Einbindung applikations-spezifischer Agenten in das SNMP Management

J. Schönwälder, H. Langendörfer
TU Braunschweig, Institut für Betriebssysteme und Rechnerverbund
Bültenweg 74/75, 38106 Braunschweig

Zusammenfassung

Netzwerkmanagement-Protokolle werden immer häufiger zur Überwachung und Kontrolle von Anwendungsprogrammen eingesetzt. Dabei stellt sich grundsätzlich das Problem, wie die zur Überwachung relevanten Komponenten eines Anwendungsprogramms als Objekte einer Management Information Base (MIB) bereitgestellt werden können und wie diese MIB als Teil eines bereits vorhandenen Management-Agenten exportiert werden kann. In diesem Beitrag stellen wir verschiedene Alternativen und Realisierungsmöglichkeiten basierend auf dem Simple Network Management Protocol (SNMP) vor.

1 Einleitung

Die in den letzten Jahren entwickelten und standardisierten Protokolle zum Netzwerkmanagement haben sich insbesondere zur Überwachung von Netzwerkkomponenten wie z.B. Brücken und Router durchgesetzt. In letzter Zeit besteht ein zunehmendes Interesse, nicht nur Netzwerkkomponenten sondern auch Anwendungsprogramme in das Netzwerkmanagement einzubeziehen. Der Grund hierfür ist zum einen die Tatsache, daß immer mehr Anwendungen von den durch die Vernetzung entstehenden Möglichkeiten Gebrauch machen und zur Klärung von Netzwerkproblemen häufig wichtige Informationen liefern und andererseits die in Netzwerkmanagement-Plattformen integrierten Werkzeuge prinzipiell ohne Änderungen zur Überwachung der korrekten Arbeitsweise einer Applikation benutzt werden können. Es ist lediglich erforderlich, daß eine Applikation Status- und Konfigurationsinformationen über ein Managementprotokoll zugänglich macht.

Zur Integration einer Applikation in das Netzwerkmanagement muß zunächst eine geeignete Management Information Base (MIB) definiert werden, die die für das Management interessanten Objekte der Applikation beschreibt. Ein Beispiel für eine MIB für Anwendungsprogramme ist die im Internet vorgeschlagene MIB zur Beschreibung von relationalen Datenbankmanagementsystemen [1]. Die Erstellung einer MIB-Definition ist ein typisches Entwurfsproblem, das mit gewöhnlichen Entwurfswerkzeugen und speziellen Hilfsprogrammen wie z.B. MIB-Compilern oder syntaxgesteuerten MIB-Editoren gelöst werden kann. Bei der anschließenden Implementierung der MIB-Objekte sind zwei Schnittstellen von besonderer Bedeutung:

1. Der bereits auf dem Zielsystem vorhandene Agent muß so erweitert werden, daß Zugriffe auf die neuen MIB-Objekte möglich werden.

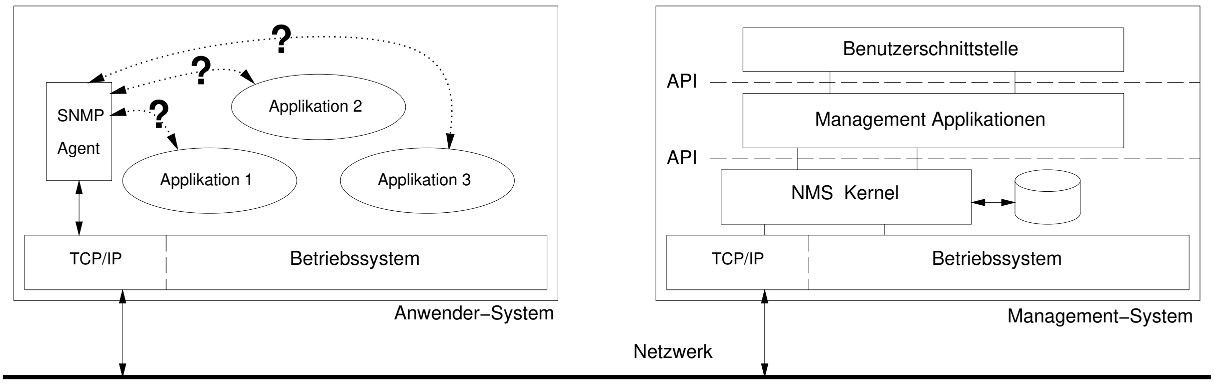

Abbildung 1: Szenario zur Integration von Applikationen in das Netzwerkmanagement.

2. Die Implementation der MIB-Objekte selbst erfordert den Zugriff auf Daten, die typischerweise in der Applikation gehalten werden.

