Economics of High Availability for Telecommunications Systems

An Intel® Primer

Table of Contents

Executive Summary

Several forces are at play in the move toward highly available telecommunications systems built with cost-effective off-the-shelf components (COTS). Deregulation, converging voice and data networks, the Internet, not to mention the fragile economy — all demand revenue-enhancing services that can be developed quickly, are easy to deploy, accommodate a growing subscriber base, and improve the service provider's overall return on investment (ROI). Balancing these needs presents new challenges to the service providers, system developers and component suppliers alike.

Solutions that continues to gain momentum is using COTS building blocks for today's demanding communications solutions. It has been shown that COTS components have the ability to drive down total system costs due to economies of scale and increased interoperability. Telecommunications equipment manufacturers (TEMs) are freed to concentrate on combining off the shelf components and adding specialized vertical services, resulting in shorter solution development time. The COTS approach also results in reduced risk as it capitalizes on available, broadly used, and mature components.

Nevertheless, the COTS approach introduces some interesting quandaries as well, since solutions still need to satisfy the system availability, quality, and performance characteristics of existing proprietary installations. Considerable analysis is needed to determine the right mix of components that, when combined, will yield adequate price points. Yet, this is only effective if all of the components work together to provide a predictable level of reliability. This strategy represents a departure from the monolithic model where scalability and reliability are designed in from the very start, completely under the control of the solution developer.

The dichotomy is challenging, but can be analyzed in a fairly straight forward way and will be shown to follow basic technical and financial principles.

The purpose of this paper is to offer an increased understanding of the market segment and technical forces shaping this important facet of the industry. It will cover the basics of availability, including how it is measured; the most common causes for downtime and best-known methods to avoid them; the difference between network, system, and component availability; and the concept of failure groups and redundancy. It will also introduce the eight most common and most economic high availability architectures and provide insight into the pros and cons of each. The concept of "High Availability Economics" is introduced to offer a greater understanding of the techniques and science behind determining the cost of availability for each of these architectures.This paper concludes by proposing the "10 Theorems of Availability" that every designer should keep in mind when selecting COTS components and architecting a highly available, yet cost effective telecommunications system.

Defining the Need for High Availability

For service providers, highly available services via cost-effective systems remain crucial to their success. The combination of deregulation and increased competition has them focusing more and more on the economics of their solutions, yet they are unwilling to sacrifice the determinism of their legacy systems.

Several factors play a role in enabling service providers to generate services revenue, including

- Extending a deployed solution's time in market; especially as the network migrates from the TDM world towards the packet (or IP) world
- Increasing a solution's availability; when they are not operating, they are not making money
- Bringing new services to market as quickly as possible and extending the capacity and performance of existing systems as cost-effectively as possible. Innovative new services increase the service providers' subscriber bases and allow them to aggregate their fixed overhead costs over a larger revenue-generating population.

Service providers are continually looking for ways to reduce the total cost of providing services — or total cost of ownership (TCO), including the cost of procurement, development, deployment, and operation. To help meet these requirements, they are demanding ever more cheaper and more flexible systems that can still satisfy the availability, quality, and performance characteristics of existing proprietary installations. This places increasing pressure on their suppliers — telecommunications equipment manufacturers known as TEMs.

Traditionally, critical communications applications were hosted on expensive, proprietary monolithic systems, built with specialized hardware and software, and designed from the ground up to deliver a high level of availability and determinism. However, the high cost of building, deploying, maintaining, and operating these systems did not compare favorably with the low cost, open standards approach found initially in the desktop PC arena and now becoming pervasive in the next generation IP-based networks modeled after the Internet. As a result, many TEMs have begun looking at the COTS approach much more closely.

It is well accepted that COTS components have the ability to drive down total system costs. COTs can enable service providers and TEMs to generate revenue and remain competitive due to component cost optimization, decreased time to market, component quality and performance improvements driven by the increased competition, multiple component sources, reduced development risk, scalability, and predictable reliability and performance (or determinism).

