


Realtime
publishers

The Reference Guide to

Network Management Protocols

sponsored by

solarwinds 

Prologue.....	1
Volume 1: The Fundamental Protocols of Network Management	2
ICMP	2
SNMP	4
ICMP and SNMP in Today's NMSs.....	6
Volume 2: The Windows Management Protocols	7
RPC	8
WMI.....	8
WS-Management.....	9
RDP	10
Windows Management Protocols in Today's NMSs	11
Volume 3: Telnet, SSH, and Syslog	12
Telnet.....	12
SSH	13
Syslog.....	14
Volume 4: The Flow-Based Protocols	17
Understanding Network Flows	17
NetFlow, J-Flow, sFlow, and IPFIX.....	18
Leveraging NetFlow.....	19
NetFlow in Today's Network Management Solutions	20
Volume 5: Cisco IP Service Level Agreements.....	21
IP SLA Extends Traditional Network Monitoring.....	22
IP SLA Responders.....	23
IP SLA and Test Processing	24
IP SLA in Today's NMSs.....	24

Prologue

Network management has been around since the very first network was connected. Any time two or more computers are plugged into each other, there eventually comes the need to manage, monitor, and analyze their communication. To that end, a suite of management protocols has been developed over time that assists in understanding the conversations that occur between networked devices.

These specialized protocols are fundamentally necessary due to the nature of network communication. You simply can't see bits and bytes crossing the wires in your data center. Thus, the only way to understand your underlying network behaviors is by relying on the reported data that is collected through these special protocols.

Your network today passes numerous protocols across its wires. These protocols are most commonly Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) in your Local Area Network (LAN), but most networks also have requirements for those that are less common (for example, AppleTalk and IPX) as well as others that are used for external routing (for example, BGP and EIGRP). What's interesting about the *network management* protocols is that they operate functionally differently than those whose job is to pass data between computers. As such, the protocol that you use for application access and data transfer operates in parallel with those you use to manage your network.

The goal of this five-part Reference Guide is to assist you with understanding and successfully employing those special network management protocols. Building upon the very simplest in Internet Control Message Protocol (ICMP) and Simple Network Management Protocol (SNMP), this guide explores today's modern solutions for finding meaning in otherwise-impossible mounds of data.

Volume 1: The Fundamental Protocols of Network Management

Starting with the basics are the two fundamental protocols of network management, Internet Control Message Protocol (ICMP) and Simple Network Management Protocol (SNMP). Having been in service for nearly as long as the earliest networks, these two protocols remain useful as core tools for troubleshooting and managing your network. Most commonly known through its “ping” command, ICMP creates a low-level request and response that ensures core connectivity between two network endpoints. SNMP goes a step further. It elevates that level of gathered data by enabling devices to share their basic configuration and onboard metrics.

ICMP

ICMP is considered one of the core protocols in the Internet Protocol (IP) suite. Unlike the common data transfer protocols, such as TCP and UDP, ICMP is not typically used by applications as a method of communication. Rooted in the IP portion of TCP/IP, ICMP operates outside the rules of traditional TCP and UDP. Simply put, modifying TCP connectivity or access rules does nothing to impact ICMP, with the reverse also being true.

This isolation from the common data transfer protocols is what gives ICMP great power in troubleshooting and otherwise managing your network. Its protocol specification enjoys an extremely limited set of commands, which are restricted by design to the tasks of exploring host and network connectivity and routing. As such, ICMP traffic is commonly available in all but the most highly-secured of networks.

ICMP’s most common command is invoked by using the command-line “ping” executable. This executable by name is common across virtually every Operating System (OS) and Internetwork Operating System (IOS) in existence today. Executing a ping against a remote IP address returns a small, but very powerful, amount of information back to the user. As the following code snippet shows, a successful ping reply tells the user that the host is up, operational, and responding to the most basic of network requests.

```
C:\>ping www.solarwinds.com
```

```
Pinging www.solarwinds.com [74.115.13.20] with 32 bytes of data:
Reply from 74.115.13.20: bytes=32 time=34ms TTL=117
Reply from 74.115.13.20: bytes=32 time=39ms TTL=117
Reply from 74.115.13.20: bytes=32 time=34ms TTL=117
Reply from 74.115.13.20: bytes=32 time=34ms TTL=117
```

```
Ping statistics for 74.115.13.20:
```

```
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 34ms, Maximum = 39ms, Average = 35ms
```

But this isn't all the information that is gained from a ping command's response. Also present in that ping reply is additional information about the state of the connection between the source and target. Its short response illuminates specifics about the number of milliseconds required to complete the roundtrip request. This information provides a very high-level perspective of the connection's latency. Latent network connections can occur due to a list of factors, including bandwidth congestion, rate of errors through any of the links between source and target, and processing issues through any of the connected devices in the path.

Ping Provides Basic Information

Although the result from a ping command provides information about a network route's latency, this information is exceptionally coarse in its granularity. Latency as defined by a ping response represents little more than the amount of time that occurred between sending the ping request and receiving its reply. As such, its response illuminates little about the actual route taken and behaviors seen through its journey from source to target. For more detailed information, alternative protocols and techniques are necessary. Future volumes of this guide will discuss these options.

The effective transfer speed between a source and target depends heavily on a number of factors. As one example, each and every time a packet "hops" through a network device—router, switch, or firewall—a small amount of delay is introduced into its roundtrip time. Thus, as the number of hops increases, the effective performance of the connection decreases slightly.

A ping response will report on the number of hops required to complete that travel from source to destination as a function of its Time To Live (TTL) metric. This is done by subtracting the configured TTL from the reported TTL found in a ping response. Today's Windows OSs use a default value of 128 as their TTL, with every hop from source to destination reducing this value by one. As a result, in the previous code snippet, a connection to the Web site www.solarwinds.com requires 11 individual hops to route between source and destination.

