SNMP – Whats Next?

Geoff Huston
April 2002

In the previous article we looked at the vulnerabilities that have been exposed in a number of commonly used implementations of the Internet's Simple Network Management Protocol, or SNMP. The problem that was exposed was one where poorly formatted SNMP data packets were still accepted by the SNMP process with a network device, and in processing the packet, the SNMP process managed to corrupt to host system's memory maps. The host device then failed in a number of quite interesting ways, some of which caused the host device to continuously reboot, or allowed the host device to be 'captured' by code contained within the offending SNMP packet. While these vulnerabilities are serious, they are the outcome of weaknesses in the implementation of the protocol, and, as such, they have not exposed fundamental flaws in SNMP itself. One could believe that the architecture of SNMP remains intact.

In spite of this assurance there remains a lingering doubt about SNMP. Maybe SNMP is simply not the best tool we can invent for managing networks, and if we thought about it some more, then just maybe there is something better that we could devise. It's an interesting thought, and in this article we'll look at where there are weaknesses in the architecture of SNMP look at what we'd like to see in the next generation of SNMP.

To recap over SNMP for a brief moment, the architectural model of SNMP is one that assumes a managed network device with relatively limited capabilities and a network management entity that is highly capable. The managed device continuously maintains a collection of registers, and the manager periodically requests the current values of some of these registers from the device. In addition, the manager can change the configuration of the managed device by sending it new values for some registers. It's a "just in case" management model, where the managed device continuously maintains a large set of counters just in case a management station may poll for the value of a counter, and the management station polls a device's register just in case the device is experiencing abnormal conditions at the time. For network management "just in case" is not enough in terms of functionality. There is a need to send an "attention alert" from a device to a management station. So, to complete the SNMP model, the device can send an alert to a management station when a register undergoes a certain state transition or value change.

This SNMP model is certainly quite simple and quite flexible, as has been shown by the plethora of devices that are managed by this model a decade after the introduction of SNMP. But there are also other protocols and other approaches that are starting to crowd in on SNMP, highlighting the fact that SNMP is not a network manager's panacea.

SNMP is quite poor at sending large quantities of data. Large data sets are commonly found in tables, such as, for example, the routing table of an Internet router. Large data sets, or tables, are somewhat of a weakness in SNMP. It's recognized that tables are essential when a common set of information elements are to be associated with each element of a collection of objects. And because the size of the collection of objects may vary dynamically, the size of a table may vary dynamically. Can a management station determine how large a table is before it attempts to retrieve it using SNMP? Sorry, but no. Such information is not available to the management station within the definition of SNMP itself, and it may or may not be contained in the Management Information Base, or MIB, together with the table itself. Commonly, with SNMP, you know get to know how big a table is only when you've retrieved the entire table and overshot off the other end. This is a classic example of only finding out what you needed to know only when its way too late to be of any value!

Even retrieving the table is not an efficient task in SNMP. To retrieve a table, the basic approach is for the network manager to request, or GET, the first item in the table, and wait for a response, then use the SNMP GETNEXT operator to request the second item, wait for a response, and so on. Each successive request awaits the outcome of the previous request. If any request packet, or the corresponding response packet, is lost within the network then the application will need to await a timeout before generating a repeat request for the data. Those readers familiar with data protocols will see this as a degenerate case of a sliding window protocol, where the sliding widow is of size one. For any practical purpose, this approach is simply not an efficient transport protocol, and changes to the protocol had to be made. The refinement with version 2 of the SNMP protocol was to allow a larger number of items to be placed in each packet, without altering the basic operation of the protocol itself. This GETBULK operator allows a sequence of items to be placed in each response, but the basic transport protocol, that of a single packet exchange, was unaltered.

It would seem that at the very least, this is a case for including the TCP protocol in SNMP, allowing large data sets to be shifted through the network using a congestion-controlled efficient transport protocol. Interestingly, this topic, SNMP using TCP, has been under consideration as a research topic for some years. Experimental implementations of SNMP over TCP have been circulated for many years now, yet it has not generated any momentum from the vendor and operator community. It's a pity, because it could make SNMP useful within a larger domain of applications.

Instead of using SNMP to retrieve large volumes of data from a managed device, many network managers use TCP by launching a script which opens an interactive session with the managed device and then issues a command to the command level interface (CLI) of the device to generate the relevant response. The use of the CLI as a network management tool is not limited to retrieving large data sets. The CLI is also strongly preferred as a configuration tool over SNMP SETs by the operator community, despite a number of attempts to introduce various alternate forms of configuration control which used SNMP SETs as the communication mode between the network management station and the router. Maybe network operators come from an old-fashioned world where ASCII still rules supreme. May be its just faster to use a language-based interface to issue configuration commands to a device than remember sequences of device register names and associated values. For whatever reason, SNMP SETs are not the protocol weapon of choice for device configuration.

