Remote Management of Narwhal Client Agents using TEMP

G. Carpenter
FARGOS Development, LLC
757 Delano Road
Yorktown Heights, NY
USA
geoff@fargos.net

G. Goldszmidt (Contact Author)
IBM Research
30 Saw Mill River Road
Hawthorne, NY
USA
gsg@us.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. The functionality and benefits of TEMP are illustrated through a case study of the remote management of Narwhal Client Agents. Narwhal is a technology that provides a local (i.e., client-resident) intermediary broker that can intelligently route the traffic among intermediaries. Its benefits include (1) improved availability of intermediary services, (2) load sharing requests across several intermediaries, (3) bypass intermediaries whenever possible, and (4) remote administrative control enabling the implementation of domain-specific policies to utilize the shared, limited networking resources.

Keywords

Management Protocols, Distributed Management, Availability, Scalability, Load Sharing

1 Introduction

The Enterprise Management Protocol (TEMP) [5] was developed to address some of the deficiencies of current management protocols, particularly with respect to their use in the remote management of applications. Soon after the publication of version 1 of the Simple Network Management Protocol (SNMP) [1], attempts were made to utilize it to manage applications and existing devices. For example, the SNMP Distributed Program Interface (DPI) appeared in IBM's TCP/IP products starting in 1989 and was eventually publicly documented as RFC1228 [2]. Efforts to develop a common DPI version 2-derived subagent technology continue today in the Internet Engineering Task Force Agent-X working group. However, over a decade of experience has demonstrated the deficiencies of SNMP.

Developers wishing to enable their application for remote management face several issues when attempting to use an SNMP subagent technology:

- 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. Application programs that offer remote management capabilities are likely to use 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 must 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 naming authority to obtain portions of the OID space is an additional inhibitor.
- The need for a subagent technology creates two major difficulties: a prerequisite that the agent on the target host implements 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. TEMP was designed to address each of the problems noted above and others.
TEMP has been used to monitor and control applications that provide services. Examples of applications that have been TEMP-enabled include Apache web servers [3] and Narwhal Client Agents[4]. Narwhal is a new technology that provides numerous additional capabilities for applications that utilize intermediaries (e.g., SOCKS servers or HTTP proxy caches). Narwhal Client Agents (NCAs) normally are deployed on every host in an enterprise, but they can act as front- and back-ends to servers. While a Narwhal Client Agent is capable of a limited amount of autonomous decision making, the full capabilities of a Narwhal-based infrastructure are realized through the active management of each NCA. Benefits of such active management include resource reservation, the proactive communication of information regarding poorly performing intermediaries, and automatic configuration of mobile clients. Narwhal Client Agents use TEMP as the mechanism by which they are enabled for remote management.

2 Overview of TEMP

TEMP was designed to meet several goals, including support for:

- Highly scalable enterprise environments
- mixed-protocol environments (e.g., IPV4, IPV6, IPX, SNA, etc.).
- multiple independent entities running on a single host.
- automatic discovery and administrative domains.
- non-scalar data.

- self-describing instrumentation and management interfaces.

In the context of remote management of Narwhal Client Agents, the attributes of scalability and automatic discovery of entities are particularly important. TEMP enables the management of a large, unknown and dynamic population of NCAs.
TEMP is datagram-based, eliminating the scaling problems created by session-oriented protocols (e.g., CMIP, most ad-hoc RPC-based approaches). 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 periodic requests 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 queue of some finite length. An event queue is logically implemented as a circular buffer, so when the queue length exceeds the entity-specific limit, the oldest entry is deleted. The STATUS messages include a timestamp and 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 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.

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 terms of their administrative relationships. Each TEMP entity 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 together 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.

