# Table of Contents

Executive Summary .................................................................................................................... 3  
What is High Availability (HA)? ................................................................................................ 4  
1. Figure 1: Availability Equation...................................................................................... 4  
2. Figure 2: HA Systems.................................................................................................... 5  
   Service Disrupters ....................................................................................................................... 5  
   HA Middleware.................................................................................................................. ......... 6  
   3. Figure 3: HA Middleware .............................................................................................. 7  
Service Availability Forum and Standards ................................................................................. 7  
Design Considerations for Developing a Highly Available System ........................................... 8  
   1. Performance: How quickly does the system need to recover from a fault condition?.. 8  
   2. How important is preserving the state of the application data being processed?........... 8  
   3. What is the proper redundancy configuration to meet your goals?......................... 8  
   4. What is the required level of automation for the recovery process? ......................... 9  
   5. What are your requirements in choosing components for your system? ....................... 9  
   6. The final consideration: buy a COTS HA solution or build your own?....................... 9  
Recommendations ................................................................................................................ ....... 9
Executive Summary

This document is an introduction to the concepts of High Availability systems and will define High Availability, explore the potential causes of service disruptions and the role of HA middleware. Next, there is a discussion of an emerging open standard industry specification being developed by the Service Availability Forum. The document also lists a set of design considerations for developing a highly available system with a brief discussion of each. Finally, the document closes with a recommendation for further exploration.
**What is High Availability (HA)?**

Critical business services and systems must provide high levels of availability and reliability to their users. Developers are constantly creating new innovative solutions that the consumer becomes dependent upon in their daily life and will not tolerate downtime.

“High Availability” is a term that is often used, but not universally understood. This is an introductory paper meant to convey how GoAhead defines high availability. By agreeing on terms, we can better understand and solve our customers’ high availability needs.

Reliability is defined in terms of “mean time between failures”. This is a calculation made that is based on the expected life of the various components of a system. MTBF is a measure of how reliable a hardware product or component is. For most components, the measure is typically in thousands or even tens of thousands of hours between failures. For example, a hard disk drive may have a mean time between failures of 300,000 hours. A desired MTBF can be used as a quantifiable objective when designing a new product. The MTBF figure can be developed as the result of intensive testing, based on actual product experience, or predicted by analyzing known factors. The manufacturer may provide it as an index of a product’s or component’s reliability and, in some cases, to give customers an idea of how much service to plan for. The MTBF of a system is then calculated using a series of formulas to statistically combine the MTBF of the various components.

Mean Time To Repair (MTTR) is a measurement of the time required to bring a system back on line when a failure has occurred. Obviously, the user of the system is just as impacted by this number as the MTBF number. This is the actual downtime.

At the end of the day, availability of the desired service, or service availability, is really what the user is interested in. Availability is defined as the probability that service is available at any given instant, and can be expressed as a formula:

\[
\text{Availability} = \frac{\text{MTBF}}{\text{MTTR} + \text{MTBF}}
\]

Increasing the time between failures and decreasing the time required to make repairs to the system can therefore maximize availability. Another way of stating service availability is simply, *how much uptime is there per year of operation*. High Availability is often defined in terms of the number of nines. While 3 nines (99.9%) availability, or about 9 hours of downtime per year, might be acceptable for a desktop user, it is simply not acceptable for carrier switching applications where a level of 6 nines (99.999%) is expected.

One method for creating a highly available system is to provide redundancy. A backup system can take over for the failed system and avoid a service disruption to the user of the application. This is called a failover.
When discussing service availability, it’s important to understand the difference between a stateless and a stateful failover. In a stateless failover of an application, for example, the application is simply restarted – without any data restoration. If we use a word processor application as an example here, the application would restart but it would not have the data, or the document being worked on, restored. In a stateful failover, the data is checkpointed to the standby application and so the word processor would failover with the document remaining fully intact. It is clear that this is a requirement for any call processing equipment, billing processors, etc.

2. Figure 2: HA Systems

<table>
<thead>
<tr>
<th>Number of 9's</th>
<th>Downtime/Year</th>
<th>Typical Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>99.9%</td>
<td>~ 9 Hours</td>
<td>Desktop</td>
</tr>
<tr>
<td>99.99%</td>
<td>~ 1 Hour</td>
<td>Enterprise Server</td>
</tr>
<tr>
<td>99.999%</td>
<td>~ 5 minutes</td>
<td>Carrier Class Server</td>
</tr>
<tr>
<td>99.9999%</td>
<td>~30 seconds</td>
<td>Carrier Switch Equipment</td>
</tr>
</tbody>
</table>

High Availability systems are an attempt to create fault tolerant solutions that will be always available to users because, again, users expect and demand that their service is always available and uninterrupted.