Abbildung 1 stellt das Problem graphisch dar. Auf einem Anwender-System läuft neben den Applikationen ein SNMP-Agent, der die Objekte der Standard-MIB [3] bereitstellt und die dazu notwendigen Informationen durch den direkten Zugriff auf das Betriebssystem erhält. Da ein SNMP-Agent meist eng mit der Implementation der Protokolle im Betriebssystem verzahnt ist, bedeutet dieses Vorgehen kein Problem. Vollkommen anders gestaltet sich die Situation beim Zugriff auf Statusinformationen einer Applikation. Da Applikationen von verschiedenen Herstellern stammen können sind direkte Zugriffe auf interne Daten einer Applikation im allgemeinen nicht möglich und auch nicht unbedingt wünschenswert.

In diesem Beitrag werden verschiedene Möglichkeiten zur Anbindung applikations-spezifischer Agenten an einen Standard-Agenten und an die Applikation vorgestellt. Obwohl die zugrundeliegenden Mechanismen prinzipiell unabhängig von der jeweils benutzten Management-Architektur sind, wird für die folgenden Betrachtungen das weit verbreitete Simple Network Management Protocol (SNMP) [9] der TCP/IP Protokollfamilie zugrundegelegt.

2 Anbindung an den SNMP-Agenten

Zunächst soll die Anbindung an den SNMP-Agenten genauer betrachtet werden. Das zu lösende Problem ist die Bereitstellung neuer, zusätzlicher MIB-Objekte durch einen bereits existierenden SNMP-Agenten. Hier bieten sich folgende Lösungsmöglichkeiten an:

- Multiplexing-Protokolle
- Proxy-Agenten
- Dynamisch ladbare MIB-Erweiterungen

Diese drei Alternativen werden in den folgenden Abschnitten anhand von Beispielen genauer betrachtet.

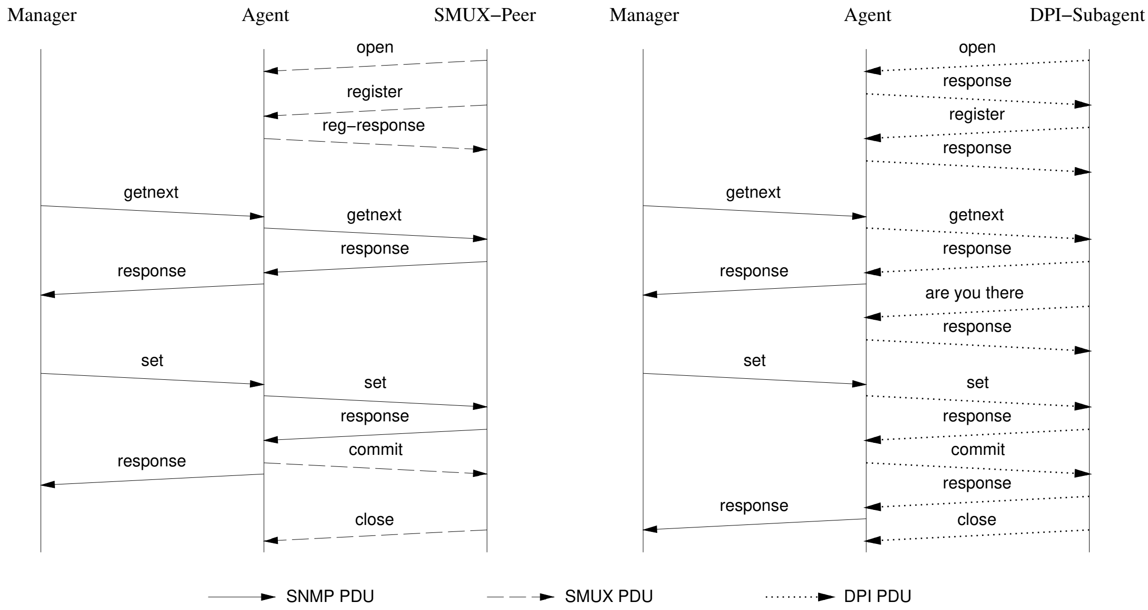

Abbildung 2: Ablauf des SMUX-Protokolls und des DPI-Protokolls.