3 Overview of Narwhal

Narwhal Client Agents provide a rich set of capabilities that improve the performance and reliability of intermediary-aware applications. Some of an NCA's capabilities are:

- Avoidance of slow or failed intermediaries
- Bypass of intermediaries when appropriate
- Prioritization of traffic based on type
- Use alternative intermediaries based on type of requested data

3.1 Avoidance of Poorly Performing Intermediaries

The avoidance of intermediaries that are poorly performing or, in the worst case, completely inoperable, is an obvious benefit provided by NCAs. Wasting time trying to use a slow or failed server when faster alternatives exist is clearly non-optimal, so the point will not be belabored; however, the process of selecting an alternative provider is interesting. NCAs are typically provided with a collection of intermediaries that provide equivalent functionality.

3.2 Bypass of Intermediaries

Many companies protect their internal corporate networks by using firewalls. Applications that need to access content outside the corporate network typically must traverse a firewall using some protocol (e.g., SOCKS, HTTP proxy, etc.); however, it is inefficient for such applications to interact with the firewall if they are accessing content or services within the internal corporate network. Unnecessary accesses introduce unproductive load on resource-constrained gateways. Narwhal Client Agents can eliminate such accesses.

3.3 Traffic Prioritization

Users that are connected to the network using slow dialup links are rarely influenced by operational, yet slow intermediaries as their extremely limited connection bandwidth is the dominant performance constraint. Narwhal Client Agents have the ability to prioritize traffic based on its type (as well as other factors). As an example, this permits a file transfer to be given a lower priority than interactive HTTP browsing or telnet sessions. This permits interactive applications to remain responsive while time-insensitive data transfers make use of the communications link when it is otherwise idle.

3.4 Use of Alternate Types of Intermediaries

Narwhal Client Agents have the capability of protocol conversion, that is, utilizing a different protocol than that used by the requesting application. This feature enables very useful capabilities, such as the use of alternate types of intermediaries. This is

illustrated by the following scenario. A large multinational firm has a single web server farm that provides employees with access to corporate news, human resource information, etc. Regardless of an employee's physical location, any access to the corporate web pages results in a hit against the single site and generates a lot of traffic "across the oceans". If an employee's host runs a Narwhal Client Agent, requests for pages that have a high probability of being in an HTTP proxy cache can be sent to a nearby cache. If the incoming request was SOCKS-based, this will require a protocol conversion. The net effect is to take load off the corporate web site and reduce the bandwidth consumed on inter-site communication lines by using the internal caches local to each country. In addition, the load is reduced on the caches themselves since they only receive requests for data that they have a high probability of having had cached.

4 Management of Narwhal Client Agents

As noted previously, Narwhal Client Agents are intended to be deployed on each client machine within an enterprise. Depending on the size of the enterprise, this can result in the deployment of a large number of actively managed applications. The design of TEMP addresses the scaling problems raised by such deployments.

4.1 TEMP Registration Agents

When started, each NCA obtains its configuration data from a locally accessible file. Every configuration parameter that controls an NCA's behavior is accessible, via TEMP, by remote management applications. In addition, a large set of performance-related statistics are also made available via TEMP.

Two of the important configuration parameters for every TEMP entity are the list of subscribers to which STATUS notifications are sent and the frequency at which such notifications are sent. The initial set of STATUS subscribers is configured to be a set of well-known TEMP registration agents. TEMP registration agents are deemed to be registration agents by the sole virtue of binding a well-known transport address. All TEMP agents implement the same intrinsic functionality and can perform the functions of a registration agent. A TEMP registration agent's primary function is to provide mappings between a TEMP agent's UUID (Universally Unique ID) and its corresponding physical transport addresses. In this role, it provides naming resolution services similar to those of a DNS server.

When a TEMP registration agent receives a STATUS notification from another TEMP agent, it updates its tables that maintain mappings between TEMP UUIDs and physical transport addresses. If the agent was not previously known, the registration agent adds to its event queue a new event that indicates the discovery of the new TEMP agent.

In the absence of any management applications, the system will have achieved stasis and remain so until the point at which a TEMP agent is terminated. After a period of time in which no STATUS heartbeats have been received, the TEMP registration agents will prune the old information from their mapping tables and place an event indicating the loss of the agent on their event queues.

