Enabling the Management of Everything Using TEMP

Geoffrey C. Carpenter
IBM T. J Watson Research Center, P. O. Box 704, Yorktown Heights, NY 10598 USA
(email: gcc@watson.ibm.com)

Abstract

The Enterprise Management Protocol (TEMP) and its affiliated infrastructure provide a highly scalable, transport protocol-independent base for enabling the management of networked devices, systems and, most importantly, applications. TEMP supports automatic discovery, interactions between incompatible protocol domains, multiple yet independent entities on a single host and self-describing instrumentation. These features enable the implementation of generic monitoring applications as well as access to entity-specific management interfaces using a conventional World Wide Web browser.

1. Introduction

Shortly after the publication of version 1 of the Simple Network Management Protocol (SNMP) [1], efforts were undertaken to utilize the then new protocol to manage applications and existing devices that had no SNMP support available. The earliest effort was the SNMP Distributed Program Interface, a protocol and API specification that was implemented for IBM's TCP/IP products in 1989 and later documented in RFC 1228 [2]. Additional DPI-like subagent technologies were developed and today the Internet Engineering Task Force Agent-X working group is trying to publish a DPI version 2-derived standard for the SNMP community.

Despite that promising start and a decade of availability of subagent technologies, the dearth of manageable applications is quite telling. Experience demonstrated a considerable disconnect between the information model imposed by SNMP and the underlying implementation of virtually any program (e.g., operating system kernel, protocol stack, application, etc.). Consider SNMP's requirements for:

- Lexicographic ordering: most data structures maintained by a program are not in sorted order, which requires either significant modification of the program or expensive on-demand conversions from more appropriate data structures (e.g., hash tables).
- Scalar-only data types: only the most trivial of programs utilize scalar-only data. If a program is sophisticated enough to offer remote management capabilities, it probably uses data structures at least as sophisticated as an array or list. Modeling these oft-used data structures as scalar-only elements is an unnatural task.
- OIDs to be assigned to identify each item of data: rarely is an OID a natural way to name a piece of data. The requirement to coordinate with a higher-level authority to obtain portions of the OID space is an additional inhibitor.
- Using a well-known port to address an agent: this leads to the use of subagent technologies. They create two major difficulties: a prerequisite that the agent on the target host supports the chosen subagent technology and the robustness of an application is affected by all other subagents attached to the master agent.

Further complicating matters, SNMP provides no underlying support for automatic discovery of manageable entities. This creates a considerable problem when attempting to develop programs that are able to manage applications that can come and go or even be instantiated as multiple copies running simultaneously on a host. Finally, for the purposes of the discussion below, SNMP is a polling-based protocol and lacks any guarantees on the delivery of events. This renders it increasingly ineffective as one moves beyond managing relatively small numbers of core network elements, such as routers, and the health of a few important shared resources, such as servers.

In response to these issues, The Enterprise Management Protocol (TEMP) was developed and utilized to manage new technologies, such as Narwhal [3], as well as existing applications, such the Apache web server [4].

2. An Overview of TEMP

TEMP-based enties are broadly classified into two distinct groups: management applications and agents. While management application is a self-describing term, the term agent refers to any entity that provides access to the instrumentation of a device, host or application. The group of TEMP agents contains a subset known as TEMP registration agents. TEMP registration agents are

differentiated from regular TEMP agents only by the fact that they bind to a well-known transport address.

2.1. TEMP Design Goals

Given the simple model of management applications and agents (some of which run at well-known transport addresses), TEMP was designed to meet several goals, one of which was to be secure. A list of other goals includes the following points:

- Highly scalable.
- Support for mixed-protocol environments.
- Support for multiple independent entities running on a single host.
- Support for automatic discovery and administrative domains.
- Support for non-scalar data.
- Support for self-describing instrumentation and management interfaces.

These major design points are elaborated upon below.

2.2. Scalability

As a rough estimate, the monitoring and management of all of the devices, hosts and applications in an enterprise is a problem two orders of magnitude greater than that dealt with by conventional network management systems. Obviously, a small enterprise with only four hosts is not going to strain the capabilities of any commercial management platform, but one with over 100,000 hosts is in a different league.

TEMP is datagram-based, eliminating the scaling problems created by session-oriented protocols. It is also an event-driven protocol, so it does not require the polling of individual entities. Dropping the polling requirement eliminates 50% of the normal traffic overhead as no request need be sent from a management application to a TEMP-based entity for verification of aliveness or detection of a significant status change. Instead, TEMP-based entities emit a periodic heartbeat to announce their existence. These heartbeats are called STATUS messages in the protocol specification [5].

