**ABSTRACT**

Today’s mission-critical safety-certified systems often require fault tolerance implemented with redundant components. Data Distribution Service (DDS™) provides the building blocks for redundancy. This paper presents an overview of redundancy and three strategies for implementing redundant distributed components using DDS.

**FAULT TOLERANCE AND REDUNDANCY**

Fault Tolerance is the ability of a system to continue operating, with little or no degradation in service, in the face of a hardware failure or software bug. Implemented by redundant components, fault tolerance is a "game of statistics" in that one needs to detect the failure, switch to a backup component to continue operating correctly, alarm the failed component, and allow replacement before a second failure, the so-called Mean Time To Repair (MTTR).

Ideally, redundancy guards against any single hardware or software failure. Redundancy relies on the probability that two failures within a short duration are highly improbable. With critical systems, controller redundancy alone is insufficient. A fully redundant system requires end-to-end redundancy, system isolation to avoid the effects of a cascading fault, and failure detection of any critical component. This means all system components, including the network, must be redundant and actively monitored so the MTTR of any single component is minimized. Failing to detect and repair even a fault of a backup component allows the system to effectively operate in a non-redundant mode (referred to as ‘simplex’ operation) until an inevitable failure of a primary component occurs, leaving the system with no fallback position to continue operating correctly.

A diagram of a diagram of a ship

Description automatically generated

*Figure 1. Basic Fault-tolerant System*

Figure 1 shows a basic fault-tolerant system. While all of the components are redundant, passive (e.g., the network), or receive-only components (e.g., Thrusters) need only be monitored for failure and able to compensate for the loss of an element. Components that drive the system should have sufficient logic to determine which components are operating correctly in the case of a discrepancy.

**REDUNDANCY ROLLS AND STATES**

A redundant logical component is typically made up of two or more physical components and defined by their role plus their state. The physical redundant components are often designated the roles of Primary and Secondary (or backups), where the Primary controller outputs are used to drive the system, and the Backup(s) stand ready to take over as Primary in the event of a failure. By definition, the Primary is in the Active State, indicating it is operational and driving the system. Each of the Backup controllers can be in any one of the following states: Cold/Warm-Standby or Hot-standby. Hot-standby designates that the switchover of the Active role from Primary to Backup is the time to detect the Primary has failed (that is, the Backup has all the state necessary to take over as Primary.)  Cold and Warm-Standby may require a lag to gain sufficient state to take over as the Primary. The method employed is dependent upon the system redundancy switchover requirements.

There are also strategies that use no role or state. Here, each redundant controller actively drives the system where its outputs are used only if the other redundant components are in agreement.

**SUMMARY OF FAULT TOLERANCE REQUIREMENTS**

1. End-to-end
2. Timely detection of a fault on both the Primary and Backup units
3. Timely switchover to a backup component
4. Alarm and Repair of a failed component. This requires the redundant components to be field replaceable.

“Timely” is dependent upon your system requirements

**REDUNDANCY STRATEGIES**

There are multiple approaches to providing redundancy of the driving components. In the context of Figure 1, the Navigation Control. We'll look at three approaches, including:

1. Simple Duplex
2. Three-way Voting
3. Three-way Selected Sample

There is also a hardware fault-tolerant approach affectionately called "Pair-and-spare." Stratus Computer Inc. popularized this design in the 1980s, but this approach only addressed hardware faults. Today, with far more reliable hardware, software faults must also be addressed.

**SIMPLE DUPLEX**

The Simple Duplex approach utilizes two off-the-shelf controllers, designated Primary and Secondary. DDS Exclusive Ownership QoS is used on the control topic with higher Ownership Strength assigned to the Primary.

A diagram of a bus system

Description automatically generated

*Figure 2. Simple Duplex*

Health monitoring would be implemented with either an explicit heartbeat topic using Liveliness Quality of Service (QoS). This allows each controller to detect the loss of the other controller and raise an alarm for repair. DDS Exclusive Ownership QoS enables the Primary's data, with higher Ownership Strength, to be passed by DDS to the receiving controller application. Both controllers receive and process data in relative synchronization. If a failure occurs on the Primary, DDS will use the Secondary controller's data once the Deadline QoS is missed. Upon the original Primary returning to service, DDS will instantly resume utilization of the restored Primary's data based on the higher Exclusive Ownership Strength. This last step could result in an unstable Primary oscillating in and out of service. To prevent potential oscillation or a primary failure but still emitting data at the Primary ownership level (referred to as a zombie process), upon the Secondary taking over, the ownership strength could be raised to a value greater than the original Primary's ownership strength. That is, three ownership levels would be used: Primary (high), Secondary Standby (low), and Secondary Active (highest). At some point, the original Primary would need to be restored Active to allow full redundancy to be restored.

This approach, while simple, assumes the failure mode is catastrophic (i.e., the process or controller completely fails). It does not cover more subtle issues of software bugs where both controllers send different results. Here, it is impossible to determine which controller is correct.