Transfer speed can also be affected by the size of the data being transferred from source to destination. Larger-sized data is usually fragmented in transit to increase its performance in transmitting from source to target. Ping by default will send a 32-byte string of repeating alphabetical characters as its payload during a request. Both the size of that payload and a setting that determines whether it can be fragmented are common options with each OS/IOS ping command. Some ping commands also enable the customization of the ping payload as an advanced option. This is often used to verify whether the composition of the data itself is presenting an impact on a connection's performance. Each of these capabilities is available to verify how well a connection is performing based on the size and contents of a ping request's payload.

A final piece of critical information found through an ICMP ping relates to the high-level quality of the connection between source and target. Listed in the final four lines of a successful ping response is information about that connection's packet loss. Packets that do not successfully make their way from source to target are considered "lost" by ping. The count of lost packets (as well as the resulting percentage of lost traffic as a function of total traffic) represents a level of transferred data that must be presented by TCP to ensure its payload transfers successfully. This designation often indicates that a problem is occurring within that network connection, such as signal degradation, oversaturated network links, packets that were corrupted or rejected in transit, faulty networking hardware or drivers, or other problems intrinsic to the network path. Although the specifics of the problem are left to other more granular protocols, this simple command provides a very easy way to gauge a connection's health at the highest of levels. Although packet loss for communications such as common file transfers tend only to reduce the overall performance of the link, its presence can be very problematic for certain types of stream-oriented traffic. For example, VoIP, videoconferencing, remote access, streaming, and other latency-insensitive traffic types are particularly affected by packet loss.

SNMP

Although ICMP is both simple and pervasive, its commands are by design very simplistic. Running an ICMP query gives you only very basic information about a host's connection, which is why it is most commonly used to verify only whether a host is listening. If you want higher-quality data, you need a protocol with greater reach into the devices on your network.

One of the earliest of these protocols remains in heavy use today. SNMP goes beyond ICMP's very simple and highly-structured information to enable the gathering of virtually any kind of data from a network device. Due to SNMP's long history and widespread use, virtually every network device—and even many servers and applications—have been made SNMP-aware. "Awareness" in this context means that the device is configured to receive and respond to SNMP requests from a central Network Management Solution (NMS).

The structure of an SNMP operation is based on a request for information, fundamentally using SNMP's GET and GET-NEXT operations. These two commands enable an SNMP requestor to ask for a particular piece of information that is stored on a target device. The GET command is followed with a type of address for the information that is desired. GET-NEXT is used when the requestor wants to locate the "next" piece of information that is stored in that device's hierarchy. Both commands leverage port 161/UDP for their communication, which is a configurable port on virtually all devices.

To access that information, each register on each device requires its own unique identifier. This identifier must obviously be unique across every device in existence and with each register of information itself requiring its own address. Like IP addresses, SNMP uses a dotted decimal notation for denoting that unique identifier, called an Object Identifier (OID). For example, to access the value of the *agentCurrentCPUUtilization* register on a particular type of Cisco switch, SNMP would be configured to GET its value from the OID 1.3.6.1.4.1.14179.1.1.5.1 at the switch's IP address.

Obviously, the list of potential OIDs for all the devices on your network is unfathomably large. Your NMS will typically arrive with a large set of OIDs already present in its database. Additional OIDs can usually be downloaded from the vendor and automatically ingested into your SNMP solution's database. Because SNMP is intended to be a cross-vendor solution, this extensibility enables a single NMS to manage and monitor information across all classes of SNMP-aware devices and applications.

The result is that a single central NMS can create an ongoing picture of the internal behaviors of the devices on your network. That picture includes information about device performance, and can include additional details such as configurations, environmental characteristics, and network statistics. Because the NMS regularly polls SNMP endpoints on your behalf, it can store this information for later retrieval.

The second job of an effective NMS is the creation of visualizations of this data that are useful for administrators. Crunching numbers from gathered statistics and presenting visualizations are key factors in finding an NMS that works for your environment.

Although SNMP's GET command enables the gathering of data from network devices, a second command, SET, is used to update information. For certain types of OIDs, specifically those that relate to device configurations, it is possible to use the SET command with an OID and the value to be changed to remotely change the configuration of a networked device.

In every case, for an NMS or other requestor to work with a remote device, that device must be preconfigured with an SNMP Community String. This string—of which one is typically used for reading information with a separate and second one used for updating information—operates as a type of shared-secret password between the NMS and every device it manages. As a shared secret, this string must be entered into each device and into the NMS before each half will communicate with the other. The management of these Community Strings is a common administrative headache in the use of SNMP, one that is eased through the use of fully-featured NMSs.

SNMP and Security

SNMP is a unified solution for gathering and updating data across all the devices and applications on your network, so it goes without saying that you must be careful in its implementation. Properly securing SNMP is critical. That security can occur through a range of tactics, including the use of access lists on firewalls to isolate SNMP traffic, ensuring that strong Community Strings are used to prevent hacking, leveraging the privacy and encryption functions that are available in SNMP v3, and/or the use of isolated management VLANs that are dedicated for SNMP communication. Any or all of these capabilities goes far in securing SNMP against the threat of external attack.

As you can see with this implementation, the commands discussed to this point are useful only when an NMS can regularly poll each device on the network. Some situations, however, may mandate that a device unilaterally inform its NMS about a particular preconfigured condition. In this case, an unsolicited SNMP TRAP command can be sent in one direction from device to NMS to inform that the condition has occurred. This trap can relate to the server being powered up or down or a preconfigured network condition such as congestion that is currently occurring within the device's sensors. SNMP TRAP commands are sent across a different port, 162/UDP by default. This port can additionally be changed on virtually all devices if desired.