All TEMP entities maintain an event history, which is of some finite length. The STATUS messages include the indices of the start and end of the event queue. A management application can determine if something interesting has happened by comparing the end-of-queue index with a value that it has previously recorded for the remote entity. If the value has changed, the management application can obtain all of the events in the queue that it has not seen using a single request. Benefits of this behavior include:

- The possible outage of a TEMP-based entity can be detected by the failure to receive a STATUS message within the guaranteed heartbeat interval.
- Events become reliable since, while STATUS messages can be lost, eventually one will be received and the management application can obtain all of the events that have been queued. This also permits a newly started management application to obtain a history of events that occurred when the management application was not operational.
- Multiple management applications can retrieve events from a TEMP-based entity without interfering with other applications because they do not modify the event queue in any way.

2.3. Mixed-Protocol Environments

All TEMP entities are identified by a Universally Unique Identifier (UUID) instead of by transport addresses. This provides several benefits, such as:

- Entities residing on multi-homed hosts always appear as a distinct entity, not as multiple entities corresponding to each interface address. This eliminates the need for management applications to discover and maintain tables that map multiple transport addresses to a particular agent record.
- UUIDs can be either pre-defined or dynamically generated as called for by the deployment situation. Among other things, this makes it trivial to identify a new instance of an application that reused a particular transport address
- Separation of entity identification from transport address allows inter-protocol domain routing to take place. This permits entities operating in incompatible protocol domains to interact as long as there is one TEMP-based entity running on a host capable of communicating using both transport protocols.

2.4. Multiple Entities on a Single Host

With the exception of TEMP registration agents and potentially some TEMP-based management applications, TEMP entities use dynamically assigned transport addresses. Because they do not use well-known transport addresses, there is no conflict with multiple applications running on a host competing for the same well-known transport address. This results in the elimination of a need for any subagent technology. As noted above, eliminating the need for using a particular subagent technology means that:

- An application can be deployed without regard to the subagent technology available on a particular host.

- An application's performance and robustness cannot be influenced by the misbehavior of another application connected to the same master agent.
- Distinct copies of a specific application can be run on a host without requiring coordination to generate unique names for each application instance.

2.5. Automatic Discovery and Administrative Domains

TEMP inherently supports the automatic discovery of TEMP-based entities. In contrast to SNMP, where a specialized management application has to probe the network in an attempt to discover reachable SNMP agents, TEMP agents register themselves with TEMP registration agents.

Unlike devices closely associated with the underlying network, where knowledge of physical interrelationships provides significant information, management of host-based services and applications are often more appropriately grouped or displayed in tems of their administrative relationships. Each TEMP entiity can be a member of multiple administrative domains and this membership is part of the information announced by every TEMP entity. As an illustration, this feature allows a view of a set of DNS servers to be displayed although the individual servers may be in different physical locations.

These two capabilities allow very efficient and reliable discovery of the active TEMP-based entities as well as their administrative relationships.

Note: well-known and critical elements can be permanently assigned a UUID (instead of using a dynamically generated UUID) and this information can be associated with static topology data. This approach allows a management application to report on devices that are down when the application begins operation.

2.6. Complex Data Types

In addition to the conventional scalar data types, such as integers, floating point numbers and octet strings, TEMP supports complex data types such as sparse arrays, associative arrays and ordered lists of elements. These data types map naturally to a wide range of data structures utilized in most programs.

2.7. Self-Describing Instrumentation and Management Interfaces

A major goal for TEMP is to eliminate as much as possible the use of agent-specific monitoring or management applications. One significant step TEMP takes towards this goal is the use of self-describing instrumentation. Consider the current situation with most SNMP agents: a vendor often provides a large number of device-specific variables in an attempt to differentiate the product in question. While potentially useful, one of the most frequent questions posted to SNMP-related mailing lists is on the order of "what variables should I monitor?". In contrast, TEMP agents make available a list of useful variables to monitor and their associated thresholds. This well-defined structure allows a generic management application to interact with an arbitrary TEMP agent and perform some useful analysis of what has gone wrong when a problem is reported.

Another problem for vendors is the implementation of agent-specific operator interfaces. Often this has required choosing a particular mangement platform and writing platform-specific applications that are able to access the management capabilities offered by the agent.

TEMP takes a different approach that eliminates the need for a vendor to choose to support a particular management platform. A TEMP agent can return HTML fragments that implement an operator interface that is accessible via a conventional World Wide Web browser. It is important to note that unlike other Web-based management approaches, TEMP entities do not respond to HTTP requests (unless that is the purpose of the TEMP-enabled application, like a HTTP server). This means that in a TEMP-based environment, an off-the-shelf browser can obtain management interface pages from a TEMP agent running in a SNA environment, even if that means traversing an IPX domain in the process.