2.1 Multiplexing-Protokolle

Multiplexing-Protokolle erlauben es, SNMP-Nachrichten von einem SNMP-Agenten auf verschiedene Prozesse zu multiplexen, die jeweils einen Teil der gesamten MIB implementieren. Typischerweise melden sich die Prozesse beim SNMP-Agenten an und registrieren danach die Objekte, die sie exportieren wollen. Zugriffe auf die exportierten Objekte werden vom SNMP-Agenten an die zuständigen Prozesse weitergeleitet. Da sich eine SNMP-Operation über mehrere exportierte MIBs erstrecken kann, muß die SNMP-Nachricht unter Umständen zerlegt und in mehreren Teilzugriffen bearbeitet werden. Der Vorteil von Multiplexing-Protokollen ist deren Eigenschaft, den SNMP-Agenten einem Manager gegenüber transparent erscheinen zu lassen.

2.1.1 SMUX

Das SNMP Multiplexing Protokoll (SMUX) [7] ist eines der ersten Multiplexing-Protokolle gewesen. Eine Implementierung dieses Verfahrens kann allerdings nicht mehr empfohlen werden, da das SMUX Protokoll mittlerweile den Status “historic” hat. Trotzdem ist es interessant, den verfolgten Ansatz kurz zu skizzieren um ihn mit anderen Verfahren vergleichen zu können. Ein Beispiel-Protokollablauf ist in der linken Hälfte von Abbildung 2 dargestellt.

Das SMUX-Protokoll wird zwischen dem Agenten und einem sogenannten SMUX-Peer über ein verbindungsorientiertes Protokoll (TCP) abgewickelt. Das Protokoll selbst besteht aus den Protokolldateneinheiten (PDUs) des SNMP-Protokolls und fünf zusätzlichen SMUX PDUs. Ein SMUX-Peer meldet sich mit einer `open` PDU beim SNMP-Agenten an und registiert anschließend die MIB-Objekte, die er über den Agenten exportieren möchte. Der Registrationswunsch wird vom SNMP-Agenten durch eine `reg-response` PDU bestätigt.

Beim Eintreffen einer SNMP-Nachricht, mit der auf exportierte Objekte zugegriffen werden soll, schickt der Agent eine entsprechende SNMP-Nachricht an die betroffenen SMUX-Peers, nachdem die Liste der Objekte für den jeweiligen Peer angepaßt worden ist. Der SMUX-Peer bearbeitet die Anfrage und schickt eine SNMP-Antwort zurück. Der SNMP-Agent fügt die einzelnen Antworten der SMUX-Peers zu einer SNMP-Antwort zusammen und ergänzt sie gegebenenfalls noch um eigene MIB-Objekte.

Zur Behandlung von schreibenden Zugriffen ist eine weitere SMUX-Nachricht notwendig, um die "as if simultaneous" Semantik von SNMP-Operationen zu gewährleisten. Zunächst wird die Schreiboperation an die SMUX-Peers weitergeleitet. Nachdem alle SMUX-Peers eine Antwort geschickt haben, wird die Schreiboperation bestätigt oder es wird eine Zurücknahme der Änderung veranlaßt. Hierzu dient die `commit` PDU, die sowohl eine positive Bestätigung als auch die Auforderung zur Rücknahme einer Schreiboperation beinhalten kann.

Der Abbruch einer Verbindung zwischen einem SMUX-Peer und einem Agenten kann durch eine `close` PDU sowohl vom Agenten als auch vom SMUX-Peer initiiert werden.

2.1.2 DPI

Das SNMP Distributed Protocol Interface (DPI) [11] wurde von IBM entwickelt und ist in der ersten Version zeitgleich mit dem SMUX-Protokoll veröffentlicht worden. Im Gegensatz zum SMUX-Protokoll basiert DPI nicht auf einem verbindungsorientierten Protokoll, weshalb etwas mehr Aufwand zur Bestätigung der einzelnen DPI-Nachrichten notwendig ist.

Die rechte Hälfte von Abbildung 2 zeigt den prinzipiellen Ablauf des DPI-Protokolls. Analog zum SMUX-Protokoll meldet sich ein DPI-Subagent mit einer `open` PDU an. Anschließend werden die exportierten MIB-Objekte beim Agenten registriert. Im Gegensatz zum SMUX-Protokoll werden alle DPI-Nachrichten in einem eigenen Format übertragen. Dadurch soll eine Vereinfachung erreicht werden, da zur Implementation eines DPI-Subagents keine Hilfsroutinen zum Ver- und Entpacken von ASN.1/BER Datenstrukturen [10] notwendig sind. Beispielsweise werden Object Identifier in einer lesbaren Punktschreibweise als einfache Zeichenketten übertragen, wobei der benutzte Zeichensatz vorher mit der `open` PDU vereinbart worden ist[1].

