# Redundancy for OPC

*by Robert McIlvride and Andew Thomas*
*Cogent Real-Time Systems Inc.*

Early one morning, Mel Farnsworth was sitting in the control booth at the Hardy Automotive Parts assembly line, drinking his final cup of coffee before the end of the shift. Watching the line meter graph, he noticed that the yield and efficiency trends for the Line 3 had dropped to zero. He looked down through the control-room window, but Line 3 seemed to be rolling right along. What was the problem?

The line was running smoothly, but Mel wasn't getting the data he needed. Somewhere between the PLCs and his HMI display there was a data disconnect. Maybe it was a fieldbus problem, or a bad network connection. Perhaps it was caused by his OPC server, or possibly even his HMI system. Whatever the reason, since Mel's data connection was a single chain, one break in the chain means that he didn't get his data. To minimize this kind of risk and ensure the highest possible availability, mission-critical systems often use redundancy.

**What is Redundancy?**

Redundancy in a process control system means that some or all of the system is duplicated, or redundant. The goal is to eliminate, as much as possible, any single point of failure. When a piece of equipment or a communication link goes down, a similar or identical component is ready to take over. There are three types of redundant systems, categorized by how quickly a replacement (or standby) can be brought online. These are cold standby, warm standby, and hot standby.

*Cold standby* implies that there will be a significant time delay in getting the replacement system up and running. The hardware and software are available, but may have to be booted up and loaded with the appropriate data. Picture the olden days of steam locomotives. The cold standby was the extra engine in the roundhouse that had to be fired up and brought into service. Cold standby is not usually used for control systems unless the data changes very infrequently.

*Warm standby* has a faster response time, because the backup (redundant) system is always running, and regularly updated with a recent copy of the data set. When a failure occurs on the primary system, the redundant system can disconnect from the failed system and connect instead to the backup system. This allows the system to recover fairly quickly (within seconds, usually), and continue the work. Some data will be lost during this disconnect/reconnect cycle, but warm standby can be an acceptable solution where some data loss can be tolerated.

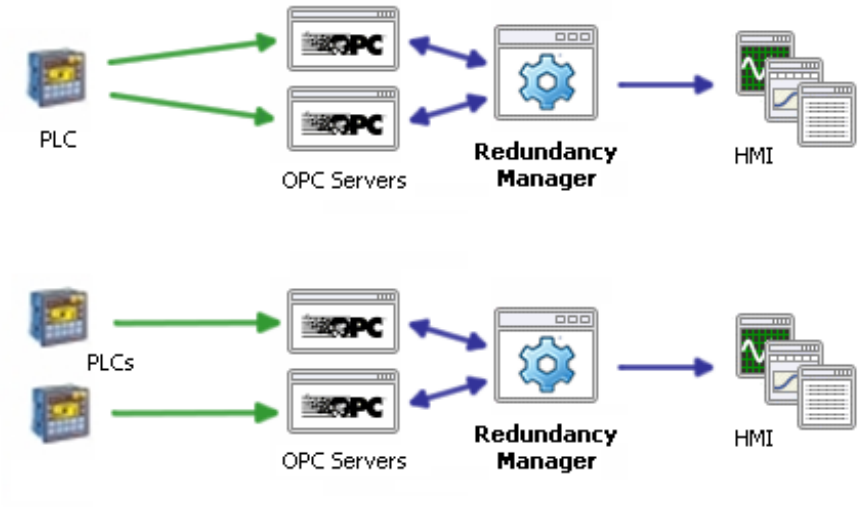
*Hot standby* means that both the primary and secondary data systems run simultaneously, and both are providing identical data streams to the downstream client. The underlying physical

system is the same, but the two data systems use separate hardware to ensure that there is no single point of failure. When the primary system fails, the switchover to the secondary system is intended to be completely seamless, or "bumpless", with no data loss. Hot standby is the best choice for systems that cannot tolerate the data loss of a cold or warm standby system.