COTS component developers have the expertise to build horizontal, cost-optimized building blocks that can be applied in a wide range of vertical solutions. For example, a well-architected building block, such as a network interface may be used for telecom switching, voice mail applications, and interactive voice response (IVR) applications. Subsequently, the supplier can market the products to several TEMs and build more of the same thing at a substantially reduced cost. In the traditional model, each TEM would need the expertise to build these common-function building blocks and, due to their specialization, would build fewer of them. With a COTS building block approach, TEMs can concentrate on adding specialized vertical business logic on top of generic building blocks to meet the needs of their customers. This significantly curtails their overall development efforts and the cost of developing the lower level components. It has been shown repeatedly that open standards-based COTS components that are highly interoperable with other elements of a solution can substantially decrease a TEM's development costs and time to market.

With building blocks available from multiple sources, TEMs can also avoid vendor lock-in while benefiting from the competition among suppliers that drives lower prices and rapid component improvements.

For service providers, COTS can help minimize the TCO, since the decreased solution development costs are passed along to them along with lower deployment and operations costs.

Calculating Availability

Many often equate reliability with availability, and certainly both concepts are critical to the concept of high availability. So when defining availability, it is helpful to note the subtle differences in the terminology:

- Reliability is the probability that something will not fail during a specific period of time
- Availability is the ratio of time that a service is available to total time. In other words:

  Availability = MTTF/(MTTF + MTTR), where MTTF stands for Mean Time to Failure and MTTR stands for Mean Time to Repair.

Availability approaches 100% as the MTTF increases to infinity, and the MTTR decreases to nothing; the higher the ratio as a percentage, the better.

9's	Availability	Downtime/Year	Examples
1	90.0%	36 days 12 hours	Personal clients
2	99.0%	87 hours 36 minutes	Entry-level businesses
3	99.9%	8 hours 46 minutes	ISPs, mainstream businesses
4	99.99%	52 minutes 33 seconds	Data centers
5	99.999%	5 minutes 15 seconds	Carrier-grade Telco, medical, banking
6	99.9999%	31.5 seconds	Military defense system, CG goal

As the chart suggests, availability is typically discussed in the range that is very close to 1. Although 99% uptime sounds good, it still results in over three and a half days of downtime per year. Most solutions are not considered highly available till they get close to 99.9% uptime — roughly nine hours of downtime per year. However, the telecom industry is used to availabilities in the range of four to five nines.

The problem is that the higher the availability, the higher the cost of providing the service. The big challenge for service providers, and their system and component suppliers, is striking the right balance between availability and cost.

Total system availability can be determined by progressively decomposing the system into the individual components — the hardware and software. The availability of hardware can be further attributed to the availability of the platform and the availability of the I/O boards, and so on for software.

Mathematically:

Availability of system = (Availability of hardware) and (Availability of software)
Availability of hardware = (Availability of platform) and (Availability of I/O boards)
Availability of software = (Availability of operating system) and (Availability of middleware) and
(Availability of software) and (Availability of application)

Not every component in the network must provide availability to the same level; typically, the number of nines is specified from an end user perspective. Component suppliers do not need to provide all components that are five nines, but they must deliver components that enable services that meet high aggregate availability requirements. The way components are combined in a system has a big effect on availability, as does the way systems are arranged in a network.

The Network's Impact on Availability

The location of the equipment in the network also affects availability. As the equipment moves towards the core infrastructure of the public network, the availability requirements become more aggressive; while at the edge, the availability requirements are more relaxed. The local loop, for example, does not have much built-in protection to guard against failures. In fact, Telcordia specifies availability objectives for local exchange networks as low as 99.93%[1], which it claims represents a balance of benefits and costs consumers find acceptable. However, the core infrastructure interconnecting these local exchanges must provide far greater availability.

The availability expectation for the different service types is also different. Critical and essential services, such as 911, are required to have much higher levels of availability than other non-essential services.