**Service Disrupters**

Let’s take a look at what sorts of events cause a service disruption. We can categorize them and then shift our focus to reduce and eliminate as much as possible, those sources of downtime.

There are both planned and unplanned types of service disruptions. The tricky part is that both types count as downtime and therefore a plan is needed to minimize all disruptions.

In terms of planned service disruption, there are several events that can create a need for downtime. Occasional required changes to system configuration may necessitate some downtime. Upgrades to any piece of hardware, such as: I/O cards, hard drives, etc. may require a power down and reboot. Revision changes to software or applications of a patch to reduce bugs can also force a system reboot to complete the process.

Unplanned service disruptions have, unfortunately, more sources. The industry publishes data that identifies and quantifies these disruptions. A full 40% of unplanned service disruptions come from software and design problems: applications, operating systems, drivers and protocol stacks represent the bulk of these but other factors like network congestion and overload are the result of poor systems design. Another 30% of service disruptions are the result of hardware problems. These can be from the card (CPU, line, DSP, switch), the chassis (fan, power supply) or from storage devices. Finally, the
remaining 30% of service disruptions come from operator error: pulling the wrong card, removing a critical cable, etc.

There are also some causes of service disruption that are beyond the scope of HA system design. For example, malicious attacks and survivability (in the event of earthquakes, floods, fire, etc.) are other forms of service disruption that have more to do with system security and administration than with HA system design.

**HA Middleware**

The role of the HA Middleware is to provide a unified management framework for components, systems and clusters that can:

- Identify all active resources both hardware and software
- Assign active and standby roles to all resources
- Monitor the health of every resource
- Resolve any fault conditions via fail-over and recovery
- Maintain the application state during failures
- Manage remote access, control, configuration, etc.

By accomplishing each of these activities, the system can be maintained in an orderly and highly available way. The HA Middleware resides in a layer between the application and the operating system. From here, the middleware can monitor and in the event of a failure, manage the hardware, the operating system, the application and even itself. For further information on the role of middleware in highly available systems, please see the white paper entitled “Building Highly Available Systems: The Role of HA Middleware” located on the GoAhead web site [www.goahead.com](http://www.goahead.com).
3. Figure 3: HA Middleware

Service Availability Forum and Standards

It becomes quickly apparent that there is a need for an interface between the middleware and the application and between the middleware and the hardware/OS in order for the middleware to be able to control these components for availability. High availability solutions are traditionally based on proprietary hardware and software that can result in higher development costs, longer time to market and increased risk.

In an effort to address these issues and create an open standard for these interfaces, the Service Availability Forum was created to develop and promote industry-wide adoption of open standard programming interface specifications. The SA Forum was founded in 2001 and to date, the Forum has developed two interface specifications for carrier-grade platform and middleware applications that offer an open standard for the unification of carrier-grade network equipment elements. The Forum interfaces specify the information that flows between software entities and the semantics of that information, but do not concern themselves with the implementations on either side of the interface.

The SA Forum Application Interface Specification standardizes the interface between compliant High Availability middleware and service applications. The SA Forum Hardware Platform Interface specification provides a standardized interface between the middleware and the platform on which it runs. Use of these specifications enables the design of highly reliable systems without the limitations of proprietary interfaces and the cost to develop them. This results in shorter development cycles, cost savings, lower total cost of ownership, improved design flexibility, reduced development risk and faster innovation.

Today, the Forum is comprised of over 30 companies working towards development and implementation of these open specifications. For more information on the Service Availability Forum, please visit their web site at www.saforum.org
**Design Considerations for Developing a Highly Available System**

High availability middleware conducts vital activities to minimize downtime. In the past, developers had little choice but to write HA software themselves. This was often expensive and time consuming to develop and maintain. Today, commercial off-the-shelf (COTS) HA management middleware products exist to reduce development time, costs and risks. GoAhead SelfReliant is an example of a COTS solution.

