EXECUTIVE BRIEF

Building Better NPM Dashboards

Network speed and complexity have increased to the point that conventional approaches to network analysis and performance monitoring are proving inadequate. Network professionals depend on knowing what is going on and, more importantly, what is going wrong. Tools that provide them with a limited set of summary statistics, sampled data, and a lot of assumptions do not allow those responsible for network performance to effectively investigate key issues and interpret the results.

Traditional methods of network performance management (NPM) have lagged not because data is unavailable for analysis, but because most network monitoring tools are not designed to harness the computational power needed to fully analyze the amount of data that is available. Instead, they present summaries or apply insufficient discrimination, bombarding the user with false alarms and low-priority issues.

Network professionals need a better NPM dashboard, one that combines full and complete analysis with an inherent focus on what he or she would find most useful. This cannot be accomplished by adding more views to a traditional dashboard or by running it faster. Instead, an entirely new approach to network performance monitoring is required. In this brief, we discuss some of our findings.

Overview of two approaches

Modern NPM dashboards typically take an "analysis up" approach, using the information they have collected or can generate as the starting point for what to display. It isn't practical to present all of this information to the user, so most NPM products provide summaries based on what and where the user anticipates issues.

When the user has correctly anticipated what will be of future interest, a view of metrics consolidated over time may provide helpful context. However, when a problem or security issue occurs, dashboards taking this approach generally function only as the starting point of an investigation. Effective action generally requires other tools and an additional workflow.

If the user didn't anticipate the location or kind of future network problem, the situation becomes even less tolerable, requiring that the user look for visibility in the general problem area, collect data over time, and initiate an investigation. When mean time to resolution (MTTR) matters, and it always matters, this lengthy process may be unacceptable.

An alternative to "analysis up" is "requirements down," in which the user starts by finding what's most useful and works down to the analysis. This approach tends to require more analysis by the NPM tool but makes fewer demands on the user and has the potential to significantly reduce MTTR. Understanding the possibilities of the "requirements down" approach requires understanding the limitations of current NPM dashboards.

Legacy NPM dashboards require a choice: aggregation loss or low performance

Most dashboards today provide a consolidated view of network traffic such as counts of nodes, conversations or other network types, or average values of some network metric. These are typically aggregated from multiple network devices, each of which keeps its own statistics. Any gain in simplicity afforded by this approach is lost by its lack of detail. 99.5 percent of a network may be performing well, but the other 0.5 percent can still bring it crashing down. This shortfall in data presentation is known as "aggregation loss."

Practitioners circumvent aggregation loss by looking for key metrics that reveal specific types of network traffic. This exposes a second problem: computing meaningful analytics on individual instances of network traffic is computationally expensive. In other words, data floods the monitoring tools faster than the required metrics can be computed. Modern NPM products generally max out at around 2 Gbps of real-time full-packet-capture analytics, so these products capture network traffic and analyze it later. The drawback with this approach is that post-capture analytics do not appear on real-time dashboards. Even worse, if the volume of traffic being analyzed is large, the analytics can take hours or even days to complete, so although the problem is occurring in real time, the solution is not.

These issues can be illustrated by flipping the problem around. Most network problems end up being characterized by a single network transaction or a group of network transactions that share a common trait such as routing path, protocol, or application. A network engineer investigates a problem by associating it with the relevant set of network transactions as quickly as possible. On a high-speed enterprise network, there may be many hundreds of thousands of transactions in progress at any given moment. As a result, network professionals typically deploy filters to reduce the scope of their investigation, then translate the results into transactions, typically an independent step, sometimes requiring a different tool. Going to a filtered set of relevant transactions is generally an iterative and time-consuming process. In other words, there is a complete lack of actionable precision in the initial graphical display which necessitates additional steps and complexity.

NetFlow is not the solution

NetFlow is a common, lightweight protocol that can often be generated and consumed at full network speeds. However, NPM dashboards based on NetFlow cannot provide individual network transaction data. At best, these dashboards require integration or coordination with other tools that can retrieve this data.

NetFlow also has another major shortcoming. Network transactions, as reported by NetFlow, cannot provide vital information about latency or quality metrics. This issue of NetFlow's inadequacy is discussed at greater length in a separate white paper here.

