Realtime publishers

The Shortcut Guide™ To

Assuring Website Performance through External Web Monitoring

sponsored by

neustar™

Dan Sullivan

Copyright Statement

Chapter 2: Examining Essential Components of External Web Monitoring

The confluence of increasing expectations for continuous service delivery and the growing complexity of Web applications has created a need for robust application performance monitoring. In the past, monitoring was primarily an internal operation with network performance monitoring checking volumes of network traffic and the availability of servers.

For example, when most mission-critical applications ran on mainframes, in-house IT professionals could monitor the state of the mainframe from management consoles using performance management tools provided by the mainframe vendor. As applications become distributed with client-server applications, performance became a function of several machines sometimes running different operating systems (OSs). Ensuring application performance on those systems required the ability to monitor servers that might run Unix, Linux, or a Windows server OS as well as client devices, typically running a Windows desktop OS. Moving applications to the Web made performance monitoring significantly more difficult.

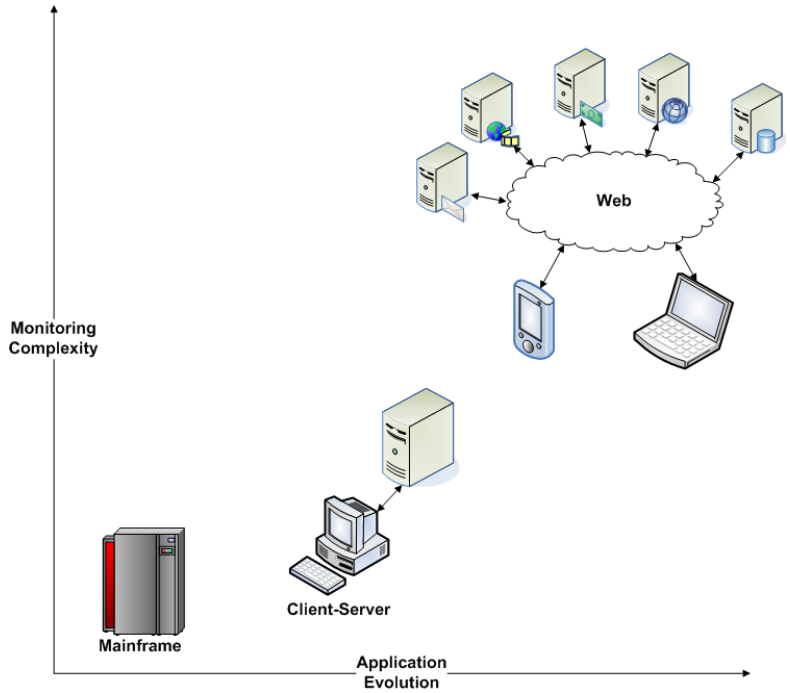

Figure 2.1: Monitoring complexity has increased with the evolution of applications toward more distributed systems.

Web applications can rely on multiple servers, possibly in several locations and maintained by different groups. To add to that complexity, end users could be located around the globe running the user interface (UI) portion of applications in different browsers on platforms ranging from desktop clients to handheld mobile devices.

In this chapter, we turn our attention to understanding the technical aspects of Web site monitoring, including:

- Key components of external monitoring
- Key types of monitoring
- Analysis methods and response procedures
- Implementation testing

We begin with the building blocks of an external monitoring system.

Key Components of External Monitoring

Any type of application monitoring solution will require three basic components to capture data and provide information to application administrators:

- Monitoring agents
- Application-specific monitoring tests
- Reporting and altering systems

External Web application monitoring requires specialized monitoring agents and application tests while reporting and alerting systems are similar to other monitoring systems.

Monitoring Agents

Monitoring agents can be either internal or external. Internal agents reside within a corporate network and provide data on the state of the network and applications relative to the internal network.

The reach of Web applications extends beyond the corporate network to include the networks used by business partners, customers, and to the global Internet at large. Internal monitoring, such as that depicted in Figure 2.2, is insufficient to monitor performance on external networks. External monitoring addresses this problem with geographically distributed agents.