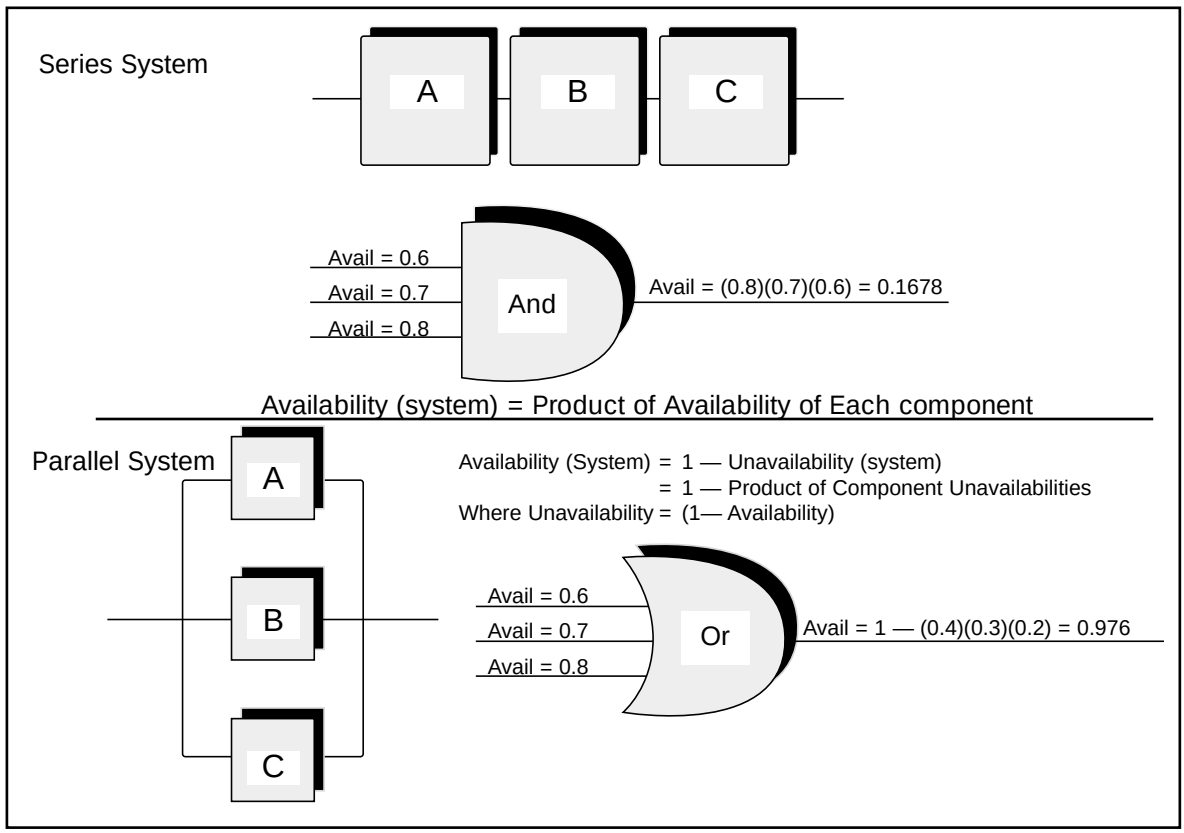
The *first* consideration in determining a system's or component's availability requirement is to determine where in the network the component will sit, what will it be used for, and how will it be combined with other systems to the ultimate end user solution.

The "Logic" of Availability

Measuring hardware availability takes into account the individual components that make up the system integrated circuits, transistors, diodes, resistors, capacitors, relays, switches, connectors, and more.

There are a number of established methods for predicting hardware reliability and availability for hardware components. For this study, various hardware suppliers provided aggregated platform-level and the telephony board level MTTF data via the Bellcore methodology. Their data was used that data as input and starting points, but were not based on individual electronic components to determine the availability characteristics.

The way components are combined has a big effect on the total availability of the solution.



If components are combined in a series, the solution relies on the availability of all components and the total system availability can be much lower than the availability of the weakest component. When If components are assembled in parallel, however, there is some leeway in the level of individual component availability. The total system availability can even be higher than that of the most available component.

The *second* consideration for developers is to utilize parallel availability wherever possible. Typically, architecting a solution for parallel availability does not add much cost to the total solution, as the cost is not realized until the parallel components are actually added. Service providers can deploy systems without the parallel component at first, then easily increase availability when the can be justified.

As redundancy is introduced into the system, the availability characteristics of a system change significantly. Availability calculations that must account for the effects of such redundancies, the success rate of failover to redundant components, the effect of the MTTR failed components, and the like — became a laborious and error prone endeavor. Better results can be arrived at using platform and telephony board MTTF data as input and employing Reliability Block Diagrams (RBDs)[2] to accurately and precisely determine the system level availability characteristics.

