the Availability Digest

www.availabilitydigest.com

High Availability Network Fundamentals

April 2009

Chris Oggerino, at the time of his authoring of High Availability Network Fundamentals,[1] was a Serviceability Design Engineer at Cisco Systems. His book is a practical guide to predicting network availability, complete with many examples using Cisco products in complex networks.

Though calculating availability of complex systems is a highly-mathematical exercise, Mr. Oggerino takes great pains to provide a detailed and accurate approach to availability analysis without requiring intensive mathematics – just a little simple arithmetic. There are only four equations in the entire book, yet he shows the reader how to do extensive availability analysis through a process that he dubs "divide and conquer." He shows the reader how to account for hardware and software faults, failover time, power outages, human error, and network device redundancy.

His book includes a CD containing the SHARC System Hardware Availability and Reliability Calculator spreadsheet. This spreadsheet does all the arithmetic work for the analyst as well as performs calculations deemed too complex for the reader.

Interestingly, his book follows the same path taken in our earlier Geek Corner articles entitled Calculating Availability – Heterogeneous Systems – Parts 1 through 4, *Availability Digest*; March through August, 2008. The primary difference is that we went into a great deal of mathematics behind the availability calculations – after all, we are writing for the geek audience. For those who found those articles a little daunting, Mr. Oggerino's book will be a refreshing and simple review of virtually the same topics.

Introduction to Availability

In his first three chapters, the author introduces availability concepts. He presents the *percentage method* and the *defects-per-million method* for describing availability. The percentage method states the percentage of time that a system is available (such as 99.9%, or three 9s). This is the method that we use throughout the *Availability Digest*. The defects-per-million method gives availability in terms of the number of failures per one-million hours.

Describing Availability

Knowing the MTBF (mean time between failures) and the MTTR (mean time to repair), one can convert between these methods. The percentage method is related to availability by noting that availability is the ratio of the time that the system is up (MTBF) to the total time (MTBF+MTTR):

[1] C. Oggerino, High Availability Network Fundamentals, *Cisco Press*; 2001.

© 2009 Sombers Associates, Inc., and W. H. Highleyman
www.availabilitydigest.com
For discussion, contact editor@availabilitydigest.com

$$\text{Availability} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}} \qquad (1)$$

The defects-per-million method is simply another way to specify MTBF. For instance, if a router has 200 defects per million hours, its MTBF is 5,000 hours. Knowing the MTTR then lets one determine the percentage availability.[2]

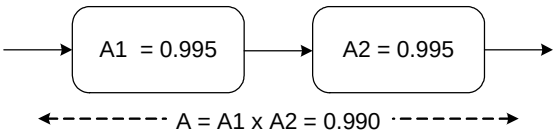
The defects-per-million method is typically used to characterize network device reliability by the manufacturer. However, the author converts these numbers to MTBF and uses the percentage method throughout the book.

Analyzing Availability of Network Configurations

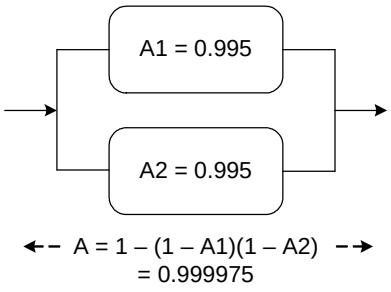
The author points out that MTBF (or the equivalent defects-per-million) is usually obtained from the device manufacturer. However, MTTR is a function of the maintenance policies of the customer. He suggests using the response time required by the service contract for MTTR. For instance, if the service contract calls for a repair technician to be on-site with a replacement part in four hours, MTTR is taken as four hours. The time to actually replace the component is ignored but is offset by the fact that many responses will be faster than that required by the service contract. This leads to a conservative estimate of MTTR.[3]

Using MTBF and MTTR to calculate the availability of a component, Mr. Oggerino next describes how to calculate the availability of a network of components. He notes that there are two basic configurations – serial and parallel.

In a serial configuration, all components must work. Should one fail, the network fails. In this case, the availability of the serial configuration is the product of the availabilities of the individual components. For instance, in a serial configuration of two components, if each component has an availability of 0.995, the availability of the serial configuration is 0995 x 0.995 = 0.990. The serial configuration is less available than the availability of any of its components.



In a parallel configuration, in the simplest case, the network will remain operational so long as at least one component is operational. In this case, the probability that the network will fail is the probability that all components fail, and this is the product of the failure probabilities of each component (the probability of the failure of a component is one minus its availability). The availability of the parallel network is one minus this value. Consider a parallel network comprising two components, each with an availability of 0.995. The availability of this configuration is $[1 – (1 – 0.995)^2] = 0.999975$. The availability of the parallel configuration is significantly greater than the availability of any of its components. This is redundancy. As we know, adding redundant components significantly improves the availability of a network.



[2] The author distinguishes between mean time *between* failures, MTBF, and mean time *to* failure, MTTF, in his Figure 1-3. Availability is MTTF/(MTTF+MTR) or (MTBF-MTR)/MTBF. However, his point is that either can be used in Equation (1) above with negligible difference. He uses MTBF in the book in place of MTTF in the MTTF equation. You can interpret MTBF as mean time *between* failures, equivalent to MTTF.