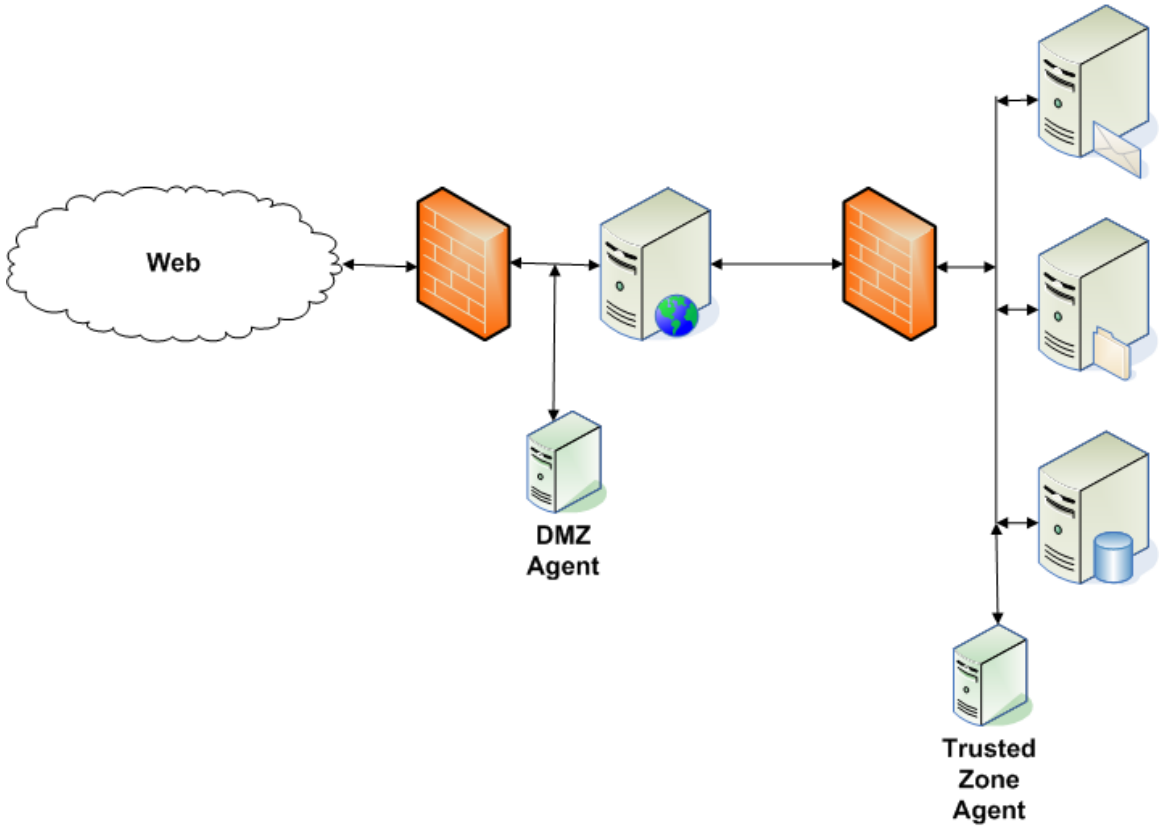

Figure 2.2: Monitoring agents are limited in the scope of their observations according to the network traffic on the segment they monitor

Geographically distributed agents can exist anywhere on the Internet. For example, an external monitoring service provider may have agents distributed across the globe. This setup allows the service provider to monitor applications from around the world and to provide information on the way customers in different parts of the world experience the performance of an application.

Let's consider a hypothetical example. A tool manufacturer maintains a corporate headquarters in the United States, a manufacturing facility in China, a support center in India, and sales offices in the United Kingdom, Belgium, United Arab Emirates, China, and Australia. The sales force depends on the corporate enterprise resource planning (ERP) system to place orders, check inventory, and follow up on customer inquiries. This is a distributed system with the following types of servers:

- A database server for persistent storage of ERP data
- An application server for running ERP software
- A Web server for providing the Web interface
- An LDAP directory used for authentication
- Several Web services provided by third parties for ancillary tasks, such as calculating local tax rates for customer orders

These can be monitored internally but that would not reflect the performance experienced by sales personnel around the globe. A better method is to use external monitoring with agents in each of the regions that supports a substantial number of users.

Figure 2.3: Relying on internal monitoring cannot provide a full picture of the state of Web application performance when users are geographically distributed.