4.2 Discovery of Narwhal Client Agents

In general, TEMP management applications do not bind to well-known transport addresses as this restriction is forced only upon TEMP registration agents. Nothing precludes, however, a TEMP management application from also performing the role of a registration agent. A conventional TEMP management application will subscribe to STATUS notifications from a set of TEMP registration agents. This enables the management application to be made aware of the additions of events that indicate the discovery or loss of TEMP agents in the network. Such subscriptions, however, are temporary because they are only valid as long as the management application is in operation. This temporary subscription is achieved by sending a two-element request to the remote entity:

- an AddOrReplaceElement command that adds the management application's UUID to the set of STATUS subscribers
- a CreateOrModify command that creates a command that will be executed at some point in the future. The future command is a RemoveElement command that removes the management application's UUID from set of STATUS subscribers.

While in operation, the management application will reissue the command before the timeout period expires. This has the effect of postponing the execution of the RemoveElement command. Once the management application terminates, the timeout period will expire and the RemoveElement command will finally be executed. This will remove the now defunct management application's UUID from the agent's set of STATUS subscribers. Exploitation of such capabilities allows dynamic subscriptions to be made to agents while enabling automatic recovery in the aftermath of an unexpected termination of the management application. This convention is followed by all TEMP management applications that need to establish dynamic subscriptions to a particular TEMP entity.

Given a sufficiently large population of agents, the maintenance of such dynamic subscriptions would become prohibitively expensive if not for TEMP's support for administrative domains. Exploitation of this support permits a management application to send out a single request packet per heartbeat interval instead of a packet to every agent to which it is subscribed. This feature is elaborated upon later.

When the management application receives a STATUS message from one of the registration agents, it compares the reported ID of the last event on the registration agent's event queue. The management application can detect the existence of new events by comparing this ID against a previously recorded value. Note that a burst of events does not result in the sending of a flurry of STATUS messages. The STATUS messages still appear at their periodic intervals, but the ID of the last event in the queue will have been incremented by more than one. The management application can then send a single request to retrieve each of the events that it has not yet seen. This means that even a burst of events results in three packets being exchanged between a management application and a subscribed agent (i.e., status, retrieve, response).

4.3 Mobility

Once a new agent's existence is discovered, a management application can take appropriate action. In the case of Narwhal Client Agents, dynamic configuration by a management application provides support for mobile clients. Consider the case of a corporate user who travels to a different site, perhaps in a different country. Her default settings, while appropriate when her laptop is connected to its home LAN, are probably inappropriate for the new location and might be unworkable due to filtering. The dynamic configuration of location-appropriate settings allows her to move freely from one site to another without needing to adjust settings.

This scenario is enabled by the deployment of several redundant TEMP registration agents throughout the various geographic domains. The addresses of these registration agents are stored in the default non-volatile configuration data read by each NCA when it begins execution. When an NCA is initially brought up, it will send out STATUS messages to each of these registration servers. A TEMP management application subscribed to each of the registration agents processes the events associated with discovery of new agents and looks for new Narwhal Client Agents.

When a new Narwhal Client Agent is discovered, its geographic location can be inferred based on its associated physical transport addresses. Configuration data appropriate for that physical location is then downloaded. One of the items that will be altered as part of the download of new configuration data is the list of STATUS subscribers. The list will be reduced so that STATUS messages are sent to only one registration agent. Note that when the NCA is restarted, the complete list of registration agents will be restored from the non-volatile data and the process will repeat itself.