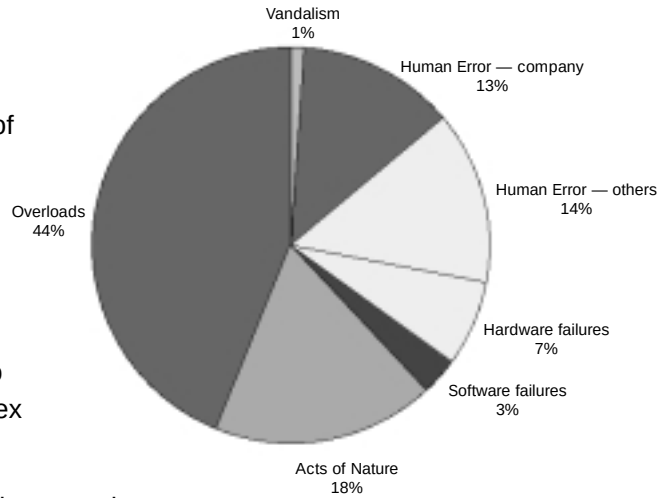
With RBDs, interconnected blocks can be constructed to show and analyze the effects of failure of any component in the system. RBDs can also account for probabilities of successful failover in cases where there is redundancy built into the system along with operational factors, such as the lack of immediately available spare parts. Software from companies such as Relex Software Corporation* can be used to derive the system-level availability characteristics. These packages calculate the comprehensive set of failure paths to determine the overall reliability and availability of the system under thousands of failure scenarios. Since the number of failure paths grows exponentially as the number of components in the system grows, the software performs Monte Carlo simulations[3] to arrive at the different reliability figures of merit for different confidence levels.

The *third* consideration for developers is to avail themselves of these relatively inexpensive tools and thoroughly analyze the availability characteristics of their solutions under the different availability configuration options. Such testing applies the necessary rigor to completely determine a system's unique availability fingerprint.

Causes of Downtime

Overload

One of the leading causes of service outage is overload of the system or network: too few resources processing too many calls. Examples include an initial introduction of a new service, or when unexpected spikes occur.



When new services are introduced, it is very difficult to predict end user response or how the service will actually behave under real world conditions. Modeling helps; but too often, when trying to predict the actual behavior of a complex system, integral factors may be overlooked.

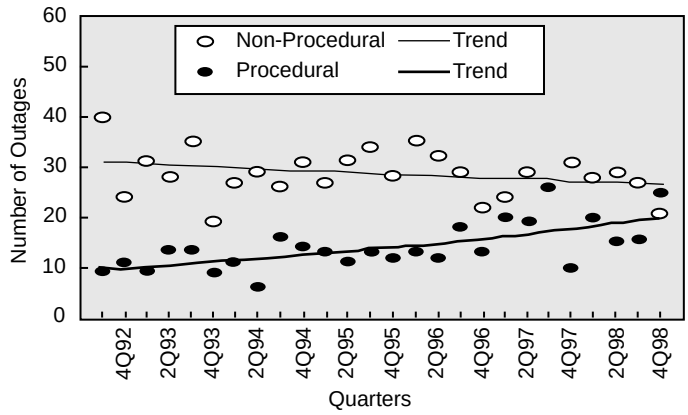
Usage spikes occur either when advertising campaigns work too well or during cyclical peaks such as Mother's Day.

Unless systems are properly designed to handle the influx of users or “shed” the load gracefully, they will fail. It may be difficult to determine exactly what component failed first. To compound the problem, minor failures often cascade into catastrophic events because the fault management systems themselves become overloaded.

Ensuring systems are designed to handle excess load is the *fourth* consideration for developers. Systems must provide some form of load shedding and allow for a graceful scale back of services when things start to go wrong. Operations, administration, and management (OA&M) systems, often relied on to assist in preventing excess load, must also be made highly available and fault resilient. Otherwise, they can bring down an entire system or compound the availability problems.

Planned vs. Unplanned Downtime

Downtime may be planned or unplanned. Planned downtime is caused by the need to upgrade, add new features, or conduct preventative maintenance. Unplanned downtime is caused by system failures or operator error that usually results from poor training, over-complexity, inadequate usability, or under-skilled personnel.



Source: Network Reliability Steering Committee

According to studies from the Network Reliability Steering Committee (NRSC), procedural errors were the root cause for 33% of reported service outages. The frequency of procedural outages has been on an upward trend, as shown in the figure on the right.

Industry analysis shows that people and process issues cause approximately 80%[4] of unplanned downtime, while the remainder can be attributed to product failures.

