Back to Basics: Health Monitors and Load Balancing

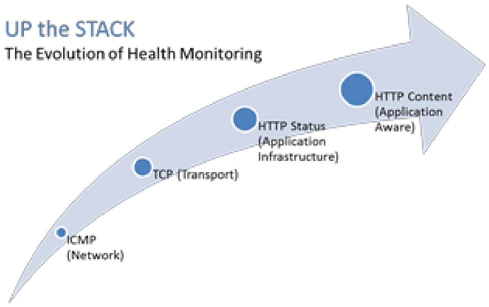
Lori MacVittie, 2012-21-11

#webperf #ado Because every connection counts

One of the truisms of architecting highly available systems is that you never, ever want to load balance a request to a system that is down. Therefore, some sort of health (status) monitoring is required. For applications, that means not just pinging the network interface or opening a TCP connection, it means querying the application and verifying that the response is valid.

This, obviously, requires the application to respond. And respond often. Best practices suggest determining availability every 5 seconds or so. That means every X seconds the load balancing service is going to open up a connection to the application and make a request. Just like a user would do.

That adds load to the application. It consumes network, transport, application and (possibly) database resources. Resources that cannot be used to service customers. While the impact on a single application may appear trivial, it's not. Remember, **as load increases performance decreases.** And no matter how trivial it may appear, health monitoring is adding load to what may be an already heavily loaded application.


UP the STACK
The Evolution of Health Monitoring
HTTP Content (Application Aware)
HTTP Status (Application Infrastructure)
TCP (Transport)
ICMP (Network)

But Lori, you may be thinking, you expound on the importance of monitoring and **visibility** all the time! Are you saying we shouldn't be monitoring applications?

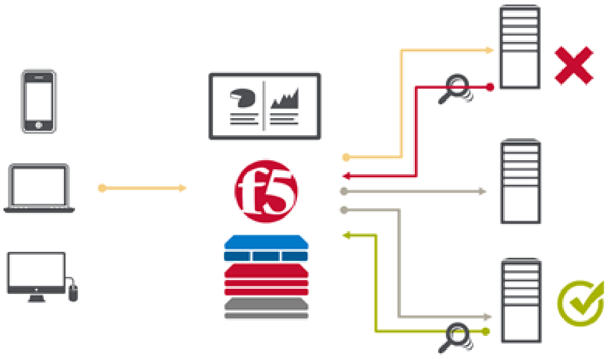
Nope, not at all. Visibility is paramount, providing the actionable data necessary to enable highly dynamic, automated operations such as elasticity. Visibility through health-monitoring is a critical means of ensuring availability at both the local and global level.

What we may need to do, however, is move from active to passive monitoring.

PASSIVE MONITORING

Passive monitoring, as the modifier suggests, is not an active process. The Load balancer does not open up connections nor query an application itself. Instead, it snoops on responses being returned to clients and from that infers the current status of the application.

For example, if a request for content results in an HTTP error message, the load balancer can determine whether or not the application is available and capable of processing subsequent requests. If the load balancer is a BIG-IP, it can mark the service as "down" and invoke an active monitor to probe the application status as well as retrying the request to another available instance – insuring end-users do not see an error.



Passive (inband) monitors are not binary. That is, they aren't simple "on" or "off" based on HTTP status codes. Such monitors can be configured to track the number of failures and evaluate failure rates against a configurable failure interval. When such thresholds are exceeded, the application can then be marked as "down".

Passive monitors aren't restricted to availability status, either. They can also monitor for performance (response time). Failure to meet response time expectations results in a failure, and the application continues to be watched for subsequent failures.

Passive monitors are, like most inline/inband technologies, transparent. They quietly monitor traffic and act upon that traffic without adding overhead to the process.

Passive monitoring gives operations the visibility necessary to enable predictable performance and to meet or exceed user expectations with respect to uptime, without negatively impacting performance or capacity of the applications it is monitoring.