**THREE-WAY VOTING**

The Three-way voting approach utilizes three off-the-shelf controllers. The controllers will unanimously select primary, secondary, and tertiary roles. Any dissenting vote is flagged as a failed unit and Alarmed. The Primary will drive the system, with the Secondary and Tertiary standing ready to take over. Technically, unless there is a double fault, the Tertiary would never drive the system and is used to validate a majority vote. DDS Exclusive Ownership QoS is used on the control topic with the highest Ownership Strength assigned to the Primary and the lowest Strength assigned to the Tertiary. Health monitoring would be implemented with an explicit heartbeat topic using DDS Liveliness Quality of Service (QoS). This allows each controller to detect the loss of the other controller and raise an alarm for repair. DDS Exclusive Ownership QoS enables the controller's data, with the highest Ownership Strength, to be passed by DDS to the receiving controller application. All three controllers receive and process data in relative synchronization. Upon a loss or gain of a controller, the remaining controllers will re-elect the Primary, Secondary, and Tertiary, if applicable. If the Primary fails, DDS will use the Secondary controller's data once the Deadline QoS is missed. It should be noted that the re-election process is independent of the deadline miss, and DDS using the secondary's data to drive the system. To prevent "Primary Role" Oscillation, upon a controller loss, controllers will promote existing controllers to the higher roles; that is, after the re-election process, there will be at least a Primary and Secondary. New Controllers enter the lowest available role after the system is up (so-called late joiners). DDS Durability QoS on the Vote Topic will allow a new Controller to understand the system state and allow it to enter with the highest unfilled role.

**A diagram of a bus system

Description automatically generated**

*Figure 3. Three-way Voting*

Upon initial bootup, three-way voting should allow a grace period to wait for all three controllers for full three-way voting and redundant operation. The grace period timeout will enable the system to come up in either Simplex or Simple Duplex mode, as described above, allowing late joiners to come in later to reach full Three-way voted Redundancy.

Because this mode of operation is a superset of Simple Duplex, it is possible to include the techniques described above to guard against a zombied Primary failure mode. That is, upon promotion from the Secondary to Primary, the newly elected Primary could use a higher Primary Ownership Strength than the original Primary strength.

**THREE-WAY SELECTED SAMPLE**

The Three-way Selected Sample approach utilizes three off-the-shelf controllers. Unlike the previous two approaches, the Three-way Selected Sample has no defined roles, exclusive ownership, or ownership QoS. All controllers are peers and process the same input to produce the same output control commands. Commands are issued as Shared Ownership QoS. Heartbeat messages are used to detect and alarm the failure of a controller. With Selected Sample, the receiver, in this case, the Thruster, will receive all three sample instances, sending one copy of the sample to the application once it matches with at least one other sample instances. Any dissenting sample instance will be flagged, and an alarm will be raised on the source of the sample.

**A diagram of a system

Description automatically generated**

Figure 4. Three-way Selected Sample

This approach can impact performance as each sample passed to the application has to be compared in the Redunancy Layer software for a match with at least one other sample instance. Further, sample instances must be coherent in that the sample instances from each controller must be presented as one collective set of samples (Connext has a mechanism for this). Also, care must be taken in emitting a control command sample that is the result of integrating several input samples at the controller. In this case, the controller has to get to a proper state of integration prior to going "online." However, the advantage of guarding against software bugs offsets these complexities. By using different sources to write the software for each controller with the same specification, the likelihood of two or more controllers sending an erroneous sample at the same time is highly improbable.

**REDUNDANCY LAYER**

To isolate and abstract the applications in the system from a particular redundancy implementation, we introduce a Redundancy Layer (RL). The RL is responsible for the following activities (note activities are dependent upon the Redundancy Approach selected:

1. Active / Standby state selection
2. Failure detection and alarm
3. Switchover selection
4. Coherent Data presentation and matching
5. Network Monitor, Alarming and switching over

Application Layer

Redundancy Layer

DDS

Figure 5. Redundancy Abstraction Layer

A single API is presented to the application, minimizing exposure of the redundancy implementation.

**SUMMARY**

Table 1 below, highlights the Redundancy Approaches and attributes.

**A blue and white table with white text

Description automatically generated**

\* included for completeness

*Table 1, Redundancy Approaches and Key Attributes*

Table 2 below, highlights the key DDS mechanisms associated with the various approaches.

A screenshot of a computer component

Description automatically generated

*Table 2, Key DDS Mechanisms*

**FOR ADDITIONAL INFORMATION:**

Please contact us at info@rti.com if you have questions or comments regarding this whitepaper.

An example of a tracking system using the Three-way voting approach can be found on [Github](https://github.com/psmass/pixytracker/tree/xml_create_waitset).

**ABOUT RTI**

Real-Time Innovations (RTI) is the largest software framework company for autonomous systems. RTI Connext® is the world’s leading architecture for developing intelligent distributed systems. Uniquely, Connext shares data directly, connecting AI algorithms to realtime networks of devices to build autonomous systems. RTI is the best in the world at ensuring our customers’ success in deploying production systems. With over 2000 designs, RTI software runs over 250 autonomous vehicle programs, controls the largest power plants in North America, coordinates combat management on U.S. Navy ships, drives a new generation of medical robotics, enables flying cars, and provides 24/7 intelligence for hospital and emergency medicine. RTI runs a smarter world. RTI is the leading vendor of products compliant with the Object Management Group® (OMG®) Data Distribution Service (DDS™) standard. RTI is privately held and headquartered in Sunnyvale, California with regional offices in Colorado, Spain and Singapore.

Download a free 30-day trial of the latest, fully-functional Connext software today: [www.rti.com/downloads](http://www.rti.com/downloads).