Human Factors in Unplanned Downtime

Humans are fallible and make errors. An often-quoted research report from the Gartner Group5 attributes up to 40% of unplanned downtime to operator errors alone. This includes the operators, the administrators, and everyone who physically touches the communications systems. The people making procedural errors tend to be semi-skilled personnel who are more familiar with hardware installation and cabling. Administrators, with a broader skill set, are expected to remotely handle the more complex tasks.

Service providers are well versed with these issues and design their networks with such considerations. They do not like to deal with complicated cabling and they like to be able to remotely diagnose problems from the safety of controlled environments – keeping as many hands off the actual system as possible. In addition, comprehensive training, certifications, and education programs can help increase technical knowledge and reduce basic human errors.

Reliance on human intervention can significantly increase the MTTR of a system. A person has to show up on site (which can not always be guaranteed), and human response time is

also usually far inferior to an automated recovery process. In addition, humans tend to make mistakes and can decrease the MTTF of other components in the system or significantly impede the MTTR of the failed component. Although system designers typically strive to remove the human element from the delivery of service as much as possible, they must determine up front if a human will be part of the fault management process in order to implement interfaces to minimize MTTR.

A *fifth* recommendation when designing a comprehensive availability strategy is to show that due consideration has been shown for human factors. As per the NRSC recommendation, high availability systems must strive to remove the human element from the delivery of service. If an error occurs, the system must be able to capture adequate diagnostic information and quickly return the system to service without waiting for human intervention. Not only does this prevent human errors, it also reduces labor costs by requiring fewer people and shifts. More tasks require fewer people and cheaper labor can operate the system.

Ensuring that your systems enable service providers to test newer versions of software while the system is running is another good way to reduce the possibility of human error. This testing allows them to cut over to new software easily. If a problem is detected with the new software version, the system can be rolled back to a known stable version of the software.

Managing Unplanned Downtime

In spite of the best components and the best quality control procedures, component faults are inevitable and both fault detection and fault repair impact MTTR. The rate at which faults can be detected directly affects the time it takes for a system to recover. If a backup component is available and is able to assume at least some of the failed component's responsibilities, a level of service availability is maintained. If the failed component has no backup or load sharing capabilities, then a service interruption occurs.

In order to properly manage unplanned downtime, a system must have a fault management plan. Fault management is typically a five-stage process, the tenets of which determine the efficiency of MTTR.

Detection — a fault is registered, but the failed component is not located

Diagnosis — the determination of which component has failed

Isolation — ensuring a fault does not cause a system failure. (Isolation does not necessarily make a system function correctly.)

Recovery — restoring system to expected behavior

Repair — restoring a system to full capability including all redundancy

Notification of the fault must occur at many points in this process. Examples of notification events include a change in system topology — when a board is taken out of service, put back in service, removed from the system, or inserted into the system. Between each of the five steps there must be notification to the next step or steps in the process. On fault detection, notification may occur to the diagnosis, isolation, and perhaps recovery software components simultaneously.

Perhaps a service provider's greatest need is for better visibility into the system. They require visibility in order to determine the health of the system, predict impending failures, and perform fault detection, diagnosis, isolation, and repair. Service providers need proactive indications when anything changes in the system beyond a certain threshold and are also asking for remote notification and alarm functionality.

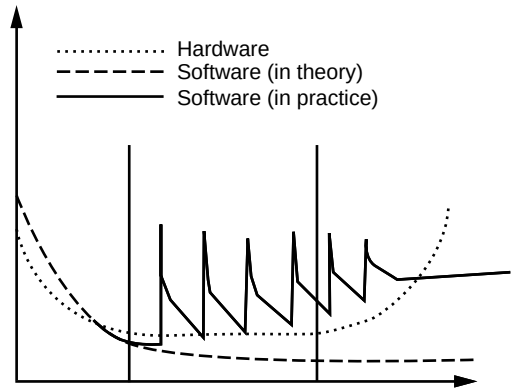
A final part of fault management is fault prediction. Fault prediction is an alternate form of fault detection, which includes built-in diagnosis. Based on predicted faults, the system operator can be given the opportunity to preemptively perform an online repair rather than wait for a fault to occur.

The *sixth* consideration for developing highly available systems is to select component suppliers that include enough hooks in their products so enough information is available to reconstruct the state of the system and fix issues with short turnaround times.