4.4 Proactive Reconfiguration

Narwhal Client Agents are permitted to autonomously decide to avoid the use of an intermediary when a performance-related failure is detected. When an NCA makes a local determination to avoid an intermediary, it also makes this information available to the Narwhal management applications by adding an appropriate event to the NCA's event queue. The existence of this event will be noted by a Narwhal management application at the time of the next successfully received STATUS message. After retrieving the event, the management application can test to see if the detected outage still exists. If the performance related problem is still in progress, the management application can instruct other Narwhal Client Agents to avoid the use of the indicated intermediary. In a similar fashion, a scheduled outage of an intermediary (e.g., for maintenance) can be invisible to end-users if the NCAs are told to avoid using the intermediary ahead of time.

Such reconfiguration messages could be sent as distinct messages from the management application to each individual Narwhal Client Agent, but this poses obvious scaling problems with sufficiently large populations of clients. Instead, TEMP's support for administrative domains can be used to reduce this overhead to a single packet sent to each administrative domain.

4.5 Administrative Domains

In TEMP, an administrative domain represents a collection of TEMP entities. TEMP packets can be addressed to either individual agents or an administrative domain and the resulting behavior is identical in concept to that of unicast and multicast packets, respectively. TEMP entities can be members of multiple administrative domains and, by convention, TEMP agent developers allocate an administrative domain for each distinct type of agent. This convention allows a management application to send a single packet that will be received by all entities of a particular type. Exploitation of TEMP's administrative domain support can reduce the number of packets sent by a management station, but the agent population named by a given administrative domain may be too large. In such situations, TEMP's If-command directive can be used to limit the actual processing of a packet's contents to a subset of the named population, based on criteria evaluated by the remote entity. This enables the exploitation of the inherent efficiencies of multicast transmission even if the message is not intended for the entire set of addressees.

4.6 Resource Reservation

Resource reservation can help ensure that a demonstration for a customer goes smoothly or an email gateway is able to contribute free cycles while being protected from being overloaded. Because the Narwhal management system is able to remotely control individual Narwhal Client Agents, resources can be reserved on an over-burdened intermediary by reducing the number of NCAs that are permitted to use the intermediary in question. The selection of which client hosts remain able to use the intermediary is a non-trivial task and will not be discussed in detail here. It involves making predictions of a client's resource utilization based on historical data. Some of the statistics maintained on a per-intermediary basis by each Narwhal Client Agent are:

Measurement	*Related Condition*
connection establishment time	Used to indicate how fast a new connection was accepted by the kernel of the host supporting the intermediary. Dominated by packet transmission time.
bytes transmitted	Indicates how much traffic was sent to the intermediary by an NCA
connections established	Indicates how many connections have been made to the intermediary by an NCA
connection duration	Used to indicate how much time was taken to transmit data

4.7 Human-Centric Interfaces

Narwhal Client Agents use a subset of TEMP's support for self-describing instrumentation to implement human-centric interfaces. Using a TEMP<->HTTP gateway, a conventional World Wide Web browser is able to access all of the

information provided by a TEMP agent. Usually this facility is exploited to enable the remote TEMP agent to provide entity-specific operator interfaces. In the case of Narwhal Client Agents, provision has been made to provide formatted tables of performance statistics or the ability to alter operational parameters, such as the list of available intermediaries and their individual rankings. In general, TEMP's support for self-describing instrumentation eliminates the need to develop entity-specific operator interfaces that will run under a particular vendor's management platform.

5 Conclusions

The Enterprise Management Protocol is a new protocol and affiliated software infrastructure, which was designed to support large numbers of manageable entities residing within a multi-protocol environment. Narwhal Client Agents are intended to be deployed on every host within an enterprise and need to be actively managed to realize their full benefit. These benefits include resource reservation, the proactive communication of information regarding poorly performing intermediaries, and automatic configuration of mobile clients. Such active management poses a scaling problem traditionally avoided by current management protocols and platforms. Narwhal Client Agents use TEMP as the mechanism by which they are enabled for such scalable active management. They also exploit TEMP's support for self-describing instrumentation to provide user interfaces using conventional World Wide Web browsers, thereby permitting their remote management without requiring a specialized management system.

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] http://www.apache.org

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

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