ICMP and SNMP in Today's NMSs

Today's NMSs rely heavily upon both ICMP and SNMP for their monitoring of network conditions. ICMP is commonly used for information on a per-device or per-node basis, such as device status monitoring, availability monitoring, and network latency to individual devices. SNMP augments this information through its added data about device behaviors and characteristics. SNMP can collect and report on deeper metrics associated with each device, such as internal performance statistics or its configuration. SNMP's deeper level of data illuminates more detail about an environment, providing information such as individual interface status, CPU/memory/disk utilization and performance, and per-interface traffic and level of errors.

Advanced NMSs such as the Orion Network Performance Monitor (NPM) from SolarWinds enables all these features in a singular solution. It also offers several capabilities above and beyond those listed. Orion NPM enables administrators to collect information and combine it with additional data.

Volume 2: The Windows Management Protocols

The successful monitoring and management of Windows-based systems requires more than standards-based protocols such as Internet Control Message Protocol (ICMP) and Simple Network Management Protocol (SNMP). Also needed are Windows-specific protocols. These protocols exist high in the IP stack's Application layer and by design rely on others below them for routing and switching support. These added protocols are used for applications such as DNS, the Web, FTP, and mail transfer among many others.

Where this discussion leads in terms of Windows management is related to those very Application-layer protocols. The Microsoft Windows Operating System (OS) leverages its own suite of protocols for communications between Windows servers and workstations. These protocols layer atop core TCP and UDP to enable server and service communication across an IP network. Two examples of these high-level protocols are the Remote Procedure Call (RPC) protocol used for Windows inter-process communications, and the Remote Desktop Protocol (RDP), which is used for transferring display and control information between a server and client.

Other protocols that are used by the Microsoft Windows OS include the Windows Management Instrumentation (WMI) protocol as well as the more-recent implementation of WS-Management. WS-Management is manifested within the Microsoft Windows OS through its Windows Remote Management v1.1 and v2.0 implementations.

Highlighting these OS-specific protocols is necessary because of the information that they can illuminate to the network administrator. Protocols such as ICMP and SNMP, discussed in Volume 1, are capable by design of presenting only certain types of metrics back to the administrator. Although SNMP can be extended to support the storage of virtually any data point, that extension must be preconfigured within each OS or application. Such extensions for SNMP awareness have not been undertaken for a large amount of on-system data that is needed for truly holistic management.

Microsoft, however, has made a substantial effort in enabling WMI awareness for its OSs and applications. Thus, deep monitoring support that could not otherwise be monitored through ICMP or SNMP is available for many applications using WMI. Thus, any Network Management Solution (NMS) that desires to peer into the Windows OS and its applications must additionally provide support for these other protocols.

RPC

The backbone of Windows network communication rests on the RPC protocol, and as such, RPC is a necessary part of all Windows environments. At the same time, that pervasiveness makes it especially problematic for network security. The Microsoft implementation of the RPC protocol leverages an RPC endpoint mapping service, which is given the job of identifying and cataloging which ports are to be used by which services on a Windows server. This determination is made when the machine starts up as well as when services are started during normal operations.

Although RPC's dynamic nature enables a large number of services on each system to intercommunicate across a local area connection, that same dynamic approach means any number of TCP ports can be open and listening on a Windows instance at any point in time. An RPC connection on Windows Server 2008 can occur between a range of ports between 49152/TCP and 65535/TCP at both the source and destination. Those ports often change as services and processes instantiate, close, as well as make and complete requests across the network. Because of this, in every Windows network, these ephemeral ports are always listening and always changing.

As a result, an effective NMS must be able to monitor for the availability of RPC-based communication as well as which services are listening on which ports. This information provides great insight into the types of communication that are occurring on the network, and helps to isolate problems that relate to inter-process communications. Further, as many forms of malware leverage the same dynamic infrastructure that is used by legitimate Windows services, this level of monitoring enables administrators to root out infections in real time as they occur.

WMI

RPC traffic in and of itself provides only a high-level representation of the overall communication flow. In its Windows OS, Microsoft has developed the WMI protocol as a proprietary alternative to SNMP. This remote monitoring and management protocol represents Microsoft's implementation of the Web-Based Enterprise Management (WBEM) Common Information Model (CIM) standards from the Distributed Management Task Force (DMTF). WMI operates much like SNMP in that WMI can be used for gathering metrics data and updating certain configurations. However, WMI is much different than SNMP in that WMI's reach is limited to the Windows OS and installed applications.

WMI is further different from SNMP through the ways in which WMI's database can be interfaced. SNMP's request/response activity is defined directly in SNMP; in contrast, WMI requests can be wrapped into one of many languages. As examples for administrative scripting, Microsoft's implementations of both VBScript and Windows PowerShell include support for creating WMI queries.

The query syntax used for WMI calls is also extensible. Whereas SNMP includes but two commands for requesting information (GET and GET-NEXT), WMI offers the much richer WMI Query Language (WQL) for queries. Specific WMI providers can leverage their own query support if an enumeration technique is built-in to the provider. For example, the generic WQL query for enumerating information from the Win32_DiskDrive provider would resemble *Select * from Win32_DiskDrive*. The resulting query would result in a collection of items from that provider. WQL includes additional constraints for limiting resulting information, although a common practice is to build constraints into the surrounding scripting language where possible. Doing so provides the greatest level of flexibility while circumventing limitations in the constraint options native to WQL.

Free Tool: SolarWinds WMI Monitor

WMI's powerful extensibility also exposes a measure of complexity. This complexity means that most administrators leverage external tools for working with WMI. One such tool is the SolarWinds WMI Monitor, which can be downloaded from http://www.solarwinds.com/products/freetools/wmi_monitor. This tool enables the monitoring of many facets of your Windows servers and applications through a single user interface (UI).