A more recent approach, sFlow uses data sampling to improve performance. However, while it is faster, the sampling can make it even less accurate than NetFlow or IPFIX (a derivative of NetFlow), causing it to miss many potential performance issues and preventing it from identifying latency or quality metrics.

Summarizing the trouble with modern NPM dashboards

Most NPM dashboards today suffer from the following flaws:

- Aggregation loss which means that even when the big picture is available, critical details may be missing
- Real-time limitations, so analytics are unable to keep up with fast networks and critical metrics don't make it to the dashboard
- Lack of actionable precision that requires multiple, time-consuming, manual steps to further isolate the problem
- NetFlow limitations that prevent direct access to individual network transactions and an inability of the dashboard to display the most important metrics: quality and latency

At best, these NPM dashboards give clues, but not guidance; their information may be useful, but it is certainly not actionable.

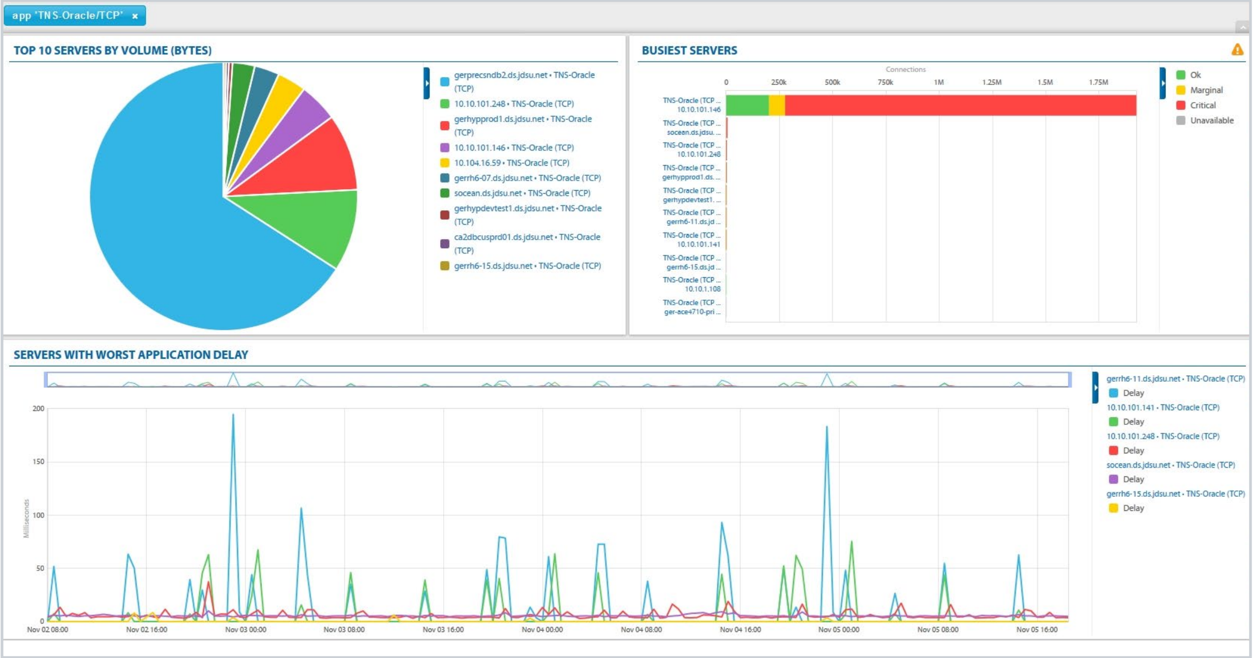
app 'TNS-Oracle/TCP' ×
TOP 10 SERVERS BY VOLUME (BYTES)
🟦 gerprecsndb2.ds.jdsu.net • TNS-Oracle (TCP)
🟩 10.10.101.248 • TNS-Oracle (TCP)
🟥 gerhypprod1.ds.jdsu.net • TNS-Oracle (TCP)
🟪 10.10.101.146 • TNS-Oracle (TCP)
🟨 10.104.16.59 • TNS-Oracle (TCP)
🟦 gerrh6-07.ds.jdsu.net • TNS-Oracle (TCP)
🟩 socean.ds.jdsu.net • TNS-Oracle (TCP)
🟥 gerhypdevtest1.ds.jdsu.net • TNS-Oracle (TCP)
🟪 ca2dbcusprd01.ds.jdsu.net • TNS-Oracle (TCP)
🟨 gerrh6-15.ds.jdsu.net • TNS-Oracle (TCP)
BUSIEST SERVERS
Connections
0
250k
500k
750k
1M
1.25M
1.5M
1.75M
TNS-Oracle (TCP ... 10.10.101.146
TNS-Oracle (TCP ... socean.ds.jdsu. ...
TNS-Oracle (TCP ... 10.10.101.248
TNS-Oracle (TCP ... gerhypprod1.ds. ...
TNS-Oracle (TCP ... gerhypdevtest1. ...
TNS-Oracle (TCP ... gerrh6-11.ds.jd ...
TNS-Oracle (TCP ... 10.10.101.141
TNS-Oracle (TCP ... gerrh6-15.ds.jd ...
TNS-Oracle (TCP ... 10.10.1.108
TNS-Oracle (TCP ... ger-ace4710-pri ...
🟩 Ok
🟨 Marginal
🟥 Critical
⬜ Unavailable
SERVERS WITH WORST APPLICATION DELAY
200
150
100
50
0
Milliseconds
Nov 02 08:00
Nov 02 16:00
Nov 03 00:00
Nov 03 08:00
Nov 03 16:00
Nov 04 00:00
Nov 04 08:00
Nov 04 16:00
Nov 05 00:00
Nov 05 08:00
Nov 05 16:00
gerrh6-11.ds.jdsu.net • TNS-Oracle (TCP)
🟦 Delay
10.10.101.141 • TNS-Oracle (TCP)
🟩 Delay
10.10.101.248 • TNS-Oracle (TCP)
🟥 Delay
socean.ds.jdsu.net • TNS-Oracle (TCP)
🟪 Delay
gerrh6-15.ds.jdsu.net • TNS-Oracle (TCP)
🟨 Delay