Application-Specific Monitoring Tests

External agents can perform multiple types of performance tests. Three types, in increasing order of complexity, are:

- Simple connectivity tests
- Basic service tests
- Application-specific tests

Simple Connectivity Tests

Simple connectivity tests monitor the ability of the network to route traffic between two points on the network. The ping utility in TCP/IP is probably the best-known connectivity test. Ping sends an echo request packet to a target device and waits for response packets. The test provides data on the average round-trip time to receive the response and the number of packets lost, if any.

Simple connectivity tests are useful for verifying low-level connectivity services, such as the ability to route a packet to the target server. Simple connectivity tests show the OS is up and functioning enough to support TCP/IP. It does not, however, provide any details on the status of applications running on that server.

Basic Service Tests

The next level up from simple connectivity testing is testing basic services, such as:

- Database servers
- Application servers
- Directory servers
- File transfer servers
- Content management servers
- Web servers

These, and similar services, are the building blocks of many Web applications. These services are often thought of in terms of a stack with databases at the bottom, application servers along with specialized servers (for example, file transfer servers) in the middle, and Web servers on the top closest to client devices.

Each of these may require specialized tests to monitor their performance. For example, to test database performance, an agent may connect to the database, issue a query, and validate the results. For an application server, a Web service may be invoked and the resulting XML results checked for accuracy. Similarly, a file transfer server can be tested by transferring a test file and reading it back again. Web servers can be easily tested by simulating a browser and navigating to pages served by that Web server.

Successful service tests indicate the components above the OS are functioning. The next level of testing is at the integration level.

Application-Specific Tests

Application-specific tests are designed to monitor the functionality of application components working together. Monitoring individual components is helpful and can help isolate problems with applications, but on its own, it does not provide sufficient information to know whether applications are functioning correctly.

Consider another example. A customer support Web site provides access to customers' information about order status, shipping details, invoices, and so on. The application includes a Web server, an application server running a Java application, and a relational database on the backend. All the components are up and running according to the basic service monitoring tests; however, the application is not functioning correctly. When a user enters an order number into the UI, instead of getting order details, the customer sees an error message from the database about a corrupt index. The components are working individually (at least to some degree), but the application fails.

This type of problem arises because basic services tests can perform only so many checks to validate the function of a service. In the case of the corrupt index, it is true the database server is working fine, but one of the data structures in it is invalid. There are two options for addressing situations such as this:

- Try to think of all possible ways a database server could generate an error and test for those, and
- Test the application as if a monitoring agent were a customer executing tasks available from the UI

The first option is impractical to say the least, while the second option helps to monitor the application from the users' perspective.

Simple tests, basic service tests, and application-specific tests all provide valuable information to application administrators, systems managers, and network managers. The last key component of external monitoring is the set of reporting tools that take that data and turn it into actionable information for IT professionals responsible for application performance.

Reporting and Alerting Tools

Agents execute test scripts that generate data on application performance. As one can imagine, even a modest number of agents running a minimal number of scripts can generate a great deal of detailed data. To avoid information overload, reports can be organized to provide several types of information presentations:

- Summary data
- Drill to detail
- Alerts
- Trend analysis and early warning

Summary data aggregates data into useful pieces of information. For example, rather than report in detail each of the pings on a server, each Web services test, and task simulated in the browser, summary data provides a high-level overview. These might include average times, number of tests, and number of failures by category of test.

There are times when details are helpful, such as when the average response time to a database query is significantly longer than expected. Is that due to many of the tests requiring longer than usual time to complete? Or are some queries run from remote locations taking significantly longer than others and thus skewing the results? In situations such as this, it is useful to have drill to detail features in reports.

Alerts are another useful form of reporting when an event requires immediate attention. For example, if a database server is down or a high number of response times for application tests exceed some threshold, a systems administrator may want to be immediately notified.