WS-Management

WMI's native transport for Windows OSs prior to Windows Server 2008 was the Distributed Common Object Model (DCOM). Relying on RPC for its connectivity underpinnings, DCOM functioned well for use in Local Area Networks (LANs). However, RPC's dynamic nature makes it both problematic as well as insecure to operate through firewalls or network architectures that aren't relatively "open." This limitation meant that accomplishing the remote management of systems outside the traditional LAN was difficult or impossible through the WMI/DCOM combination alone.

To address the problem, Microsoft later adopted the industry-standard WS-Management framework in Windows Server 2008. Microsoft's implementation of WS-Management is represented with its Windows Remote Management (WinRM) service. This industry specification is based on DMTF open standards and Internet standards for Web Services. It leverages the firewall-friendly Simple Object Access Protocol (SOAP) for its exchange of information, extending the monitoring reach of WMI to many new areas.

As a Web Services implementation rather than relying on RPC, WS-Management (and thus WinRM) can much more easily pass WMI data over firewalled networks. A Web Service represents a Web-based endpoint whereby a client can interface to request and submit information. As a Web Service, this endpoint can operate over a single, known port (commonly, 80/TCP or 443/TCP) rather than a range of dynamic ports.

It is important to recognize that Microsoft's implementation of WS-Management layers atop the traditional WMI/DCOM stack, enabling exposure to the stack through the structured Web Service. When enabled (as it is not enabled by default), this architecture retains the former WMI/DCOM compatibility with traditional WMI scripts and application infrastructures while adding the ability for SOAP-aware clients to interact with the server via a Web-friendly transport.

What this means to the average IT organization is that traditionally unmanageable systems—such as those in DMZs or on Extranets—can now be managed through WS-Management-enabled NMSs. In effect, the same types of information that can be gathered through WMI-based solutions can be now obtained in these systems that aren't on the LAN.

RDP

The final Windows management protocol of note is RDP, which has dramatically grown in use since its inception with Windows NT 4.0 Server, Terminal Server Edition. Originally designed for specialized use in connecting remote users to applications on a centralized set of servers, use of RDP has grown to include administrative connections to server desktops.

Unlike the more transaction-based protocols seen in SNMP, WMI, and others, RDP can be considered more like a “streaming” protocol, although this description is not completely accurate. RDP sends screen updates from server to client when the server session's screen changes. It sends keyboard and mouse commands from client to server when keys are pressed or mouse interrupts are processed. Intercepting data from an RDP connection will result in a jumble of screen-part updates interspersed with keyboard and mouse commands, making reconstruction non-trivial without specialized tools.

By default, RDP operates over the target port 3389/TCP. Thus, this single TCP port must be routable through firewalls or through network ACLs for connectivity to RDP hosts. RDP can be reconfigured to route through alternative ports. High-security environments often use tools to encapsulate RDP traffic within SSL, which changes its default port to 443/TCP, adds security and authentication, and enables the secure passing of RDP traffic across the Internet.

RDP's stream-driven nature enables it to pass updates across very low-bandwidth connections. Thus, high-use connections can remain in the data center, with only the resulting presentation data being submitted to the user. For certain applications, this is a boon to remote support. However, this same interactivity makes the protocol highly latency-insensitive. In effect, if you move your mouse, you want its cursor to track with your hand's movement rather than delaying by seconds or partial-seconds.

As a result, monitoring needs for RDP tend towards aggregate statistics over and above specific session details. With remote application support use on the rise, your network management of RDP will likely include the necessary monitoring integrations (both at the network level and via Windows-specific integrations such as WMI) to identify and report on user-experience conditions.

Windows Management Protocols in Today's NMSs

Today's NMSs are tearing down the traditional barriers between the old management silos of "network" versus "servers and applications." This is accomplished through built-in integrations directly into the Windows OS and its installed applications. With an effective NMS in place, it is possible to measure aggregate network statistics across Windows-based protocols such as RPC and RDP, as well as integrate with Windows WMI through both RPC/DCOM and WS-Management (WinRM). The result is an ability to gather OS and application statistics off your servers just like the network statistics you get from ICMP and SNMP.

Orion Application Performance Monitor (APM) from SolarWinds is one solution that accomplishes all this in an easy-to-use package. Orion APM includes native support for WMI, including the ability to define custom WMI monitors. With its internal credential management capabilities, Orion APM can quickly connect to servers and applications for the automated gathering of and reporting on Windows-based data.

Orion Application Performance Monitor (APM) adds advanced application monitoring capabilities such as end-user experience monitoring to create an end-to-end solution for verifying application performance. Orion APM provides deep insight into distributed applications, presenting alerts as well as trending and analysis for every part of an application's infrastructure.

The net result is a single pane of glass that enables the central monitoring of network health alongside application health. Viewing and being alerted on server disk space consumption or Microsoft Exchange database performance can be accomplished via the same interface as network bandwidth or latency information. By aggregating the Windows focus with the network focus, root causes of environment problems can be better tracked to their problem domain. This functionality improves the overall troubleshooting process while ensuring that network teams and server teams all work from the same set of information.

Volume 3: Telnet, SSH, and Syslog

Although the Windows network management protocols tend towards specifically-designed extensibility, the traditional UNIX-based protocols, such as telnet, ssh, and Syslog, tend to be very handy in troubleshooting across a range of network issues. These protocols can be used for troubleshooting basic IP network connectivity, and, in the case of telnet, can even be used as a command-line solution for testing the status of individual TCP ports. In fact, the use of these protocols has grown so ubiquitous across all kinds of networks that referring to them as “UNIX-based” no longer truly represents their cross-platform utility.

In today’s networks, protocols such as telnet, SSH, and Syslog are commonly used in connecting to and managing all manner of network devices. This third volume will explore these three protocols in depth, discussing where they fit into connecting to and managing the business network. In all three cases, these protocols enable vision into network devices and server behaviors that illuminate specific conditions or configurations on the network.