For dynamic configuration, such as found in network access servers and differentiated service network access units SNMP has also failed to gain traction. Here, the configuration weapon of choice appears to be a protocol termed the "Common Open Policy Service", or COPS. There are certainly some similarities between COPS and SNMP, and possibly enough to ask why two solutions are required when one could do. But COPS is different enough to have a continuing existence. COPS uses a different network management model where it is envisaged that the device's managed configuration will undergo continual incremental change in response to user's individual requirements. To achieve this incremental update the network management system uses a local copy of a configuration state for each managed device, and incremental updates are sent to each managed device whenever the configuration state is altered. This introduces a new concept into the network management architecture, that of a synchronized state between the management station and the managed device. If the managed device is reset, the management station should be aware of the need to reload the configuration state, something that is not intrinsically easy in SNMP. COPS also supports resource locking, allowing multiple management stations to operate without corrupting the managed devices with conflicting updates. SNMP is also criticized because of its use of UDP as a transport protocol in this context. It is noted that complex policy updates may require a sequence of updates, and a reliable transport protocol, such as TCP, allows the policy update to be conducted over a shared state between the managed device and the management station. The COPS protocol addresses these concerns through the use of a single control channel with a shared state supported by a single TCP session.

The SNMP security story has always been a point of some contention. The use of passwords in the clear and data in the clear in SNMP version 1 is certainly simple, but perhaps it is a little too open for many secure management applications. The extreme alternative is to use out-of-band network management, and install an entirely distinct network that only carries management traffic. If “good” and “cheap” are at opposed ends of the spectrum, then some level of compromise is called for. A more modest approach is to admit slightly greater levels of capability in the management model to allow a network management station and a managed device to validate each other's identity, and then conduct a conversation using some level of encryption of the data. SNMP version 3 allows for multiple security models, including user-based security. In this model unauthorized entry is protected, as are attempts to modify SNMP data, replay of SNMP requests and disclosure of data. The only problem with this is that it may just be too late, as there is little noted vendor or customer urgency to see SNMPv3 deployment, and without v3 deployment, SNMP security will continue to be an issue.

It is useful to look around and see what is happening in other realms. Network Management is a shared data application, and one of the issues with shared data systems is how to create a framework where all parties share a consistent interpretation of the data. SNMP's use of ASN.1 as a data description language illustrates its now venerable age. SNMP assumes that both the sender and receiver are aware of how parse the data, and the data is sent in its encoded form without any accompanying reference to the definition of the data encoding. These days there is considerable push to use XML and the Simple Object Access Protocol (SOAP) in distributed data systems. SOAP uses XML to allow the data definition to be referred to with the data objects, allowing the object to contain both a reference to the definition and syntax of an object as well as its value. The object is now self-contained as it now contains both the data and information on how to interpret the data.

The other observation about the current state of the art comes from distributed object-oriented computing environments. Here there has been significant development in distributed object-oriented computational techniques. Network management should be more than a polling system of paired simple queries and simple responses where an "intelligent" management station repeated polls a collection of "stupid" managed devices. There is an abundance of processing capability on almost any silicon device, and there may be some value in altering the network management model to one where the management station and the managed device cooperate on defining what information is of interest, and under what circumstances. One way of looking at this altered management model is illustrated by the approach of taking some of the management station's processing rules that it uses on the gathered data and passing these rules to the devices, allowing the devices to respond with processed information at times determined by these processing rules. It takes network management away from the task of mindless reporting of current conditions to one of enabling every device to differentiate between a normal operating condition and an exceptional circumstance, and allowing the network management task to focus on those conditions which are regarded as out of the ordinary.

Extending the network management model is also not just about extending the protocol elements and the data models. SNMP is a device-oriented network management architecture, where the network is seen as an arbitrary collection of active devices. However, a network is a service platform, and the ultimate focus of network management extends beyond the management of individual devices to the task of management of the parameters of the service. Increasingly customers of an Internet Service provider are interested in service metrics such as latency, or jitter signatures from a service provider, or at a higher level in the service definition spectrum, the sustainable quality of a voice over IP stream, or the quality of multimedia streaming sessions. The task of instrumenting the network to be able to report on the service levels that are achievable from applications that use the network remains perhaps the most challenging tasks confronting network management.

I suspect that what we are talking about is more than a few tweaks to SNMP. It seems that we have to stop thinking of devices as being equipped with a tiny slow processor and very limited

amounts of memory, and allow the managed device to undertake the primary processing of environmental data. This, in turn, allows the management model to move away from a “just in case” extravagant model of data collection and processing to one of meeting a well-defined set of operational requirements. Perhaps it’s a case of taking a broader view of network management and instead of asking the operators for suggestions about more counters to include in the management model, ask the researchers for ways to winnow down the raw quantities of data while at the same time produce management outcomes that concentrate on maintaining the quality, consistency and efficiency of the delivered network service.

I suspect that there is much yet to happen with network management and SNMP in the coming years.