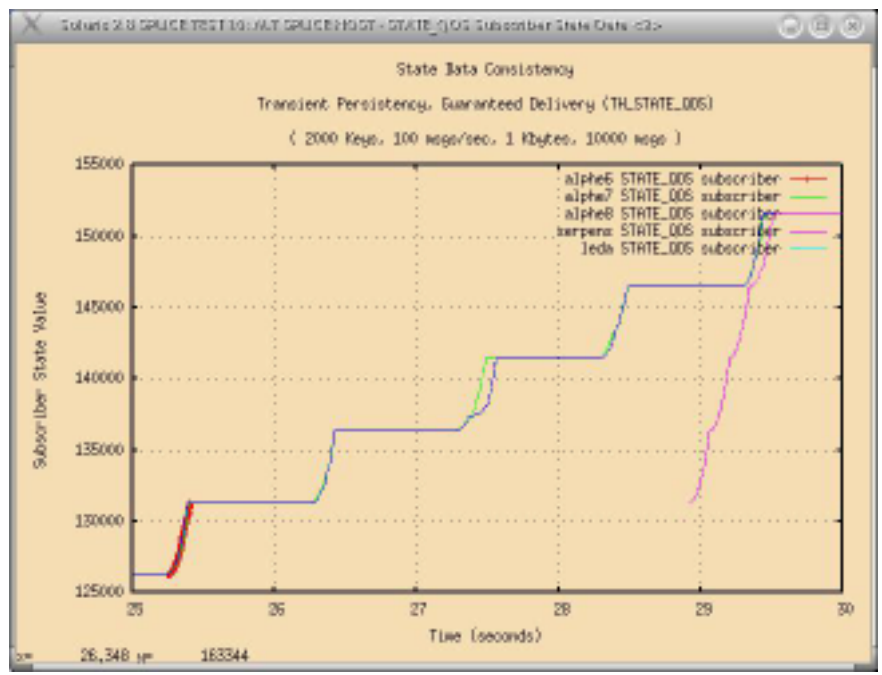


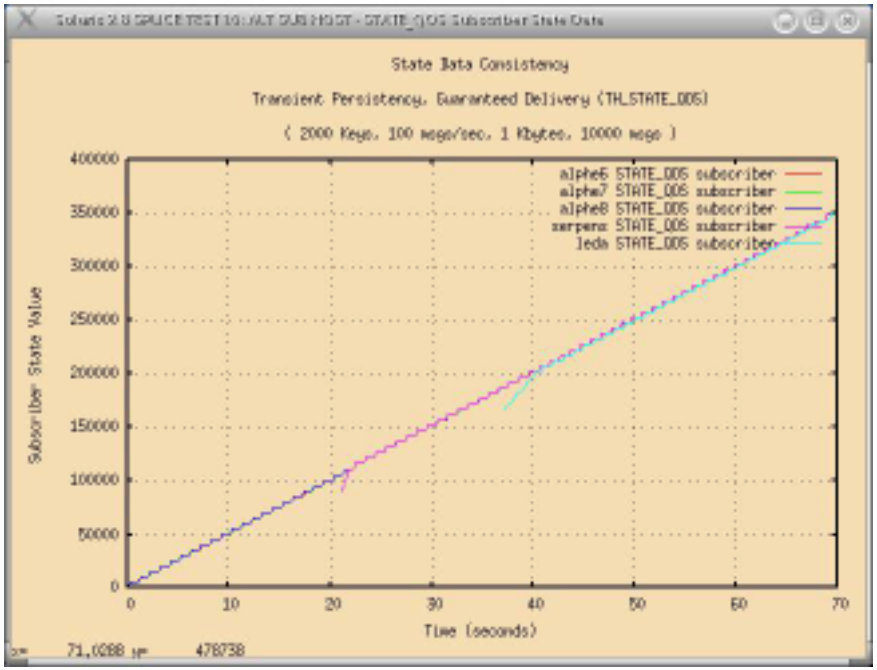


SPLICE Data consistency during shutdown of a subscriber node and start of its alternate node replica.

- **Test required a COLD start of a SPLICE daemon on a replica node, before initiating a replica subscriber process (*daemon started AFTER node shutdown*).**
- **Human factor and architecture contributed to delayed subscriber replica start-up.**
- **Lower right graph shows clearly the ability of the STATE_QOS subscriber replica to quickly reach its appropriate state value upon start-up.**