Telnet

Telnet is considered one of the oldest protocols still in common use today. Developed first in 1969, “telnet” actually represents an acronym that stood for “teletype network.” In that time, the telnet command was used in connecting to early mainframe systems, with that same utility relatively unchanged throughout its lifespan. Today’s use of the telnet command is primarily as a mechanism for accessing a UNIX or Linux system’s command-line interface. In networking domains, telnet is also commonly used for connecting to a network device’s command-line management interface.

Since the release of Windows NT 4.0 SP3, the Microsoft Windows Operating System (OS) has incorporated a telnet server into its Services for UNIX add-on product, which was later renamed to Subsystem for UNIX-based Applications (SUA) in Windows Server 2003 R2. Although Microsoft’s telnet server indeed creates a Windows service (daemon) for receiving telnet requests, its use on the Windows platform is relatively uncommon in networks today.

The telnet command encapsulates user data into the same channel as its own control information as it passes to the target computer or device. Thus, telnet functionally operates as both a management protocol and a data transfer protocol. Telnet resides in the IP suite’s Application layer, riding atop the TCP transport and commonly using 23/TCP as its network port.

Today’s use of telnet has fallen out of common practice for many business networks due to its inherent incapability to authenticate target servers. Telnet also lacks key encryption capabilities that ensure its data is secure in transit. The combination of these limitations means that telnet and “telnetting” to a server or device is no longer a security best practice. Replacing the telnet command is the ssh command, discussed in the next section, which includes the necessary capabilities for connection security.

However, this limitation on telnet's use has far from eliminated it from the administrator's quiver of management tools. One feature of telnet that remains in common use even today is in its ability to create "raw" TCP connections between a source and target. This raw connection is used to verify layer 3 and layer 4 (TCP and port) connectivity between two hosts. The following code snippet shows how the telnet command can be invoked against a remote server and TCP port as a low-level test for a listening service at the target IP:

```
telnet 192.168.0.22 80
```

In this snippet, you can see how the telnet command is invoked against the remote server 192.168.0.22 and pointed towards port 80/TCP. Invoking this on a Microsoft Windows instance against the remote server will accomplish one of two results: either the request will be rejected with an obvious error message or the command window will refresh to show a blank screen. A screen refresh to a blank screen tells the administrator that the remote server is indeed listening on that remote port and that networking access control lists (ACLs) or host firewalls are not blocking its communication.

Finding telnet in Today's Microsoft OSs

Today's Microsoft OSs (Windows Vista or later clients and Windows Server 2008 or later servers) no longer have the telnet command natively available. The command must be installed after an OS installation from within the Control Panel. To do so, navigate to the Programs and Features node, and select Turn Windows features on or off. In the resulting window, choose to enable the Telnet Client feature.

SSH

The ssh or "secure shell" command has in most environments superseded telnet as the best practice for connecting to remote servers and desktops. From the perspective of the user, ssh's most basic functionality performs essentially the same function as telnet—it opens a connection to the command-line interface on an identified remote server. Where ssh differs is in how it secures that connection from end to end.

Originally developed in 1995, ssh uses public-key cryptography to first authenticate the remote computer. Once authenticated, ssh creates a connection that is then secured to ensure both data integrity and confidentiality as it crosses the network. Unlike telnet, whose codebase has relatively stabilized through its long history, a number of ssh implementations are available today. Its myriad versions have been created in many ways due to the discovery of security vulnerabilities in previous versions. The currently-accepted version of ssh is titled "SSH-2" and is a proposed Internet standard under review by the Internet Engineering Task Force (IETF).

The actual use of ssh differs from telnet in that ssh can be a platform upon which additional functionality can be hosted. In addition to creating a secure connection to a command-line interface, the ssh protocol can be leveraged for many uses:

- Single-line command execution
- File transfer, manifested as SCP, SFTP, or rsync
- Port forwarding or tunneling
- Creation of VPN connections, enabled through the OpenSSH distribution
- Web browsing via the SOCKS protocol
- Remote directory mounting, manifested as SSHFS

In short, although most network devices retain the capability to use telnet for remote management, it is considered the industry best practice to use ssh. Although ssh is not natively available in the Microsoft Windows OS in any edition, clients for ssh are available as installed applications to accomplish necessary management.

Syslog

Grouped in this volume because of its UNIX-based roots, Syslog is a fundamentally different protocol in terms of its utility to network management. Syslog is a mechanism whereby event log information from one or more servers or devices can be aggregated into a single database for storage and historical analysis. Like telnet and ssh, Syslog has a long history, starting with its initial development in the 1980s with the Sendmail project.

The aggregation of event log data provided by Syslog is useful for the business network, although most specifically for use in auditing purposes. Being event-based data in nature, the kind of data aggregated by Syslog tends to relate to the contents of various logs on individual devices themselves. This is much different than the kinds of sensor data that is used in viewing network packets, analyzing flows, or monitoring performance metrics.

Due to the recent surge in more-useful technologies for these kinds of analysis (to be discussed in Volumes 4 and 5), today's use of Syslog is often focused on data for auditors. At a high level, today's industry and regulatory compliance laws mandate that information about user and administrator actions must be consolidated into a protected database. That database generally must include non-repudiation features that protect it from malicious corruption or deletion. This capability is not typically available on individual devices. Using Syslog to consolidate this information across multiple devices and into a separate database fulfills those necessary auditing requirements.

Free Tool: SolarWinds Kiwi Syslog Server

The servers that gather and store Syslog information from devices across the network come in many shapes and sizes. One solution that accomplishes the job with many desired features is SolarWinds Kiwi Syslog Server, which can be freely downloaded from http://www.solarwinds.com/products/freetools/kiwi_syslog_server. This download includes both the Syslog server software for gathering and storing log information as well as the necessary Log Forwarding software that is installed to each Windows device under management.