The Impact of Aging Components

Understanding the statistical rates at which different components fail over time can be a very powerful tool in designing systems for maximum availability.

Generally, hardware follows what is called the “bathtub curve” — failure rates drop off during the first few months a system is in service, but then increase again after a period of time. The initial drop off is due to “burn in”, where inferior components fail relatively quickly and are replaced. More stable components harmonize and quiesce in the system. After a period of time, they begin to degrade and ultimately fail. Hardware behavior is represented by the dotted line in the following graph.



Software, on the other hand, typically improves during the initial stages, due again to “burn in”, but does not exhibit the same pattern of degradation with age. Theoretically, software behavior is represented by the dashed line in the graph. Though in reality, software behaves like the solid line, showing peaks at times of upgrades or bug fixes. Ultimately, the software does stabilize and flattens out indefinitely.

A good axiom to remember — deployed hardware softens while deployed software hardens.

High Availability Configurations

The key to protecting against system level failures is redundancy. The type and amount of component redundancy determines the downtime characteristics of the system. This paper discusses eight different high availability architectures, points out their pros and cons, and identifies their availability characteristics. It must be noted that the key difference between the eight architectures is what is redundant and how the system recovers from a failure.

Clustering

In clustering, entire computers or systems are duplicated, so if/when a system in a cluster fails, the operations on that system are transferred to a spare system. The number of spare systems provisioned may vary from 2N where there is a spare system for each provisioned system to N+1 where there is a single spare for N systems. The spare systems may be deployed in an active/standby mode such that the spare standby is in a ready to go, but currently idle state. The more ambitious may employ active/active configurations; where all systems, including standbys, are in sync with each other’s activities and dynamic load sharing may be possible. Active/active configurations are harder to implement, but do provide

a financial payback if load sharing can be achieved such that when all systems are operating, the total system capacity is maximized and hardware is not just sitting idle waiting for a failure.

The advantages of clustering are that it works with any PC-based system, accommodates the PCI form factor, and uses standard network connections to keep systems informed of each other, and most importantly, it accommodates geographic diversity. In the case of a natural disaster such as a flood, fire, or an earthquake, clustering allows for continued service availability. Clustering also protects against service downtime caused by software failures. The disadvantages of clustering, as typically cited, include the duplication of costly peripherals and the relatively long fail over times (on the order of seconds as opposed to milliseconds for some other approaches). Resynchronizing systems after a failure — sometimes they have to be taken off line in order to restore a cluster to the necessary redundant state — is also a drawback to this architecture.

Hardware Fault Tolerance

Hardware fault tolerance is the replication of the CPU processing logic, which executes the same instruction set simultaneously and in lockstep[6].

The outputs from the replicated CPUs in fault tolerance are compared to determine if there is a difference in results. Since it is not possible to quickly and efficiently determine the errant CPU if there are 2 different results from 2 processors, typically, triple modular redundancy (TMR) is employed. TMR, which employs 3 processors, allows for a more effective fault isolation process. If the outputs from one CPU do not match the output from the other 2 CPUs, that CPU is considered errant and removed out of service and online repair can then take place.

The primary advantages of this scheme are protection against hardware malfunction with transparency at the application level. If hardware malfunction is detected on a set of components, those components can be replaced quickly and easily without any special failover logic required in the application level software. The user of the service does not notice any service degradation, even momentarily. On the other hand, this configuration does not protect against software bugs and failures. An errant software pointer can bring the entire replicated system down. Similarly, PCI implementations of these systems have trouble accommodating failures of interconnected media processing peripheral cards due to the limitations of the CT Bus ribbon cable. Systems that require these peripheral cards would also need to employ clustering or one of the cPCI architectures in addition to Fault Tolerance.

Peripheral Hot Swap and Redundancy

Peripheral hot swap (PHS) allows the online repair, upgrade, or addition of peripherals in a cPCI chassis, without the need to power down the entire system. Peripherals may be telephony boards, disk drives, fans, power supplies, management and alarm modules, and more. Peripheral hot swap can have a significant impact on reducing downtime, whether it be planned or unplanned.

