

Network Reliability Testing

Overview

Today's networks are large and complex beasts that require constant attention and management. There are hundreds, if not thousands, of tools aimed at the network management space. While some are very good at what they do, they all miss one key point—testing. The software industry is very sensitive to application testing and making sure that applications are both reliable and secure in a stable atmosphere. Applications in the wild, however, are subject to myriad activities that can break or otherwise hobble them.

Most network management systems (NMS) do a fine job in relatively stable environments, but when real emergencies occur, several limitations are exposed. Complex rules meant to help an environment recover from crashes may fail due to age and missing system dependencies; or perhaps a failover target is unavailable for one reason or another. Usually it's the simple things that cause the biggest problems—flaws that structured testing would uncover quickly. And therein lies the problem. The notion of deep and structured testing so familiar to the software development world is foreign to most IT managers. Though they spend a great deal of time making sure that systems have backups and that data flows reliably, their efforts are inevitably thwarted by inconsistencies that creep in over time.

What network managers need is a tool or system that, in conjunction with the NMS, forces failures across the network to live-test the rules and dependencies. Just as battlestation drills on warships ensure that systems are working and all personnel know what to do in a crisis, a regular live-test of network management systems would help ensure that critical network components and procedures do the right thing and are ready for any emergency.

Network Management

Network management is typically divided into two sections: one for monitoring and the other for active management. WhatsUp Professional™ from Ipswitch is a good example of a tool that provides monitoring and some basic management capabilities. The tool actively scans networks for server, application and service availability. When one of the monitored items doesn't respond as expected, WhatsUp issues an alert to the network manager and executes simple actions, such as restarting a service, to try to recover the system. WhatsUp Professional's management strategy works quite well in smaller networks where administrators can keep the whole network in their heads and physically touch each of the systems. However, in larger, hierarchical or distributed systems typical of those found in today's enterprises, such a tool simply cannot deal with the demands.

Larger systems need sophisticated automation that manages complex recovery scenarios, service ticket origination and reporting. HP OpenView is a good example of a sophisticated tool that is capable of managing large networks and executing the actions necessary to keep networks running and the managers informed.

Managing large, heterogeneous networks that provide multiple services requires specific knowledge of each of those services. Systems like HP OpenView rely heavily on third party vendors to provide those monitoring and management systems, which leads to varying depths of support, inconsistent user interfaces, different scripting methodologies and notation—in a word—complexity. Managing systems like this can be effective in the near term, but changes in either the monitored service or the monitoring software can rapidly render the management system useless, and these do inevitably change over time. Knowing when a rule or recovery action is broken is never simple, and flaws often come to light only during the very emergency against which the management system is supposed to protect. The trouble is that these failures are really nobody's fault. Network managers are expected to manage systems with huge numbers of moving parts and to keep them rolling smoothly. The complexity presented to them by the network management system does not help them manage the

dynamic growth of the system, it just provides them with another thing that they need to learn and manage. What managers really need is a way to tell whether their system is set up correctly and will do its job when disaster strikes.

Network Testing

For network managers, it would be helpful to simulate disaster scenarios in order to force the network management tool to execute the responding action. Hopefully the NMS will work properly, but if not then the network manager has an immediate opportunity to address the issue. Software testers refer to this kind of testing as “fault injection,” and are beginning to employ it as a matter of course in the software development lifecycle. When they test applications, they use fault injectors to simulate flawed environments in which they can observe the results. Faults can be anything from simulated broken disk drives and network controllers to corrupted configuration files and registry keys. Each fault is designed to drive an application down a specific error path, and if there’s insufficient error handling on that path, the application crashes or becomes otherwise unstable. This gives the development organization the opportunity to address the fault before the application is deployed.

From this standpoint, the parallels between application testing in the software development lifecycle and the deployed server in the network are obvious. In both cases, fault injection-based testing can expose deficits in error handling and protect the organization concerned from the pain that results from failure. In the case of the development organization, testing helps them to deliver more robust and secure applications to their customers, and build a reputation for quality. In the case of the network environment, preventative testing would help them avoid down-time and revenue loss resulting from an unexpected service or server failure, such as the transaction management server in an ecommerce organization.

Conclusion

Imagine a simple fault injection test scenario executed monthly during a [relatively] quiet time against all the servers, services and application on the network. Would all systems recover every time? Would it be valuable to know on a regular basis which systems the network management system can recover and which it cannot? Would network managers be more effective if they were required to execute such a reliability drill as part of their normal management process?

The answers to these questions will of course depend on the organization. In general, should network managers apply such testing, the quality and reliability of any network will absolutely improve.