[3] This assumes, of course, that the device in failure has been identified.

© 2009 Sombers Associates, Inc., and W. H. Highleyman
www.availabilitydigest.com
For discussion, contact editor@availabilitydigest.com

The more complex case is N+M redundancy, in which N components are required and in which there are M spares. Though the author gives the equation for the most common case of N+1, he doesn't use it in his later examples, relying instead on the SHARC spreadsheet to make these calculations.

This section concludes with availability analyses of some serial, parallel, and serial/parallel network topologies using Cisco components.

Predicting Availability

The next two chapters focus on how to predict the availability of network devices and topologies.

Availability Factors

The author begins this topic by discussing the various factors that can affect availability and shows how to account for them. Factors include:

- hardware
- software
- environment
- human error
- network design

Hardware

Calculating hardware availability according to Equation (1) requires knowledge of the MTBF and MTTR of each device in the network. MTBF is determined by most telecommunications companies via the Bellcore TR-232 specification, which takes into account the known failure rate of every elemental component in the device being measured – integrated circuits, resistors, capacitors, fuses, transformers, etc. – as a function of temperature and other stresses. From this, the *failure-per-ten-billion-hours (FIT)* for the device is determined and can easily be converted to MTBF.[4]

So far as MTTR is concerned, the author suggests that it be assumed that the actual time to replace a failed device is short. It is the time to get a service technician on-site with the replacement device that is the time-consuming part of device repair. Therefore, using the service-contract response time (4 hours, 24 hours, or whatever) is a conservative way to estimate MTTR for hardware failures.

Examples of calculating the availability of simple nonredundant devices, redundant devices, and network segments are given.

Software

The determination of software MTBF requires the measurement of software failures for a large number of devices over a long period of time. Cisco has, in fact, done this and has determined that a conservative value for software MTBF is 30,000 hours for mature software and 10,000 hours for software that has been recently (within a year) released. Actual measurements of some large networks showed software MTBFs of 45,000 to 70,000 hours for mature software, so the suggested results are truly conservative.

[4] For instance, if a capacitor has a failure rate of 1,000 failures every ten-billion hours, it will have an FIT of 1,000. Adding the FITs for each component in the device gives the FIT for the device. If the device has a total FIT of 1,000,000, it will fail once every 10,000 hours, which is its MTBF.

© 2009 Sombers Associates, Inc., and W. H. Highleyman
www.availabilitydigest.com
For discussion, contact editor@availabilitydigest.com

Since most software faults are cured by rebooting the device, software MTTR can be taken as the reboot time. Averaged over router sizes, the author suggests using six minutes for software MTTR. Larger routers will take longer as they recompute routes.

Examples of calculating availability, including software failures for a simple device and a redundant device, are given.

Environment

Environmental faults include power outages, hurricanes, earthquakes, tornadoes, forest fires, and other events. Most cannot be predicted and are often catastrophic, putting data centers out of service for days or more.

However, these events all tend to disrupt the power source to the network. Therefore, the author focuses on power outages. He suggests that in the absence of better data, a national average seems to be about 29 minutes of power loss per year. This is an availability of 0.999945 (a little over four 9s). The effects of power loss can be mitigated by backup power devices such as battery or diesel power backup, which themselves may fail.

Several examples of network availability with different power-loss mitigation strategies are given.

Human Error

Mr. Oggerino points out that human error is by far the predominant factor in network availability. Human error typically occurs when standard operating procedures allow it to happen. Operational processes that mitigate human error are fundamental to highly available networks.

Human errors most frequently occur as a result of changes. Adding, upgrading, and reconfiguring network devices can be a very complex process. As the author states, "Nearly every human error that occurs will be more than the downtime for all other reasons combined. … Once enterprise networks are operational and the bugs have been worked out, they hardly ever fail. When they do have problems, it is just as likely to be a result of someone changing something as it is for any other reason. … We assume a human-error contribution of four hours every other year."[5]

A massive communications failure that occurred in 1997 is described as an extreme example. A large network in the United States was down for 24 hours because network administrators decided to simultaneously upgrade the software in two major switches that backed each other up. Even worse, they had no failback plan. On reboot, the switches did not perform properly; and it took a day to correct the problem.

A more common problem that is discussed is duplicating IP addresses when adding PCs to a network. Improper security control reconfigurations are another common problem.

As noted above, the author suggests reasonable values for human error to be one failure every two years (MTBF) that takes down the system for an average of four hours (MTTR). This leads to an availability due to human error of 0.99977, a factor an order of magnitude more dominant than power loss, as described above.

He suggests a procedure for measuring and minimizing human error. Good change control procedures are key to improving this availability factor.

[5] Page 171, Human Error and Process Contribution to Downtime in an Enterprise Network, *High Availability Network Fundamentals.*

© 2009 Sombers Associates, Inc., and W. H. Highleyman
www.availabilitydigest.com
For discussion, contact editor@availabilitydigest.com

Network Design

The incorporation of redundancy into the network configuration is a key to achieving high availability. Redundancy means the provision of two or more devices that can provide the same functionality. If one should fail, the other can take over and keep the network in operation.