While peripheral hot swap is effective in reducing the time to repair, it alone does not protect against operational downtime or the time taken to procure a spare and to dispatch a craftsperson to make the repairs. To protect against operational downtime, redundancy of peripherals is introduced. With peripheral redundancy, if a peripheral malfunctions, the spare peripheral takes over operations of the malfunctioning peripheral, without operator intervention. A craftsperson can then be dispatched, somewhat less urgently, to restore redundancy to the system.

Not only does PHS enable the replacement of failed components with minimal downtime, but it also allows for preventative maintenance. Peripheral redundancy can also let a service provider comfortably increase system capacity knowing there is a fallback should a fault occur.

Redundant System Slot

Redundant system slot (RSS) systems provide redundant, hot-swappable single board computers (SBCs) in a cPCI system. Such a system builds on the capabilities of a Peripheral Hot Swap cPCI system by eliminating the SBC as a point of failure.

Each SBC has a separate instance of the operating system and the application. The SBCs may be in an active/standby mode where the active SBC controls both cPCI bus segments in the chassis. If the active SBC goes down, the standby SBC takes over the processing of the failed SBC and assumes control of both the cPCI bus segments. In the active/active mode, both the SBCs are active and control their own bus segments. However, if one SBC goes down, the other SBC takes control of the bus segment controlled by the other SBC and operation of the system continues.

The primary advantage of RSS is the elimination of the SBC as a single point of failure and can be accomplished without the need for duplicating costly peripherals and extensive application changes. In addition, in order to bolster peripheral availability, it is possible to introduce peripheral redundancy with RSS, providing a very high level of system availability. On the negative side, depending on the restart mode selected on fail over, the reduction in restart time may not be significant. Also, the RSS standard (PICMG 2.13) has not been ratified and many cPCI platform vendors have proprietary and incompatible solution in the market-place today.

Cluster-in-a-Box (also called “Locked Bus”)

In the cluster-in-a-box (CIB) configuration, there are two or more logical systems in a cPCI chassis. Each logical system is a self-sufficient whole computer that contains its own independent cPCI and H.110 buses, its own SBC, peripheral cards, operating system, and applications. It is similar to combining Clustering and Peripheral Hot Swap in the same solution. Similar to multi-chassis clustering, if the SBC card goes down, the entire logical system goes down and it is not possible for the I/O cards in that system to be managed by an SBC card in another node. The systems within a chassis are independent of each other and only share the same card cage, power supplies and cooling.

The primary advantage of cluster in a box is the cPCI form factor that allows peripherals to be hot-swapped on failure. With respect to RSS and PHS, there are even fewer shared resources, so single points of failure are minimized. Due to collocation, the failover times may be better than that of multi-chassis clustered systems; however, collocation eliminates the advantage of geographic diversity.

Integrated Peripheral (also called “Computer in a Slot”)

Currently, integrated peripherals are cPCI form factor cards that include an embedded host processor, typically as a daughter board on a peripheral card. The embedded host processor daughter card hosts the operating environment comprised of operating system, the telephony drivers, libraries, APIs, and telephony application — functions similar to those performed by the SBCs in other configurations.

The benefits of an integrated peripheral are that it is a complete (host + peripheral), stand-alone computer in a slot. Each peripheral and its host processor are independent of other peripherals, which reside in the same chassis. When a fault occurs, it is isolated to the single peripheral card and only that peripheral and its host need to be restarted or replaced. The peripheral that is restarted does not have any effect on the other peripherals in the chassis. On the negative side, with no PCI or TDM bus for resource sharing, the resources on the card limit the application capabilities. In other words, each integrated peripheral card can only implement functions with the resource it has on its single card. Solutions that require multiple cards (i.e., one for fax, one for conferencing, etc.) cannot be easily implemented using this architecture. Further, each card needs a copy of the operating system, which can be costly and software vulnerabilities remain, although limited to a single card.

Packet Switched Backplane

Packet backplane configurations introduce a redundant high-speed packet bus right into the backplane of the system for high bandwidth traffic such as control, media or data. Such a backplane can replace and/or supplement the cPCI bus or TDM bus improving throughput as well as availability.

Packet switched backplane (PSB), as defined by PICMG 2.16, overlays a packet-based Ethernet architecture onto the cPCI backplane. Organized in a four-wire single- or eight-wire dual-redundant star topology, data is passed between nodes by routing IP packets to destinations, enabling a connection from each slot to each of the two redundant Ethernet switchboards. The system processor, PCI and TDM buses are then removed as single points of failure in the system. Other variants of packet backplane configurations have been proposed including StarFabric, InfiniBand*, and others.

