

# Open architecture for high availability

By James G. Lawrence

Telecommunications, data communications, and e-commerce services are converging over packet-based networks, increasing the complexity and diversity of systems and networks that provide these services. Also increasing are expectations on the part of service providers and customers that these services will be available 100 percent of the time. This demand is driving the requirement for ever-increasing levels of system availability. The traditional benchmark of 5-nines (99.999 percent) system availability is no longer sufficient to meet customer expectations for *always-on* continuous service delivery. Customers increasingly want uninterrupted services and connections, regardless of internal system faults or failures.

Keeping customers connected and preserving data in real time requires a new approach to system design. As illustrated in Table 1, designing for high availability (HA) has historically meant building a communications and computing infrastructure with a minimal amount of downtime. The current HA network standard within the network infrastructure industry is now at least 5-nines (99.999 percent) availability, or slightly more than five minutes downtime per year, including outages resulting from either planned upgrades and maintenance or from unplanned failures. In some telecom applications the industry is beginning to see requirements for 6-nines. The problem with these traditional measurements is that they allow for some downtime, and that is becoming increasingly unacceptable.

## Limitations of traditional architectures

Designing for availability has traditionally focused on providing reliable hardware, backing it up with redundant architectures in case of failure, and improving switch-over techniques to backup components. This approach has led to the current generation of HA platforms. While providing system availability at a level of 5-nines, or even 6-nines, is praiseworthy, it does not fully meet needs and expectations of

Number of Nines	Downtime per Year	Typical Application
3-nines (99.9 percent)	~ 9 hours	Typical desktop or server
4-nines (99.99 percent)	~ 1 hour	Enterprise server
5-nines (99.999 percent)	~ 5 minutes	Carrier class server
6-nines (99.9999 percent)	~ 31 seconds	Carrier switch equipment

**Table 1. Allowable downtime per year for selected network applications**

today's network service providers and their customers.

The emergence of 24x7 e-commerce has raised the bar for what constitutes adequate availability. Internet-based services and businesses need to be *always-on*. The central problem with traditional approaches to HA is their failure to address the issue of maintaining transaction state integrity during failover intervals. Customers are not interested in metrics of system resources or switchover times. They want continuous service, with any system issues handled transparently with no interruptions. The system or network must continue to deliver the acceptable service regardless of hardware, software or even operator event. Neither scheduled maintenance nor unexpected failure should be allowed to prevent or disrupt the provision of service to a customer.

## Beyond reliability

Before HA became a requirement, system developers who wanted to design dependable systems thought in terms of hardware platform reliability. A well-designed system with reliable hardware satisfied the common-sense notion of "a system that works." Reliability is a measure of how dependable a system is once it is actually put into use, a measure of "not breaking down." As such, reliability is still an important and useful principle. Beginning with good specifications and design practices, well-manufactured and well-tested components go a long way toward assuring that the system can remain available, up and running, and ready for use. Historically, the reliability strategy was first to build great hardware that doesn't break down and then a backup system in the event of a primary system fault.

Redundancy (the foundation of all HA strategies) was introduced to provide system reliability in an increasingly complex and unreliable world.

Hardware reliability has continued to improve, and MTBF (mean time between failures) continues to rise. Regardless of this fact, hardware reliability alone cannot meet the requirement for continuous availability, because eventually hardware will fail, and increasingly more faults are originating in software. The fact that software is becoming inherently less reliable than hardware is explained by its increasing complexity – and the fact that every possible configuration of use and data cannot be tested prior to release. As a result, software problems are often discovered after the software is deployed. As systems become more and more software-driven, total fault prevention has become virtually impossible to achieve, and it has been replaced by fault management involving the integration of multiple strategies.

Present fault management strategies include:

- Maintaining high levels of hardware reliability
- Hardening of software and drivers
- Rigorous system validation
- Hardware and system redundancy, with automatic failover protection
- First-generation fault management middleware.

## First-generation HA systems

First-generation HA systems are an important step forward, featuring redundant hardware architectures, standardized and hardened drivers, and HA fault management middleware. The function of the mid-

middleware is to discover faults, often requiring multiple layers of messaging. Once the fault is detected, the middleware re-provisions system resources to recover from the fault, through failover mechanism, for example. The problem with these solutions is that the time needed to detect the fault and complete the failover process can result in a service interruption.

### Building continuously available platforms

Traditional strategies for hardware reliability, software hardening and system-level validation provide a good foundation, but by themselves they cannot ensure continuous service delivery. Meeting customer expectations for service continuity requires providing HA at the level of 5-nines or better, and adds the requirement of maintaining the integrity of connections and transactions without interruption, despite hardware or software failures. This new definition of availability goes beyond merely reacting to faults and includes strategies for avoiding them. In contrast to traditional HA, hardware or software faults are not avoided but are expected to occur. The system is designed from the outset to anticipate and work around faults before they can trigger system failures. This requires involves additional requirements including availability validation, real-time manageability, and anticipative fault management.