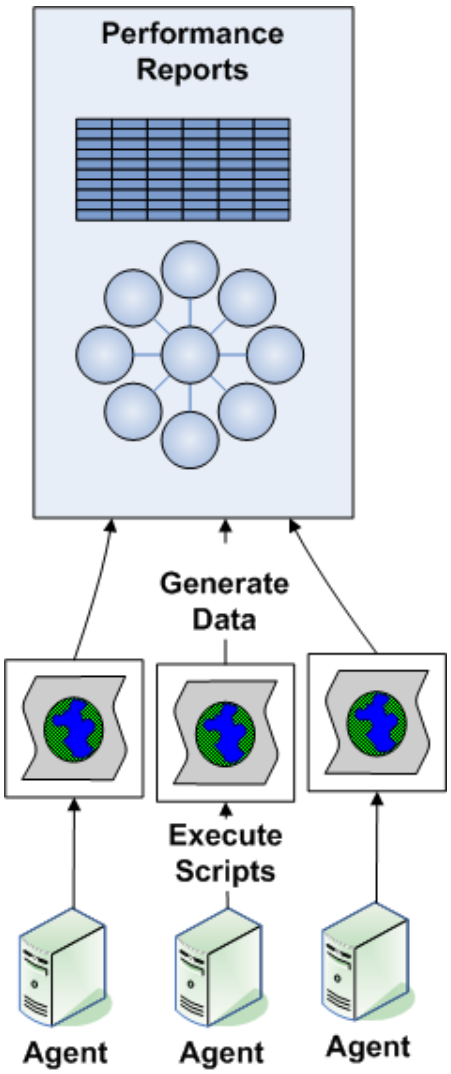

Figure 2.4: Agents execute scripts that generate data that populates reports on the state of Web application performance.

Trend analysis is useful for long-term planning. With these reports, the goal is to understand any long-term changes in performance. For example, is there a steady increase in average response time to a particular UI task? Is the volume of network traffic to the application server increasing, perhaps indicating a need to modify the UI to reduce traffic or to add application servers in a load-balancing configuration? Trend analysis reports are also useful at budget review time when one needs to justify additional resources to maintain performance levels.

To summarize, the key components of external Web application monitoring are geographically distributed agents that execute a variety of tests (connectivity, basic service, and application), and reports and alerts. With these components in place, you can perform a wide variety of types of monitoring.

Key Types of Monitoring

Monitoring Web sites is much more than ensuring users do not receive a 404 error (that is, a page not found error). Web applications are complex assemblies of multiple layers of programs providing different types of services. Each of these requires its own particular type of monitoring to ensure that the service is not only responsive but also functioning as expected. This section will consider a full range of monitoring types:

- Web site monitoring
- Application server and transaction monitoring
- Web services monitoring
- Streaming audio and video monitoring
- Mobile monitoring
- Database monitoring
- Network monitoring
- Port monitoring

This is a fairly substantial list that reflects the diversity of Web application components that must be checked with an external monitoring system to ensure adequate application performance.

Web Site Monitoring

Web site monitoring requires several types of checks. The most basic is that the Web server is up and running. When a user browses to http://www.my-company-xyz.com, for example, it should return something other than a missing page or a timeout error. If you have that level of functionality, you can move to the next stage of Web site testing.

Web sites can have hundreds or thousands of pages. Some are static and some are dynamically generated; both types should be checked by monitoring scripts. A common technique is to check the top-level entry pages of a Web site along with other popular pages. Web pages are checked, first, to make sure they are available and, second, to measure the time to retrieve the page.

Performance monitoring is especially important for pages that could disrupt a transaction if performance is slow. If looking up product details or stepping through a checkout process is too slow, customers may abandon their carts before completing the sale. One could regularly execute a monitor script that checks a list of products in a catalog site to detect performance issues with those pages. Some things to keep in mind with Web site monitoring:

- Certain links on the Web site may lead to external Web sites. It may be useful to verify these links even if one cannot solve particular problems with those sites.
- If a script is designed to follow all links in a page, use a history mechanism to record which pages have been visited. Web pages can have links that, if followed long enough, loop back to the starting page.
- If different features are available to authenticated users than to anonymous users, test both sets of features.
- Pages that generate content dynamically should be tested for Web site access, but the underlying application that generates the content should be tested as well.

The last bullet point leads to the next topic: application server testing.

Application Server and Transaction Monitoring

Web applications are often built in multiple tiers. As Web application users, we are most familiar with the client-side components that operate in the browser. Like an iceberg, much of the functionality of a Web application lies "below," that is, on a server. Sometimes this server-side processing occurs on the Web server, and in other cases, there are separate application servers that run supporting applications. Regardless of where this additional processing occurs, it is essential to monitor critical transactions such as login, checkout, and search. We just considered monitoring issues with regards to Web server applications, so we will now turn our attention to application server monitoring.