The implementation of Syslog in servers and network devices can and usually is dramatically different depending on the device. For example, configuring a Cisco PIX firewall for Syslog can use a command structure similar to the following example:

```
logging on
logging standby
logging timestamp
logging trap notifications
logging facility 19
logging host inside 192.168.1.100
```

In contrast, configuring the same kind of logging on a Cisco CatOS switch can use the following command structure:

```
set logging server enable
set logging server 192.168.1.100
set logging level all 5
set logging server severity 6
```

UNIX, Linux, and Microsoft Windows OSs also have different mechanisms for enabling Syslog logging. Refer to your product documentation for specific information associated with enabling and configuring Syslog on each device. Be aware also that a Syslog service or daemon must be enabled on the target host (192.168.1.100, in the previous examples) to receive the information.

Telnet, SSH, and Syslog in today's NMSs, as with those focused on Microsoft Windows, are useful for the management of your network. Telnet and ssh are both useful for network testing as well as connecting to the devices across your network. The enabling of Syslog across devices on your network automatically creates an auditable database of information useful for compliance auditors.

Yet in all these situations, the command is only as useful as the endpoint to which you target it. Knowing that you can use telnet to verify TCP port listeners is useful if you're willing to repeatedly enter TCP ports to verify. Creating a proactive environment where open ports are automatically tested and alerted isn't easy to create with command lines only. Simply finding the right IP addresses to connect and interface with grows unwieldy as your environment scales. Setting up Syslog and configuring it across a range of devices (and device configurations) is a manual and error-prone activity.

Tools such as SolarWinds Kiwi Syslog Server, SolarWinds Orion Network Performance Monitor, and the SolarWinds Orion Network Configuration Manager (NCM) ease the management burden of each of these processes through high levels of built-in automation. Once installed, SolarWinds software can easily discover the devices on your network, immediately creating a map of connections for your administration while quickly bringing those devices under unified management. Integrating Syslog capabilities with its other fault and performance utilities, the components in the SolarWinds Orion family aid in finding performance issues and updating configurations through their unified interface.

Volume 4: The Flow-Based Protocols

The management protocols discussed in this Reference Guide have thus far dealt heavily with the needs for configuration management and inventory. Basic troubleshooting capabilities were also discussed with tools such as ping and telnet. Yet any network that grows beyond just a few components requires advanced analysis capabilities that simply aren't available with these basic toolsets. When users call to inform you that "the network is slow" today, you need additional functionality that clues you in to the problem's root cause.

Traditional tools to accomplish this deep level of troubleshooting have often focused on packet analysis and inspection. However, the process of finding meaning in individual packets is exceptionally difficult for all but the most experienced of network engineers. Additionally, these tools aren't designed to give that engineer a high-level view of network behaviors. Their low-level perspective simply doesn't provide the kind of metrics that explain how data moves around the network in aggregate.

Needed is that higher level of perspective. With such a perspective, a network engineer can look at high-level flows to quickly isolate bandwidth utilization and identify traffic behaviors based on ports, protocols, endpoints, and even individual networked applications. That high-level perspective is gained through a suite of protocols commonly referred to as NetFlow.

Understanding Network Flows

To begin, consider first what a network "flow" is. A flow is identified by combining a set of key fields from a stream of network packets. Those key fields tend to include the following:

- Source and destination IP address
- Source and destination port number
- Layer 3 protocol designation
- Type of Service (ToS) byte
- Logical interface index

Any network flow by definition will be comprised of all network packets that have the same information in each of these fields. For example, if a network service on one server is communicating with another, that service's communication will tend to occur between those two computers, over an unchanging port, using an unchanging protocol and ToS, and via the same logical interface.

Once a set of data that contains these same seven elements is gathered, four sets of statistical information can generally be acquired:

- System uptime at start of flow
- System uptime at end of flow
- Number of packets in flow
- Number of bytes in flow

Although small in count, these four pieces of information illuminate a large amount of detail about the conversation between the two computers. The rate of data exchange can now be measured. The time it takes for communication to occur as well as the amount of time that the two computers spend communicating with each other can similarly be gathered. The sheer amount of data being transferred during that time is another important metric gained through this measurement.

Obviously, NetFlow information must occur from the perspective of the device that is sending and/or receiving the packets. Thus, any flow-based information must be viewed with recognition of its measurement source. For example, if a network spans three geographically-dispersed sites, flow-based metrics between two computers are likely to be drastically different if it is measured in Site A versus Site B or C.

NetFlow, J-Flow, sFlow, and IPFIX

Industry vernacular uses the term “NetFlow” generically, but the NetFlow protocol is actually a development of Cisco Systems. As such, any Cisco-based devices will use the NetFlow implementation of flow-based analysis. Four versions of Cisco’s NetFlow protocol remain in use today, with two rarely seen in today’s business networks:

- **NetFlow Version 5**—Originally developed by Cisco Systems but currently in use by other vendors including Adtran
- **NetFlow Version 7**—Rarely seen today; specific to Cisco Catalyst switches
- **NetFlow Version 8**—Also rarely seen today, as it has been superseded by version 9; version 8 introduced aggregation technology
- **NetFlow Version 9**—Most common version in deployment today; includes mainstream availability of earlier-introduced aggregation features while introducing flexible NetFlow concepts

Although Cisco Systems is credited with developing the first implementation, other vendors have developed their own variants of the architecture for use within their hardware. These alternative implementations enable a similar set of operational functionality but with each supporting their own unique feature set. Three major examples include:

- **J-Flow**—Developed by Juniper Networks for use in their hardware; effectively the same as Cisco NetFlow Version 5
- **sFlow**—A standards-based implementation (RFC 3176) whose development is shared by HP, Extreme, Foundry, Juniper, and Nortel, sFlow is unique in that its measurements are based on a statistical sampling of flow data, which has the effect of reducing the total amount of data that is required to be sampled to achieve a statistically similar result
- **IPFIX**—Commonly considered the next version of NetFlow, or NetFlow Version 10, IPFIX is based on Cisco NetFlow Version 9; among other capabilities, IPFIX offers template-based exporting of data

Leveraging NetFlow

The power of the flow-based protocols is in their ubiquity. As with the Simple Network Management Protocol (SNMP), the necessary code and processing to enable and use flow-based protocols is already included with virtually all network hardware available today. Thus, existing hardware components need only have NetFlow configured and enabled to begin enjoying its analysis capabilities. As with Syslog configurations, different network device Operating Systems (OSs) will have different mechanisms for enabling the export of flow information.