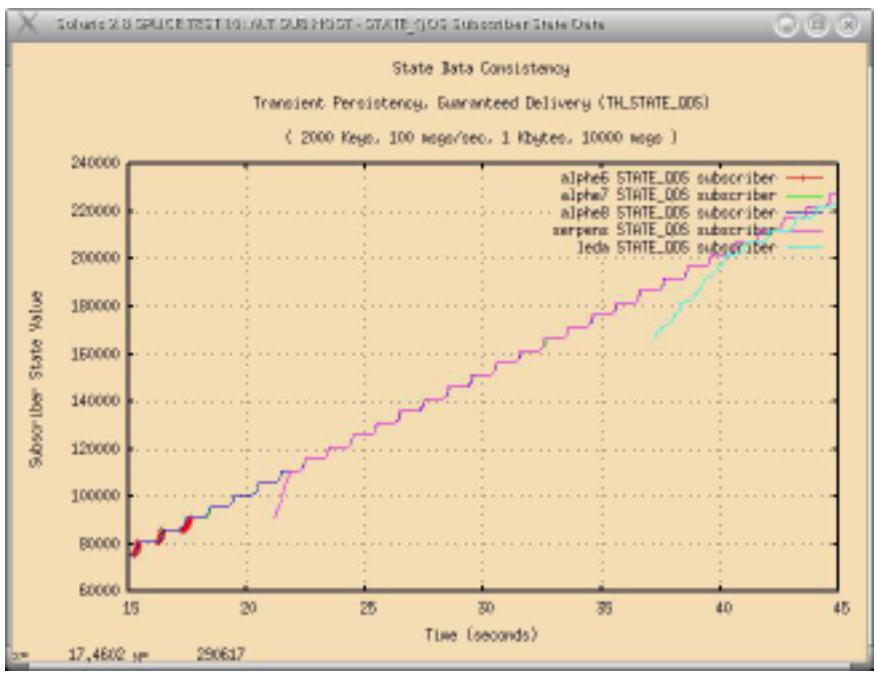


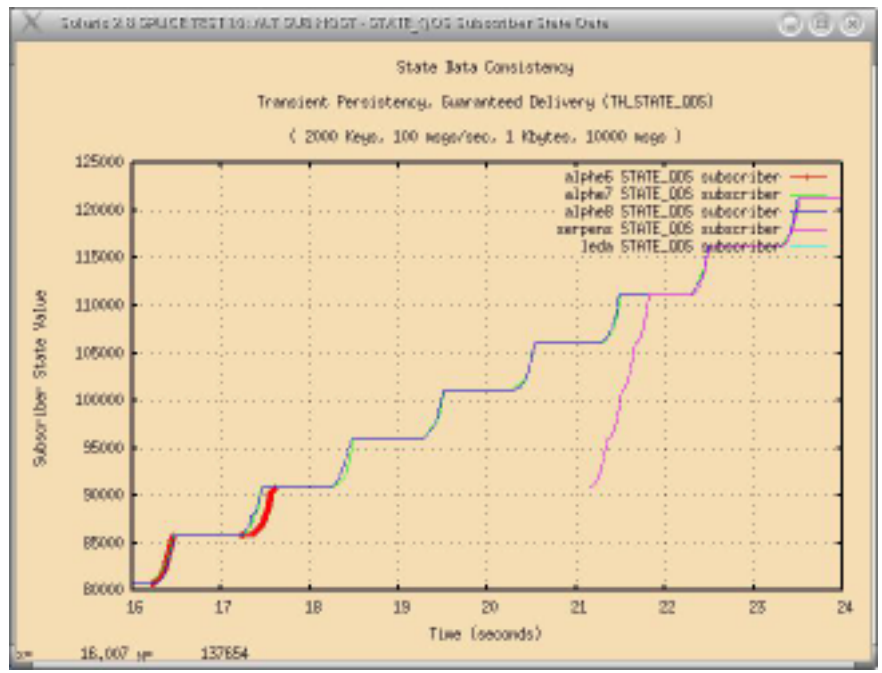




SPLICE Data consistency during shutdown of a subscriber node and start of its alternate node replica.

- **Test required a WARM start of a SPLICE daemon on a replica node, before initiating a replica subscriber process (*daemon started BEFORE node shutdown*).**
- **Human factor and architecture contributed to delayed replica start-up time.**
- **Lower right graph shows clearly the ability of the STATE_QOS subscriber replica to quickly reach its appropriate state value upon start-up.**





Preliminary Results

- **State QoS - Transient & Guaranteed**
  - SPLICE maintains consistency of state through multiple subscriber replica process faults and restarts, even when a replica is restarted on a previously unused host.
  - Issues transmitted prior to subscriber replica restart may be lost if subsequent issue with the same key field is transmitted in the interim. Otherwise, all issues will be received.
- **Default QoS - Volatile & Guaranteed**
  - SPLICE maintains consistency of state through multiple subscriber replica process faults & restarts when a replica is immediately restarted on the same host, but not when it is started on a previously unused host.
  - Issues transmitted prior to subscriber replica restart may not be received by restarted replica.

SPLICE Data Persistence Behaves "As Advertised"

Next Steps

- Collect and analyze data from longer duration executions
- Collect and analyze data from different types of faults (e.g. host faults, network faults)
- Analyze performance data for ability to meet combat system fail-over requirements
- Assess other DDS implementations of persistence profile as they become available
- Develop prototype applications to assess applicability of technology in a multi-middleware technology, heterogeneous, system-scale environment