There are two primary forms of redundancy – standby and load sharing. With a standby backup, the standby device is not operational. Should the primary device fail, the standby is pressed into service to take over its functions.

In a load-sharing configuration, two or more devices are providing the same function - sharing the load. Should a device fail, traffic is simply routed to the surviving devices.

In terms of the availability calculation, redundant systems are parallel systems that provide significant additional availability for that device. However, they add their own form of additional downtime that must be considered. That is failover time. If the network will be down for one minute as the standby device is brought into service, this minute adds to the total downtime of the network.

Failover time is a function of the router protocol. The author suggests using a failover time of 35 seconds for OSPF (Open Shortest Path First)[6] and five seconds for HSRP (Hot Standby Router Protocol, a Cisco proprietary protocol similar to the industry standard VRRP[7]). In the book's examples, it is clear that failover time is the predominant factor in redundant-device availability – often by a factor of 100.

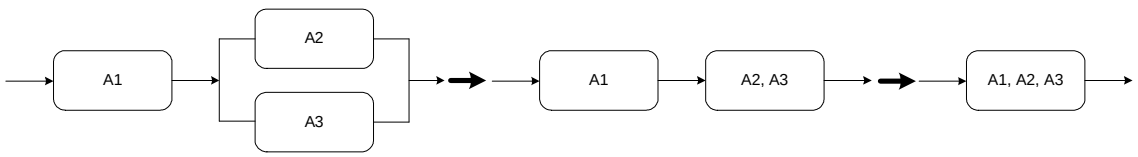
There is also the possibility of a failover fault, in which the failover doesn't work and the network is down until a repair can be made. The author ignores this as it is beyond the scope of the book.[8]

Several examples are given for incorporating redundant systems and failover times into the availability calculation.

Divide and Conquer

Having described the various factors that go into the calculation of device availability and simple serial and parallel configurations of devices, the author introduces the concept of "divide and conquer" to analyze complex networks. The first step is to diagram the network with a *reliability block diagram (RBD)*. This diagram shows the connections between each of the devices in the network and clearly shows the parallel and serial constructs.

The next step is then to calculate the availability of all parallel constructs and to replace each with an equivalent device with the availability of the redundant devices. Serial constructs are then replaced with a single device with the availability of the serial devices. If this process should create additional parallel constructs, the process is repeated until the entire network is represented as a single device with a known availability. This is the availability of the network.



[6] OSPF is the most widely used internal gateway protocol in large enterprise networks.
[7] VRRP – Virtual Router Redundancy Protocol, *Availability Digest*; October 2008.
[8] Calculating Availability – Failover Faults, *Availability Digest*; March 2007.

© 2009 Sombers Associates, Inc., and W. H. Highleyman
www.availabilitydigest.com
For discussion, contact editor@availabilitydigest.com

The book then introduces the concept of network scenarios by way of an example that includes interconnected VoIP (Voice over IP) networks in two buildings that also interface to the public switched telephone network. A network scenario is a path through the network whose availability is desired. For instance, in this case, scenarios include connecting two users in the same building, connecting two users in different buildings, and connecting a user to the public telephone network. Each scenario will have a different availability. The author takes the reader through the detailed availability calculation for each scenario.

Real-World Case Studies

The final four chapters use the techniques taught in the book to calculate the availability of several actual networks. The first chapter leads off the effort by calculating the availability of certain Cisco routers.

The first case study is that of a small ISP network. The second case study is for an enterprise network. The final case study is the availability analysis of a very large and complex VoIP network.

In each case, all availability factors are considered – hardware failures, software faults, power outages, human errors, and network device redundancy. These chapters make extensive use of the SHARC spreadsheet included on a CD with the book. The SHARC spreadsheet results are all included as figures.

The SHARC spreadsheet is described in detail in an Appendix.

Summary

High Availability Network Fundamentals is a very readable discussion on how to calculate the availability of very complex networks. The only mathematical knowledge required is simple arithmetic. Calculations both simple and complex are provided by a spreadsheet provided on a CD with the book.

The book's technique includes the primary contributors to network failure – hardware, software, power, and people. A simple diagramming technique using Reliability Block Diagrams (RBDs) guides the analyst through the decomposition of complex networks into bite-sized chunks for arithmetic analysis (nothing more than addition and multiplication).

Perhaps the most daunting task for an analyst is understanding the network in enough detail to create an accurate RBD. The next most difficult task is determining the availability of each of the components in the RBD. This may require conservative guesstimates for such factors as power failures and human errors. Once these hurdles have been overcome, the actual calculation of network availability, even for very complex networks, is reduced to a series of very simple steps by the divide-and-conquer techniques taught in the book.

Though the book focuses on network availability, its techniques are directly applicable to redundant systems such as clusters, primary/standby configurations, and active/active systems.

© 2009 Sombers Associates, Inc., and W. H. Highleyman
www.availabilitydigest.com
For discussion, contact editor@availabilitydigest.com