Application servers are systems that run on servers and provide a framework for running applications fairly independently of the underlying OS. A Java program, for example, may run in a Java application server that is executing on a Windows server in much the same way it would run on a Java application server running on a Linux distribution. The Web applications running on these application servers could be as simple as a weather forecast report, or complex applications, such as an enterprise portal.

When testing application servers, you should ensure basic responsiveness and the ability to execute transactions. When monitoring application servers, you should test for the ability to:

- Authenticate against the service, if logins are required
- Execute functions provided by the application, such as checking out an order
- Complete transactions in a reasonable amount of time (the time will vary by transaction type)
- Navigate to various pages generated by the application as expected based on privileges granted to the user

Key measures to watch for with application servers are response time and the ability to complete transactions.

Web Services Monitoring

The widespread adoption of Web services was a major step forward for distributed systems development. Web services expose functions as services to other applications using well-defined interfaces. One of the best known is the Simple Object Access Protocol (SOAP), which allows for exchange of messages between services and clients over HTTP and other protocols. Messages are specified in XML and bundled in a SOAP "envelope," routed to the Web service, which extracts information from the message, executes the appropriate code, encodes the results in another XML message, and sends it back to the caller. As this list shows, there are several points at which a Web service can fail. When testing Web services, scripts should be configured to test:

- A range of parameters passed into the Web service
- With and without optional parameters
- Properly formatted results
- Accurate results
- A WSDL registry, if one is provided (this is essentially a directory of Web services)

In addition to generic Web services testing, external monitoring also provides media-specific testing.

Streaming Audio and Video Monitoring

Audio and video can be streamed using multiple protocols. The Real Time Streaming Protocol (RTSP), for example, is used to establish streaming media sessions. The Real-time Transport Protocol (RTP) is typically used for transmitting content.

Unicast and multicast are two ways of transmitting video over the Internet. Unicast requires a connection for each user receiving a video stream. This protocol allows greater user control but at a cost of higher bandwidth consumption because each recipient has a separate connection. Multicast routing sends a single stream of media to multiple recipients. This protocol requires properly configured routers to support the protocol, which is more like broadcasting and is more practical for internal use than across the Internet.

From a testing perspective, audio and video monitoring should test for:

- Ability to stream content at sufficient speeds to meet user expectations
- Reliability of connections so that sessions do not have to be reestablished and media streamed again from the beginning
- Assurance that audio and video play correctly in commonly used browsers
- Support of appropriate media players

Content such as audio and video can go to multiple types of devices, so you should also consider monitoring on different platform types.

Mobile Monitoring

The goal of mobile monitoring is to ensure content is delivered correctly to mobile devices. The growing popularity of smartphones with full-featured browsers and high-speed connections will likely promote the use of Web applications using handheld devices. External monitoring of mobile devices may include several types of tests:

- Ability to display content in mobile browsers
- Ability to execute application functions from a mobile browser
- Performance metrics, such as the time to download content to a mobile device

Database Monitoring

Databases are often the lowest level of services in an application stack. The UI, application servers, Web services, content management systems, and other services often depend on a database for persistent storage of various types of data. Often, higher-level services will fail to function properly if a database is down, but they may not fail completely.

Software engineers have developed ways to "gracefully degrade," which is a euphemism for not crashing all at once. Basically, the idea behind this concept is that a complex system, like a distributed Web application, can have failures isolated to particular components and still continue to function with a reduced set of services. For example, a failure in a database storing content management system data may leave a Web site without dynamically generated content but with some statically managed content. Similarly, if an application depends on multiple databases and one fails, the services supported only by the failed database would be unavailable.

Gracefully degrading database applications requires you to monitor databases directly. Just because a Web site is up does not mean the database is functioning properly. It is not uncommon following a database failure to see Web pages displaying arcane database error messages or Java stack dumps that end with errors in JDBC connections to the failed database.