For a Cisco IOS router, the configuration steps might resemble the following example:

```
router#enable
Password:*****
router#configure terminal
router-1234(config)#interface FastEthernet 0/1
router-1234(config-if)#ip route-cache flow
router-1234(config-if)#exit
router-1234(config)#ip flow-export destination 192.168.1.100 9996
router-1234(config)#ip flow-export source FastEthernet 0/1
router-1234(config)#ip flow-export version 5
router-1234(config)#ip flow-cache timeout active 1
router-1234(config)#ip flow-cache timeout inactive 15
router-1234(config)#snmp-server ifindex persist
router-1234(config)#^Z
router#write
```

These steps enable the export of flow information on the interface FastEthernet 0/1 to be directed to the server at 192.168.1.100 over port 9996. This configuration must be enabled for each of the interfaces on which flow information should be exported.

Obviously, configuring NetFlow information across the network devices and interfaces in your environment only accomplishes one half of the setup. The level of information being gathered by a NetFlow-enabled interface is large and requires additional calculation by a server at the target endpoint if its data is to be useful to an administrator. Thus, the job of that server is in calculating the data, measuring it against other inbound flow information, and creating visualizations that display actionable information.

Free Tool: SolarWinds NetFlow Configurator & Real-time Netflow Analyzer

If the previously mentioned process to configure a device for NetFlow appears challenging, be aware that tools exist to automate the process. SolarWinds' NetFlow Configurator is a free tool that leverages SNMP to configure NetFlow on supported devices. Statistics from NetFlow-configured devices can be analyzed using SolarWinds' free Real-time NetFlow Analyzer tool. Both tools can be downloaded from the SolarWinds Web site at <http://www.solarwinds.com/products/freetools/>.

NetFlow in Today's Network Management Solutions

Getting the best data about your network infrastructure means leveraging modern protocols such as NetFlow and its platform-specific variants. Free tools such as the SolarWinds NetFlow Configurator and Real-time NetFlow Analyzer bring a limited but useful view into those metrics. With these tools, it is possible to visualize network traffic flows across a single configured device in real time.

For large-scale requirements, other more comprehensive tools are often necessary. Orion NetFlow Traffic Analyzer (NTA) greatly expands your vision beyond the range of free tools through both real-time and historical views of flow-based data. The result is a quick implementation, gathering the right level of data to improve troubleshooting and gain a better understanding of the types and levels of data passing across your distributed network.

Volume 5: Cisco IP Service Level Agreements

The final management protocol to be discussed in this Reference Guide is a proprietary feature that is included with the Cisco IOS. This feature, called IP Service Level Agreements (IP SLA), represents another mechanism by which statistics and other performance and utilization metrics can be measured across network wires.

Although the information gained with IP SLA is often used for the same types of troubleshooting and analysis activities as is done with NetFlow data, the actual information gathered is fundamentally different. With NetFlow, individual devices gather aggregate statistics across individual interfaces, presenting the results (also in aggregate) to a centralized server for processing. This kind of data can be considered volume-based data because it illuminates the behaviors on a network from a very high-level perspective.

The data gained by enabling IP SLA metrics is different in that it represents the results of a series of ongoing “tests” that are preconfigured for specific network devices. These periodic tests are used to validate connectivity as well as evaluate instantaneous performance metrics across specified interfaces. The “SLA” in “IP SLA” refers to this protocol’s capability to automatically and repeatedly test whether a particular network behavior is functioning to expected levels.

For example, business networks that incorporate Voice over IP (VoIP) services into the network require an awareness that their users’ quality of service remains at a high level. When the network cannot keep up with the needs of its VoIP users, the call quality for all users suffers. Although NetFlow information provides an understanding of the volume of data occurring across different devices on the network, additional test data is useful for repeatedly measuring and reporting on that user experience.

For example, in the case of VoIP services, the level of User Datagram Protocol (UDP) jitter across the network is one metric whose value can be directly translated into call quality. UDP jitter occurs when latency across network connections occurs but not at a constant rate. Due to the nature of VoIP services, jitter’s rapid shifting of connection latency has a dramatic effect on call quality. To identify when this behavior goes beyond acceptable thresholds, an IP SLA test can be configured to measure jitter between two endpoints.

This single test helps the administrator understand the behavior that occurs at that single location. But where IP SLA’s greatest power comes is in the ability to synchronize similar tests all across the distributed network, effectively showing the same metric across the entire environment at once. For a distributed application such as VoIP, this widespread vision assists network administrators with tracking down network problems as well as their locality.

Precise Timing Is Critical

IP SLA's capability to distribute test loads directly to the devices that participate in networking means that resulting statistics will be submitted to the Network Management Solution (NMS) from multiple locations at once. Thus, the maintenance of precise time across all devices is exceptionally important. The central NMS that gathers these statistics must rely on submitted time across all devices to analyze inputs in generating visualizations. As such, incorrect time across any device on the network will inadvertently skew IP SLA results.

IP SLA Extends Traditional Network Monitoring

This concept of being everywhere at once goes far into extending the reach of traditional network monitoring devices. In traditional network monitoring, the architecture assumes a type of hub-and-spoke methodology. In the hub lies the NMS and radiating away as spokes are that NMS' polls outward to individual devices.