Figure 1: This recent dashboard illustrates the weakness of "analysis up" approaches: filtered to an application of interest, it presents one server as busiest and others as having worst delays. The actual users' experience cannot be determined without an investigation, nor is the best starting point for the investigation made clear.

The Logical Step: Real-Time, Actionable NPM dashboards

For network engineers to become truly proactive in managing their networks, they require dashboards that not only identify that there are issues but also quickly provide critical points to investigate and resolve those issues without burdening them with separate steps.

When deploying NPM, network professionals expect a continuous baseline view of what is happening on the network. Many products log anomalies or provide alerts, but these alerts are meaningless without understanding their larger context. In some instances, extensive manual configuration of NPM tools can result in more of the context dependency. However, in both scenarios the result is too many false positives while genuine problems are buried under too much noise.

There is a new approach to NPM which harnesses modern parallel computing power to perform analytics in real time on all network transactions. Instead of simply populating dashboards with a visual representation of network traffic as most dashboards do today, users of NPM tools need the problem areas to be exposed in real time. By focusing on a network's worst-performing metrics and optimizing the navigation filtering, rather than simply showing a summary of network traffic on a dashboard, the amount of computational power can be greatly reduced.

In addition, the types of metrics displayed on NPM dashboards needs to change. The most informative network performance metrics are: latency (delays in service); quality (accuracy and number of errors); utilization (demand for services); and saturation (how much traffic is used relative to capacity). These four "Golden Signals" were first proposed by Google's SRE (Site Recovery Engineering) team for their value in indicating network health and pinpointing trouble spots. Because the use of Unified Communications (UC) is now common on high-speed enterprise networks, monitoring MOS scores for voice and video transactions is also relevant.

Summary

To troubleshoot today's networks with the lowest possible MTTR, IT professionals need the ability to analyze network traffic at the actual speed of the network itself. From a technical perspective, this means that NPM vendors need to develop solutions with the underlying power and granularity to perform analytics on every network transaction at 20 Gbps or higher network speeds and present the results to the practitioner in an interactive, fully real-time fashion.

The most valuable solution is one that reveals where the worst performance on the network is occurring in real time. Who is experiencing the poorest TCP quality? Where are the worst network saturation or latency problems occurring? Where is the worst application latency showing up? With this information and troubleshooting power at their fingertips, IT professionals will find that their initial view becomes actionable. The goal is the development of NPM tools that give IT professionals the power to provide users a quantifiably more robust (and more secure) network environment. Real-time, actionable NPM dashboards are a crucial step toward achieving that goal.