- *Availability validation* goes beyond traditional scope of system validation, because it is no longer enough to simply state a measurement of “nines.” Validation suites must test and validate the level of availability that a system can provide and what takes place following a fault, including the interval required for failover and the function of recovery mechanisms.
- *Real-time manageability* is another central requirement, including the ability to support initialization and provisioning on the fly. This requires high speed, highly accurate system models to enable dynamic reconfiguration, along with the validation techniques for manageability. So ease of OAM (operation, administration, and maintenance) is really a key aspect of continuous service delivery.
- *Anticipative fault management* involves the proactive detection of faults to enable *rational isolation*. With traditional failover techniques, when a node

appears to be failing or performance becomes unacceptable, the system shuts down the entire node and fails-over to a backup node. Because this level of redundancy is not cost-effective, it is preferable to take down an isolated I/O board, for example, rather than an entire network node.

Instead of counting on the hardware platform alone to avoid faults, the platform design relies on availability management software to mitigate and manage faults that can be expected to occur. This approach goes beyond HA because designing for fault management takes precedence over designing for fault avoidance. The goal is to anticipate faults and execute recovery as quickly as possible without service interruption.

Providing the quickest possible fault detection and recovery time is of key importance. This means that availability management software must execute the fault recovery policy before the application or service is impacted in a manner that interrupts continuity of service. In addition, achieving a greater-than 5-nines system availability leaves no room for downtime for upgrading or maintenance. Availability management must be able to handle hardware and software upgrades without taking the system out of service.

### Building blocks for continuous availability

Creating a system characterized by virtually no downtime requires interoperable hardware and software building blocks that can then be combined as needed to create a system. Major functional blocks are separated into hardware, OS, management middleware and applications.

- *Hardware capabilities* of fault-managed systems can be generally categorized into three sets: redundancy to allow continued processing after failure, highly reliable (often redundant) communication among components, and chassis management, including fault detection, diagnosis, isolation, notification (alarms), recovery, and repair.
- *OS capabilities* include functions to isolate hardware and software faults, prevent their propagation, and mask their impact. These functions help prevent application errors or faulted hardware from bringing down the entire

system. Dynamic reconfiguration and enhanced device drivers allow graceful replacement of failed hardware. In addition, the OS provides services for autonomous fault-management, reporting faults externally, and interfacing with HA capabilities in the middleware and application layers.

- *Management middleware* is the software component that oversees the system’s configuration and fault management services. The availability management component operates automatically in real time without operator intervention. A state-aware system model represents components, modeling and monitoring their status, topology, and dependencies. The middleware collects system information and uses it to detect and diagnose system anomalies or faults. The middleware then acts on faults by dynamically reconfiguring the status, configuration, and dependencies of the components to rapidly recover and maintain service. Management middleware also checkpoints (periodically transfers) data between a component and its redundant units in order to maintain operation during system reconfiguration.

There are many ways for an application to participate, control, and operate within a highly available system. The management interface should allow the application to monitor operations and send status, heartbeat and checkpoint information. An application may also need to receive this type of information from other applications. Finally, an application may need to initiate a failover or other recovery action and be able to be unloaded, loaded, and restarted while the system is operational.

Continuous services depend on the ability of each part of a system and network to work reliably together to deliver services without interruption. Hardware and software redundancy enables the management software to replace failed resources with the appropriate standby. In addition, accomplishing this without downtime or loss of client state requires a comprehensive, unified, high-performance solution. Given both the architectural complexity of such systems, as well as the performance standards required to actually achieve continuous availability, it is crucial for the management software to carry a low overhead. This software must be optimized for

speed and efficiency, to promote overall system performance.

The implementation of uninterruptible service requires management software with the following capabilities:

- Collect system data in real time
- Configure and maintain a state-aware model of the total system
- Checkpoint data to redundant resources
- Detect, diagnose, and isolate faults
- Generate alarms and inform external network management of status
- Perform rapid, policy-based recovery
- Dynamically manage configuration and dependencies of all components
- Provide administrative access and control.

**Needed: integration, balance, and performance**

The telecommunications industry has long understood the concept of delivering uninterrupted service, because telephone users have come to expect dial-tone or “911” levels of service. Over the years the industry has developed a number of in-house availability management solutions, many of which are proprietary, tied to particular operating systems or hardware, and complicated to upgrade as new technologies emerge. In today’s environment of explosive network growth such ad hoc and application-specific development approaches are far too time-consuming and expensive to acquire and support. Traditional systems are much too large and complex for a single vendor to provide all of the required functionality. Due to the cost and complexity of developing proprietary systems, telecom and Internet infrastructure equipment manufacturers are now realizing the need for standard open architecture building blocks.

Today’s communications and networking industry requires a comprehensive, cross-platform and off-the-shelf solution, based on a standards-based open architecture that can provide a combination of integration, balance, and performance (see Figure 1).