Whether you decide to purchase or develop the HA middleware yourself, there are several design considerations that are useful to review. These six questions, compiled from GoAhead’s experience working with our customers, can help you address the common trade offs faced when creating a highly available system. Your answers to these questions will help to shape and determine the HA design.

1. **Performance:** How quickly does the system need to recover from a fault condition?

This is a key consideration and one that will drive much of the design process. There are three elements here to be considered: fault detection, notification and recovery. A time budget needs to be created for all three. Faults are detected by several methods, for example, a lack of response to heartbeating. In this case, the heartbeat frequency needs to be fast enough to fall within the detection time budget. The messaging performance of a distributed system will determine the notification speed and finally, the recovery time is determined by the configuration of the management service and system design (see question #3).

2. **How important is preserving the state of the application data being processed?**

There are three methods of state preservation that can be used based on the sensitivity of the application’s data. If the sensitivity to the data is low, then the transaction or application can be simply restarted and the data is not saved or preserved. If the sensitivity is medium, a disk based storage and retrieval method can be used. In this case it might take around a second to retrieve the data. If the sensitivity of the application to state data is very high, then the data should be continuously checkpointed to a standby system allowing initiation of a hot stateful failover within a few milliseconds.

3. **What is the proper redundancy configuration to meet your goals?**

The system needs to have design goals for failover time, scalability and cost. The HA system should have the flexibility to provide redundant configurations of 1+1 and N+M. 1+1 allows one active and one standby or an active/active configuration. The Active/Active policy assigns all members of the group the active role. It is common for resources participating in load balancing scenarios to apply this policy. The N + M policy allows you to configure the desired number of active resources (N) in the cluster as well as the desired number of standby resources (M) in the cluster. The N + M policy is frequently applied to hot-standby scenarios where a standby resource needs to be available to take over for a failed active resource without losing state information.

For example, consider a configuration in which N = 3 and M = 1. In this configuration the cluster will contain three resources actively providing service (role = active) and one resource with the role of standby (role = standby). Thus, if any of the active resources fail, the standby is available to replace the failed active resource.
The redundancy policy selected should also have the ability to scale as the demand for the application grows over time. The N+M policy is effective in allowing for considerable growth but this needs to be balanced against the design cost budget.

4. **What is the required level of automation for the recovery process?**

The design consideration for the HA system must take into account the level of automated recovery. Understanding and developing fault scenarios requires planning. The number and complexity of fault scenarios varies with each project. To develop an automated response to fault scenarios, a system model is developed which defines all system resources and defines logical dependencies of the resources to each other and creates redundancy groups. For example, if a credit card verification link should fail, the whole billing application should fail over to a redundant node. If the system model is built correctly, when the power supply fails, the whole system can be failed over to a redundant system.

A fully automated system must be able to follow all the steps of fault management: detection, location, isolation, reporting, repair, and reintegration.

5. **What are your requirements in choosing components for your system?**

It is beneficial for the HA system to have full control of the components in a system, for example, if a fan fails, it might be possible to speed up the remaining fans to properly cool the system. For this degree of flexibility, the hardware components each have a control requirement and require the use of HPI (the Service Availability Forum’s Hardware Platform Interface specification). The operating system, database, communication stacks and other software selected in the system should be HA aware and have features that enable HA management to function properly. There are application-ready platforms available, systems that have pre-integrated hardware, OS, database and HA middleware ready for a new application.

6. **The final consideration: buy a COTS HA solution or build your own?**

Just as a developer would not recommend developing an operating system today when there are a number of commercial systems available, it is no longer necessary to develop HA middleware internally. Flexible and capable solutions are available off the shelf and will save time, effort, cost and risk allowing valuable resources to focus on the application being developed. A further key advantage to commercial off the shelf (COTS) solutions is that they are flexible and reusable and require significantly less maintenance than an internally developed hardware specific solution. COTS solutions will continue to be enhanced to match industry hardware developments and the evolution of standards bodies.

**Recommendations**

Like any development project, proper and diligent planning most often precedes a successful outcome. For your highly available system, be sure to define your long-term requirements. Carefully consider the options and establish an architectural model. Consider the benefits of gradually implementing high availability over the course of the development project. Seek out and incorporate pre-integrated solutions where they are available and get assistance from proven partners.
Commercial off-the-shelf open-standard HA middleware solutions have become increasingly popular for common platform development teams as it can reduce time, cost and risk and provides greater flexibility over the life of the project.