**A Typical Redundant OPC System**

What does redundancy look like in an OPC-based system? A typical scenario would have two OPC servers connected either to a single device or PLC, or possibly duplicate devices or PLCs. Those two OPC servers would then connect to some kind of OPC redundancy management software which, in turn, offers a single connection to the OPC client, such as an HMI. The redundancy manager is responsible for switching to the secondary OPC server when any problem arises with the data coming from the primary OPC server. This scenario creates a redundant data stream from the physical system all the way to the HMI.




The most common use of redundancy in OPC is for OPC DA, but it is possible to configure redundant OPC A&E or OPC UA systems. The principles are the same. Sometimes, on large systems, it is necessary to configure multiple redundant pairs. Redundancy can also be configured over a network, using DCOM or OPC tunneling. For a networked configuration, the redundancy manager would normally reside on the OPC client machine, to minimize the number of potential points of failure.

Although cold or warm standby may be useful under some circumstances, typically an engineer or system integrator implementing a redundant OPC system is looking for hot standby. This is the most useful kind of redundancy in a process control system, and at the same time the most difficult to achieve. Let's look a little more closely at that all-important task of the OPC redundancy manager in a hot-standby system—making the switch.

### Making the Switch

Put simply, a hot-standby redundancy manager receives data from two identical inputs, and sends a single output to the OPC client. It is the redundancy manager's job to determine at all times which of the two data streams is the best, and switch from one to the other as soon as possible whenever the status changes. The switch can be triggered by a number of different kinds of events:

- Single point value change – to or from a certain value, achieving a threshold, etc.
- Single point quality change – for example, from "Good" to any other OPC quality.
- Multiple item monitoring – if the quality or value of *any* point in a group goes bad.
- Rate of change monitoring – if points change value more slowly than expected.
- Network breaks and timeouts – checked with some kind of heartbeat mechanism.

Once the switch has occurred, the system or the redundancy manager itself might have the ability to send an alarm or email message, or even launch some kind of diagnostic or investigative program. It might also be able to log diagnostic information about the state of the primary OPC server or network connection. And in a system that distinguishes between primary and secondary inputs, there will often be a means to favor the primary input, and switch back to it when possible, sometimes referred to as a *fallback*.

## Practical Considerations

The idea of redundancy is not difficult to grasp, but implementing it takes some thought. An initial decision on cold, warm or hot standby will impact all aspects of the implementation. The choice of proper hardware and software is critical for a well-functioning system. Robust system architecture is also important, especially if the connection is across a network. In addition to selecting OPC servers and planning the network infrastructure (if necessary), an important decision will be the software used to manage the redundancy. Good redundancy management software should be easy to use, with no programming necessary. The technology should be up to date, capable of running on the latest version of Windows. There should be an absolute minimum chance of data loss during a switchover, even over a network.

### The Timer Pitfall

In practice it is not possible to achieve a completely seamless switchover in all cases, even with a hot standby system. For example, if a network failure occurs on the primary connection, a certain amount of time will pass before a redundancy manager can detect that failure. Data transmitted during this period will fail to arrive, but the redundancy manager will not be able to distinguish between a failure and a normal pause in data flow.

Many redundancy managers implement timers to periodically check the network connection status to try to minimize this delay, but a switchover mechanism based on periodic timers will always suffer from data loss. Systems with multiple timing parameters will often result in additive delays, where the fastest possible switchover for the system is the sum of these timing delays. In addition, the use of timers to detect network failure can result in a configuration problem where the system integrator must trade off switchover latency against false-positive network failure detection. This effectively becomes a tradeoff between system stability and responsiveness.

Using timers to periodically check data values or qualities, or poll the OPC servers, is also problematic because timers introduce unnecessary latency into the system. Whereas a network failure must be detected based on timing, a data value or quality change can be detected immediately as the event occurs. It is usually best to avoid systems based on time-based value change detection, and use event-based object monitoring instead.

**Object and Link Monitoring**

A good redundancy manager should be able to support both object monitoring and link monitoring. *Object monitoring* means the ability to monitor individual points, and make a switchover based on an event. For example, if a designated watchdog tag changes in a significant way, such as turning negative or going over a specified threshold, it can trigger a switch to the secondary OPC server. Or maybe you'd like to monitor a group of points, and if the quality of any of them goes to "Bad" or "Unconnected", you can switch.