To enable this functionality, an HTTP server is configured with a set of CGI executables and an HTML page (to provide an obvious starting place). The operator browses the HTML page that provides a link to access the TEMP-based management functionality.

The CGI scripts are invoked as appropriate by the HTTP server and they communicate with TEMP-based entities using TEMP, not HTTP. When a TEMP agent is queried for a variable that provides a management interface, the returned HTML fragments are potentially rewritten on the fly before being forwarded via the HTTP server back to the web browser. The rewriting capabilities work in both directions, allowing, among other capabilties, a retrieved HTML fragment containing FORMs to be modified based on data entered into fields. When data that is not HTML fragement is returned via a HTTP<->TEMP gateway (like integers, floats, arrays, etc.), the results are converted into formatted text and displayed as a page. This provides a simple variable dump facility.

2.8. TEMP Protocol Elements

TEMP has several command elements defined for the protocol. The are the minimal set of directives found to be useful to date are:

- RETRIEVE (an existing variable)
- MODIFY (an existing variable)
- CREATE (a new variable)
- CREATE-OR-MODIFY (a variable)
- DELETE (an existing variable)
- LIST (elements under a particular point in the naming tree)
- BEGIN-BLOCK (to introduce a sequence of commands)
- END-BLOCK (to end a sequence of commands)
- IF (conditional execution of commands)
- ADD-ELEMENT (new element into a complex variable)
- REMOVE-ELEMENT (from a complex variable)
- REPLACE-ELEMENT (an existing element of a complex variable)
- ADD-OR-REPLACE-ELEMENT (in an existing complex variable)

A TEMP agent is able to send the following packets to a manager:

- RESPONSE (to a RETRIEVE, MODIFY, CREATE or DELETE request)
- STATUS (an unsolicited notification)

The IF directive, combined with appropriate BEGIN-BLOCK and END-BLOCK directives, can be very powerful when used in conjunction with broadcast or multicast packets. For example, the execution of particular command elements can be conditionally performed based on the type of the device or other arbitrary criterion. Less obvious is the potential for use in specialized broadcast-only media, such as the vertical blanking interval of a television signal.

3. Experience with TEMP

TEMP has been in use for about a year as the management protocol for Narwhal, a new technology that provides numerous additional capabilities for applications that utilize intermediaries (e.g., SOCKS servers or HTTP proxy caches). Narwhal Client Agents are intended to be deployed on every host in an enterprise and are actively managed. Such active management enables resource resource reservation, the proactive communication of information regarding poorly performing intermediaries, and automatic configuration of mobile clients. The nature of the application as well as the number of deployed units creates a problem for which TEMP is uniquely suited.

TEMP has also been used to add management capabilities to the Apache web server as part of an ongoing project to add management capabilities to key parts of the Internet infrastructure.

4. Future Work

A significant amount of existing instrumentation has been deployed using SNMP agents or the Desktop Management Interface [6]. Such instrumentation represents a notable investment on the part of various vendors. It would be impractical to ignore this deployed infrastructure; consequently, gateways between TEMP and SNMP and TEMP and DMI-acessible data are under development.

Discussions are also underway to utilize TEMP in the management of Internet-2-related services and applications.

5. Conclusions

TEMP is a new protocol and associated infrastructure that is intended to address the deficiencies of existing approaches that have prevented the widespread development of manageable applications.

TEMP supports automatic discovery of manageable entities and the development of generic management applications through its use of self-describing instrumentation and Web-accessible management interfaces. TEMP enables management interactions between incompatible protocol domains and has very attractive scaling properties.

For the application developer, TEMP provides several advantages. It is designed to eliminate the need for any subagent technologies and the TEMP data model maps naturally to the data structures used by most programs.

6. References

[1] J. Case, M. Fedor, M. Shoffstall and J. Davin., "A Simple Network Management Protocol", *RFC 1067*. http://www.ietf.orf/rfc/rfc1067.txt, 1998.

[2] G. Carpenter and B. Wijnen, "Simple Network Management Protocol Distributed Program Interface", *RFC 1228*, http://www.ietf.org/rfc/rfc1228.txt, 1991.

[3] G. Carpenter and G. Goldzsmidt, "Improving the Availability and Performance of Network Mediated Services", anticipated publication in INET'99 conference proceedings.

[4] Apache web server, http://www.apache.org

[5] G. Carpenter, *The Enterprise Management Protocol*, Unpublished IBM Research report, 1998.

[6] Desktop Management Interface, http://www.dmtf.org/spec/dmis.html.