On the other hand multiple, loosely coupled CPUs can be difficult to manage as a single system, and additional software may be needed for failover in dual Ethernet situations. Products that support this architecture are just now starting to enter the market, so the ecosystem may still be a little sparse when architecting complex solutions using this methodology.

Network Routing

Network routing is an effective high availability configuration that lets service outages be reduced in a very reliable manner as calls can be rerouted to entirely different installations. In addition, the network is partitioned into horizontal layers (similar to the OSI model) with service outage survivability built into the network at various layers including physical, system, logical, and service.

However, the techniques used in these layers may differ. Some of the techniques used to survive outages in the network include: reserve capacity, system diversity, geographic diversity, size limits, dynamic routing, restoration switching, self-healing protection switching, and others.

The network routing architecture is deployed widely today via the Intelligent Network (IN) build on top of SS7. It is also gaining momentum in the Internet buildout as it strives for higher and higher overall availability on par with the public switched telephone network (PSTN). This architecture has a lot of promise and is an area of continued research in the next generation network.

Theorems of Availability

Based on independent research, commonly accepted principals of availability, market rules of thumb and experience, this paper proposes the following theorems of availability.

- The key to high availability is redundancy
- Increasing component redundancy increases overall system availability
- In N+M component redundancy, as M increases, the increase in availability diminishes (i.e., N+1 is often most effective)
- Availability of a system is directly related to the availability of its components
- Decreasing MTTR also increases availability; examples include minimizing boot time, improving diagnostics, and enabling fast upgrades
- Overall, CompactPCI (cPCI) is more efficient than PCI in terms of handling operational downtime
- At any point in time, customers can migrate through a form factor to decrease downtime, commonly referred to as “buying back minutes”
- Over time software hardens, hardware softens
- System cost increases faster as availability increases
- Once a certain chassis size has been reached, density has little affect on the availability/cost ratio. As channels increase, most configurations scale linearly

Conclusion

The emphasis in the configurations discussed above has been on the availability characteristics of systems. However, the public network is made up of the aggregation of several of these systems. Further, in addition to the system availability characteristics, several other factors come into play in determining the availability characteristics of the entire network. The effects of natural disaster, terrorism, human error such as accidental fiber cable splicing, network congestion, and more must be understood and accounted for.

Network design plays a key role in the availability of the network. Techniques to detect failures and generate alarms are a critical first step in containing the duration of an outage. Beyond the detection phase, the diagnosis, isolation, recovery and repair phases are the critical next steps. As part of the recovery strategy, the network may be designed with additional capacity. If this additional capacity can continue to fulfill a user’s needs while repair is in progress, then an outage is not perceived.

Which high availability configuration is appropriate for a specific service provider is a question that requires additional analysis unique to the particular business model and IT infrastructure. This paper was designed to identify and compare and contrast the eight high availability configurations that can be implemented. The take away for the reader is to be able to make a more knowledgeable and informed choice on which path they elect to proceed.

For additional information or assistance in determining what configuration is most appropriate, and how to build specific high availability networks, contact an Intel® technical sales representative at 1-800-755-4444, and ask the operator for sales.

*Other names and brands may be claimed as the property of others.

1 “Sources of Failure in the Public Switched Telephone Network” by D. Richard Kuhn, National Institute of Standards and Technology; http://hissa.nist.gov/kuhn/pstn.html

2 For the purposes of this analysis, the Bellcore model was not considered suitable for determining system level availability since it assumes that the components are configured in series and does not take into account the effect of redundancies. RBDs provided a more sophisticated, analytical, and combinatorial form of reliability analysis.

3 A technique that provides approximate solutions to problems expressed mathematically. Using random numbers and trial and error, it repeatedly calculates the equations to arrive at a solution.

4 Gartner Group, 1999

5 Gartner Group, 1999

6 Source: Stratus Technologies; http://www.stratus.com

Intel Corporation
1515 Route Ten
Parsippany, NJ 07054
Phone: 1-973-993-3000
Fax: 1-973-993-3093

For more information
To learn more, visit our site on the World Wide Web at *www.intel.com*

Printed in the USA

00-7787-001 11/01

Printed on recycled paper.