*Link monitoring* is especially useful for networked connections. Your system will need a way to detect a network break very quickly, to prevent data loss. For hot standby on high-speed systems with fast data update rates, timeout detection with a sub-second response rate is essential. In any event, the system should be able to detect a timeout for a failed network connection, as well as a failure to receive data. This distinction is important. It may take seconds or even minutes to detect a communication failure, but a redundancy manager should be able to detect a stoppage of data flow in an amount of time very close to the true data rate from the physical system. The redundancy manager should be able to switch from one source to the other based solely on an observation that data has not arrived from the primary connection, but has arrived from the backup system.

Some systems use COM timeouts for link monitoring. This may be acceptable for circumstances where relatively long data outages are tolerable, but we do not recommend relying on COM timeouts for hot or warm standby.

**Smart Switchover**

The behavior of the redundancy system during a switchover can be significant. For example, suppose the primary and secondary connections have both failed for some reason. A typical redundancy manager will begin a cycle of attempting to attach to one and then the other OPC

server until one of them responds. The redundancy manager will flip-flop between the two indefinitely, injecting sleep periods between each flip-flop to reduce system resource load. This sleep period is itself a source of latency. A smarter switchover model is to maintain a source health status that allows the redundancy manager to only switch over when a source status changes. This allows the redundancy manager to effectively idle, or perform simultaneous reconnection attempts, until a source status changes, then immediately respond without introducing extra latency. Smarter switching logic can result in substantially reduced system load and switchover times.

**Forced Switching vs Preferred Source**

It is useful to be able to select one data source over another, even if the currently attached source is healthy. A naïve redundancy manager will "force" the user to switch, even if the backup system is not available. This will again result in a flip-flop behavior as the redundancy manager attempts to switch to the unavailable backup source. A much better approach is for the redundancy manager to understand the concept of a *preferred* source that can be changed at runtime. If the preferred source is available, the redundancy manager will switch to it. If the user wants to switch from one source to another, he simply changes the preferred source. If that source is available, the switch will be made. If it is not, the redundancy manager will make the switch only when it becomes available. This eliminates the flip-flop behavior while at the same time eliminating the data loss associated with the minimum of two switch cycles that the naïve redundancy manager will impose.

**Accessing Raw Data**

A good hot redundancy system will give the client application access not just to the redundant data, but also to the raw data from both sources. This gives the client application the option of presenting diagnostic information about the system on the "far side" of the redundancy manager. Most redundancy managers hide this information so that a client application would have to make and manage multiple connections to access the raw data, if it is possible at all.

**Other options and features**

In addition to the above capabilities, a good redundancy manager may offer additional features for your convenience. It might provide the option to refresh the entire data set at switchover. Maybe it will send out emails or even launch additional programs at each switchover. This can be useful for notifying key personnel of the system status. It may log diagnostics to provide valuable information about the reasons for making the switch. Some redundancy managers can connect to multiple servers, and create multiple redundant connections. Others can let you work with subsets of the data. Another desirable feature is the ability to assign the primary and secondary data sources, and to trigger a fallback from the secondary to the primary data source once the problem that caused the switchover has been resolved.

As control systems continue to grow in complexity, and as we rely more and more on them, Mel Farnsworth's situation will become more common, and more costly. If data connectivity is crucial to the success of the company, it would be wise to consider the possibility of installing a redundant system, and to weigh the options carefully when choosing the key components.

▫ ▫ ▫

Founded in 1995, Cogent Real-Time Systems provides versatile and reliable middleware products to enable real-time data integration and access for industrial, embedded, and financial systems. Customers include Siemens, ABB, Honeywell, IBM, GE, Statoil, Goodyear, BASF, Cadbury Chocolate, and the Bank of Canada. For more information, please contact Cogent at info@cogent.ca or visit our web site at www.cogentdatahub.com. You can also call us at +1 (905) 702 7851.