When performing external monitoring of databases, scripts should check for:

- Basic connectivity with the database listener
- Ability to authenticate to the database
- Ability to execute read queries
- Ability to update the database

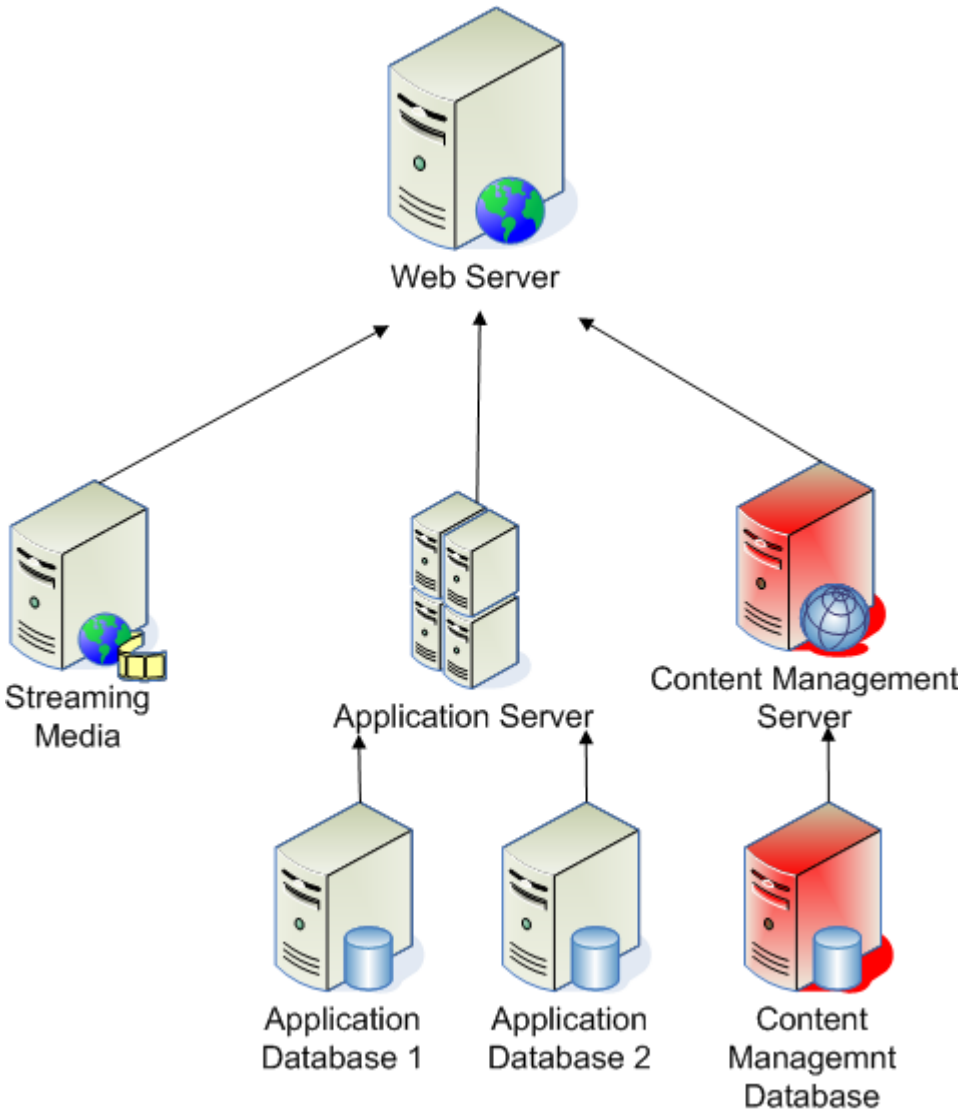

Figure 2.5: A database failure may disrupt a limited set of services while other Web application services continue to function.

Network Monitoring

Network monitoring is a complex task because there are so many protocols, routes, and performance metrics to collect. The goal of network monitoring is to ensure the efficient routing of all data between nodes on the network. This includes preserving the availability of network services by efficiently routing traffic to avoid bottlenecks, blocking disruptive traffic (for example, Denial of Service—DoS—attacks), and checking for unauthorized traffic (for example, ensuring there is no ftp traffic when policy calls for disabling all ftp traffic).

Failures or problems with other types of external monitoring can be indicative of a failure at the network level. Sometimes it is errors in higher-level applications and services that first alert administrators to problems at the network level. Ideally, with network monitoring, administrators can capture early warning signs of emerging problems before they manifest themselves at the application level and thus expose customers to failed services.