- Integration is required to accommodate the new platform vision of tightly integrated functions across interoperable hardware and software building blocks from a variety of providers. This requires the integrating “glue” consisting of standard interfaces needed to ensure that hardware components, drivers, operating systems, and middleware will work together. The industry

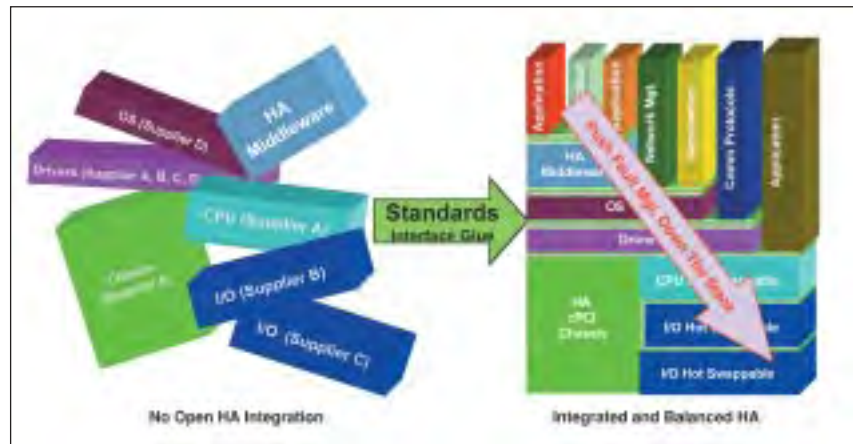


Figure 1

needs to develop standard interface specifications.

- Balance requires driving fault management as close to the point of failure as practical. This helps avoid placing the entire fault management burden on middleware. Identifying and mitigating faults closer to their point of origin helps the system avoid service interruptions. For example, hardware components that can detect and resolve faults before they impact drivers, the operating system, or the application can save valuable time. New HA standard interfaces need to be developed to push fault management to the lowest practical level. In addition, managing faults at lower levels of the solution stack (HW, OS, management middleware, application) increases the scalability of the system.
- Performance optimizations are also needed for HA, which means meeting the requirements of different system configurations and classes of equipment. This requires quantifying levels of availability and then tuning the system’s fault management capabilities appropriately through the ability to selectively turn-on and turn-off different high-availability features. For example, a data communications system typically requires high levels of data availability and integrity, where a control point requires connection availability and integrity. These examples would require significantly different availability implementations. This requires a standard set of tools and designed-in capabilities to dynamically activate the appropriate availability function.

**Benefits of an open architecture specification**

Open architecture for availability allows manufacturers to reduce their time-to-

market while maintaining the highest levels of dependability for their customers. Realizing an open architecture in turn requires the design and implementation of solutions within an open specification framework. Interoperable building blocks will then integrate and perform in a uniform and predictable manner. In addition, the framework will provide scalability, which will enable manufacturers to match system performance and price to meet specific sets of user requirements. In addition, an open architecture specification formally manages the integration of what would otherwise be mutually exclusive technologies from multiple vendors, and this will ultimately improve dependability and provide a higher degree of confidence in the validation tests used to qualify network services prior to deployment. An open architecture specification will also encourage the development of commercial-off-the-shelf components and foster the competitive environment needed to improve product features, cost, and performance. In addition, manufacturers will be able to focus critical design resources within areas of core competency rather than on the integration, validation and certification of system building blocks.

While the primary application area for an open architecture specification for availability is broadly aimed at systems for telecommunications, data communications, and Internet infrastructure, many other application areas share similar requirements, and these will also benefit. Some examples of telecommunications and Internet infrastructure applications that will benefit from an availability specification include:

- Softswitches and voice over IP gateways
- Telecom network service control points

- Telecom network switch transfer points
- Wireless base station controllers, radio network controllers
- Remote access controllers
- Web hosting, caching, and e-mail servers
- Broadband distribution nodes
- Voice portals and unified messaging systems.

### Conclusion

The ability to meet customer expectations for highly available, dependable services without interruption is an absolute requirement for packet-based networks. The traditional availability metric of 5-nines is no longer sufficient going forward, because consumers, network users and network service providers now require services to be available 100 percent of the time. Existing techniques for hardware reliability, software hardening, and management middleware do not provide the ease of integration, balanced fault management architecture, and performance required for ultra-dependable service provisioning.

This situation calls for a new approach to integration, balance, and performance optimization for HA, achieved through a new set of open standards.

An open, HA architecture is based on the concept of interoperable building blocks that can be integrated to build a system. Independent building blocks for hardware platforms, operating systems, HA middleware packages, and application software provide telecommunications and network equipment manufacturers with the flexibility to select from multiple vendors at each level in the system design. Systems based on such open standards for availability can meet the demand for continuously available, ultra-dependable services, while speeding the development of new devices and applications.

*James G. Lawrence is currently responsible for the High Availability Strategies and Architecture group within Intel's Embedded Intel Architecture Division (EID). He has more than 20 years in the computer platform industry as a developer, manager, and director of leading edge technologies. Over the last four years, Jim has been focused on the telecommunications industry as director of engineering, director of program management, and business unit director.*

For more information, contact Jim at:

James G. Lawrence  
**Intel Corporation**  
5000 W. Chandler Blvd.  
Chandler, AZ 85226  
CH6-238  
Tel: 480-552-0576  
E-mail: james.g.lawrence@intel.com  
Web site: www.intel.com