This solution functions acceptably for simple networks—those that don't spread across multiple sites or geographic locations. However, the traditional hub-and-spoke methodology sees its limitations when attempting to measure performance from one site to another. This setup is particularly problematic when the central NMS does not exist in either of the two sites to be measured. Also challenging are network architectures that do not route Internet access through a single, central connection. When individual sites connect via Multi-Protocol Label Switching (MPLS) connections and/or leverage their own Internet connections, gathering the right kinds of statistics is challenging using traditional methods.

One early mechanism for circumventing this limitation was the use of hardware probes, whose installation required a physical connection to the network they were charged with monitoring. Network probes passively monitor data across the network to which they're connected, submitting metrics back to the central NMS. Because probes operate in-line with the network, the behaviors they see relate to the individual network segment where they're installed.

Yet the fact that hardware probes are physical devices adds a significant level of administrative overhead to their use. Installing the probe means being physically present at the monitoring location. Purchasing probes involves cost, which scales linearly as more probes are needed. Ultimately, although probes can provide the level of detail necessary to measure the aforementioned metrics, they do so with an accompanying burden.

Like the code base that enables NetFlow metrics collection, IP SLA's code base is mature and likely already installed into network equipment. Thus, enabling it for use involves little more than reconfiguring network devices, determining the tests to be run, and pointing the results to an NMS for processing. For example, to configure a basic Internet Control Message Protocol (ICMP) ping test to occur every 30 seconds between the network device and the remote IP address 192.168.1.100, a command structure similar to the following could be used:

```
Switch(config)# ip sla 1
Switch(config-ip-sla)# icmp-echo 192.168.1.100
Switch(config-ip-sla-echo)# frequency 30
Switch(config-ip-sla-echo)# exit
Switch(config)# ip sla schedule 5 start-time now life forever
Switch(config)# end
```

In this structure, the ICMP test sourced from the switch and with the host 192.168.1.100 as the target is configured to occur every 30 seconds and run forever. IP SLA information is passed back to a central NMS through the same channel used by Simple Network Management Protocol (SNMP). As such, the device's SNMP configuration must be enabled for IP SLA information to be submitted.

Free Tool: SolarWinds IP SLA Monitor

Although a fully-featured NMS is required for support of the entire range of IP SLA tests, often only a small sample of tests is necessary. For those limited uses, SolarWinds provides a free IP SLA Monitor tool that can be downloaded from http://www.solarwinds.com/products/freetools/ip_sla_monitor/. This tool configures and enables IP SLA on Cisco routers and switches, configures the test details for each device, and displays the resulting information in a useful heads-up display.

IP SLA Responders

A simple ICMP test with IP SLA requires very little for its full functionality. Its ping response statistics can be trivially submitted back to an NMS with no additional numerical processing. However, some network tests require the presence of a partner on the network to validate data transmission statistics. The aforementioned jitter test is one example where that second party in the test is required. In IP SLA terms, this second party is referred to as the IP SLA Responder and is manifested as an additional component in the IP SLA code base that is already present on network devices.

To participate in the test, an IP SLA Responder must be configured in much the same way as the IP SLA test. Be aware that the IP SLA Responder code base is present in only some classes of network devices, limiting where Responders can be enabled in the infrastructure.

In the case of the jitter test, a Responder is required due to the nature of the test itself. A network device can require tens of milliseconds to process incoming packets. This can be due to the regular processing of network traffic or because of other high-priority processes enabled on the device. This added delay will inadvertently skew jitter statistics without compensation and subsequent recalculation by both sides of the test. Configuring a responder for such a test can be done with a command structure similar to the following example:

```
ip sla responder udp-echo 192.168.1.100 5000
```

IP SLA and Test Processing

Although IP SLA's instrumentation is present in the network device itself, the processing of its data generally requires the support of an external server. As with NetFlow data, the role of an external server is to receive inbound IP SLA metrics data, calculate useful metrics, and ultimately create visualizations from those metrics.

This capability is particularly necessary with IP SLA environments as its data can be arriving from multiple locations at once via SNMP traps. With the instrumentation in the individual network device, that device's job is to measure and report. Meaning from that data typically arrives when that individual sensor's data is correlated with others across the network. Upon receipt of metrics, calculations are required by the NMS to generate the right statistics for use by the administrator.

An additional requirement of the NMS is in the maintenance of test characteristics. Although the previous example shows how a single test can be created between two endpoints, IP SLA is designed for scalable use across a multiple-site WAN. Thus, although creating a single test can consume just a few lines, re-creating that test across multiple devices in multiple locations adds an administrative burden as the environment scales. Combining multiple tests with multiple devices means a geometrically-increasing number of configurations to manage and maintain.

IP SLA in Today's NMSs

SolarWinds Orion Network Performance Monitor (NPM) and Orion IP SLA Manager are two solutions that combine the correlation and visualization capabilities of an NMS with the configuration control needs of that NMS' administrators. Both the configuration and the data transfer of metrics for IP SLA tests occur through SNMP. Thus, once a network device is configured for use within Orion NPM, it is possible to quickly distribute a test to that device through its SNMP channel. At the same time, the SNMP connection between device and NMS provides a location to send resulting test data for processing.

Once tests are running and data is being submitted, Orion NPM provides a suite of visualizations that integrate with built-in maps. These maps enable Orion to display information about tests across a range of sites, allowing administrators to drill down to problem areas with a few clicks. By controlling both the test configuration and the resulting visualization, SolarWinds Orion provides a single location for monitoring and managing the entire distributed network. In addition, Orion IP SLA Manager enables you to monitor key WAN applications by analyzing the performance of the underlying network protocols, including DNS lookups, FTP, HTTP, TCP connect, and UDP jitter. Lastly, you can continue to monitor VoIP call paths to ensure quality of service for your voice traffic on your network with Orion IP SLA Manager.