Port Monitoring

Port monitoring is a subset of network monitoring and focuses on ensuring protocols function correctly between nodes on the network. Port monitoring can be used to ensure that ports required for supported protocols are in fact open. For example, the Simple Network Management Protocol (SMNP) should have its port open on many devices, while the DNS port should only be open on DNS servers. Port monitoring clearly is helpful from a security perspective as well and can help to detect when an unauthorized port is open.

The wide variety of ways we monitor Web applications is due to the many types of components that are required to build these applications. The results of these different tests are useful on their own but can be even more informative when combined to allow systems managers and network administrators to "connect the dots" and detect problems before they become critical failures.

External Monitoring Analysis and Response

With data collected from external monitoring systems, you can more effectively manage networks and applications. External monitoring analysis can be incorporated into network and application management practices in several ways:

- Detecting, pinpointing, and diagnosing problem areas
- Providing early warning analysis
- Improving the effectiveness and speed of response to technical issues
- Incorporating monitoring with IT support services

External monitoring data can provide valuable insight into problem areas before they become obvious problems. For example, external monitoring can detect trends of increasing response times from a Web site that correlate with increased traffic to the site. The combination of internal log data and external monitoring data can be used to build a business case for additional resources. Similarly, external monitoring data can provide early warning signs of imminent failure.

Of course, external monitoring data is not just for warning of potential upcoming problems; it can be useful for measuring the effectiveness of responses to problems. You could, for example, measure the effectiveness of adding servers to an application server cluster or a set of load balanced Web servers.

Another advantage of external monitoring comes with the ability to script events to trigger the automatic creation of a support desk ticket. For example, if an external monitor detects that a server is down, the monitoring system could submit a ticket along with details from the external monitor testing script. This type of tight integration can reduce the window of time between detection and response and therefore potentially reduce the overall impact of the problem.

Implementing Tests

The quality of the external monitoring data you collect is directly related to the quality of the tests you perform. As noted earlier, you can check a Web site to ensure it does not return a page not found error or a timeout error and then assume the Web site is up and running. The problem is that this is such a limited test that it does not tell you much, other than the hardware is functioning, the OS is running, and the Web server can return an HTML page. It does not tell you:

- Whether authentication is working
- Whether data entry forms accept data as expected
- Whether queries return reasonable results
- How fast the page is generated, except that it is fast enough to avoid a timeout error

It also does not inform you about the state of databases, media streaming servers, content management systems, routing analytics, or a wealth of other information that could be useful to network and application administrators.

Scripting Tests

To get the kinds of detailed information you need requires robust scripting capabilities. Ideally, an application manager should be able to record a series of steps navigating a site, entering data, selecting menu options, and running services from a browser. If those responsible for developing test scripts are relieved of the low-level, tedious details of formulating proper script syntax, there is more time to focus on functional aspects. For example, testers could ensure all key functions are tested and primary navigation paths through a site are traversed.

Robust scripting mechanisms help to reduce troubleshooting because steps are captured as they are executed by a knowledgeable user. Another advantage of robust scripting is that it can promote reuse. This is especially true if scripts can call other scripts. This allows for the complex tests to be built on a series of simpler, previously tested scripts.

Browser-Based Monitoring

It should be noted that script-based testing of today's rich Internet applications (RIAs) is more complex than conventional HTML-based pages. In RIAs, some code is executed on servers and some on the client. Some functionality is provided with server-side scripting and some is done with the client using JavaScript or other scripting language. To accurately reflect the end user experience, it is important to use browser-based testing. Testing should not be done, for example, from a Perl script running on another server testing a set of services called by the UI. For complete end to end testing, scripts need to use the same browser-based features that end users use when interacting with the Web application.

Summary

External monitoring constitutes a set of technical resources and procedures for utilizing the data gathered by those procedures. Effective external monitoring builds on a range of tests, from simple connectivity tests to basic service tests on up to fully integrated application testing. To achieve comprehensive external monitoring, you must be prepared to test according to the types of services you use. For example, the way you test a Web server is not sufficient for testing a database, which in turn is not how you would test a streaming media server. With the detailed data you can collect from a variety of external monitor types, you can implement a number of analysis and response procedures that can support both early warning and post-modification monitoring of changes to the IT infrastructure. To realize the full potential of external monitoring, though, you have to develop and deploy robust scripts that test the full range of functionality in your applications.