Schreiboperationen werden durch eine `commit` PDU bestätigt oder mit Hilfe einer `undo` PDU zurückgesetzt. Eine `commit` Nachricht wird im Gegensatz zum SMUX-Protokoll ebenfalls vom DPI-Subagent bestätigt. Ist die Bestätigung nicht bei allen beteiligten DPI-Subagents positiv, so folgt der `commit` PDU noch eine `undo` PDU. Die `are-you-there` PDU kann von einem Subagent benutzt werden, um die Erreichbarkeit des Agenten zu prüfen.

2.2 Proxy-Agenten

Das SNMP-Protokoll sieht sogenannte Proxy-Agenten vor, die SNMP-Nachrichten an andere SNMP-Agenten weiterleiten[2]. Abbildung 3 zeigt den prinzipiellen Ablauf einer Proxy-Kommunikation. Im Gegensatz zu den Multiplexing-Protokollen werden hier lediglich SNMP-Nachrichten weitergeleitet. Das hat zur Folge, daß das Verfahren nicht vollständig transparent ist, da die Managementstation entsprechende Kontexte (SNMP Version 2) oder Community-Namen (SNMP Version 1) zur Adressierung der Proxy-Agenten benutzen muß [9]. Es ist daher nicht möglich, mit einer SNMP PDU auf MIB-Objekte in verschiedenen Proxy-Agenten zuzugreifen.

[1] DPI wurde von IBM entwickelt, was die Unterstützung der Zeichensätze ASCII und EBCDIC erklärt.

[2] In den neuesten SNMP Spezifikationen wurden die Proxies für andere Managementprotokolle (foreign proxies) komplett gestrichen und es sind nur noch SNMP-Proxies (native proxies) im Standard definiert.

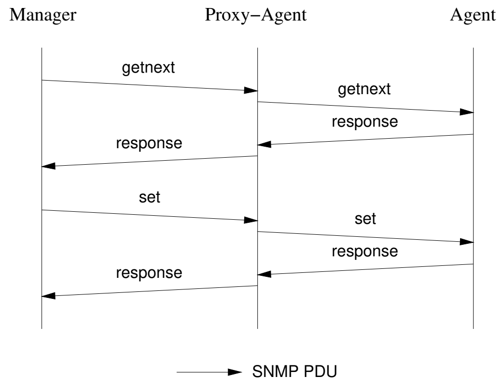

Abbildung 3: Ablauf einer SNMP Proxy-Kommunikation.

Obwohl der Ablauf innerhalb der Proxy-Agenten im Vergleich zu einem Multiplexing-Protokoll wesentlich einfacher ist, muß die empfangene SNMP PDU größtenteils dekodiert werden, da die zur Identifikation benutzte Request-ID abgebildet werden muß [2].

2.3 Dynamisch ladbare MIB-Erweiterungen

Ein ganz anderer Ansatz zur Erweiterung eines Agenten ist das dynamische Binden von Objekt-Modulen, die zusätzliche MIB-Objekte implementieren. Die Spezifikation des Open SNMP Agent [5] beruht auf dieser Idee. Abbildung 4 zeigt an einem Beispiel, wie `getnext` und `set` Operationen auf entsprechende Funktionsaufrufe in sogenannten MIB-Server Modulen abgebildet werden.

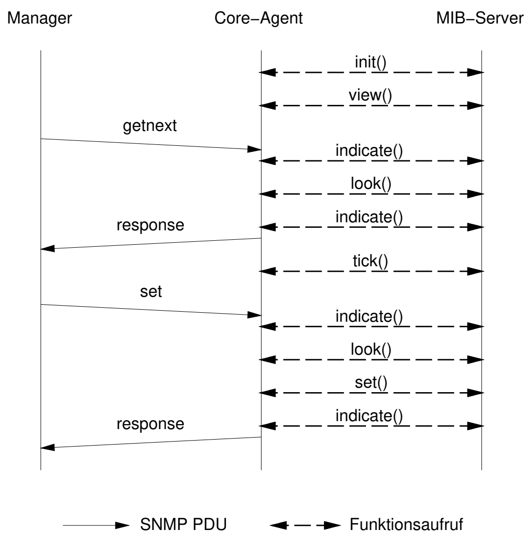

Abbildung 4: Ablauf im Open Agent.

Die Funktionsaufrufe, die den Nachrichten der Multiplexing-Protokolle entsprechen, werden beim Laden eines MIB-Servers registriert. Die `init()` Funktion dient zur Initialisierung des MIB-Servers und die `view()` Funktion übergibt statische View-Informationen an den Core-Agent. Der Anfang und das Ende der Bearbeitung einer SNMP-Nachricht wird mit je einem Aufruf der `indicate()` Funktion bei jedem betroffenen MIB-Server angezeigt. Dies erlaubt die

Implementation einfacher Caches, was bei den oben genannten Multiplexing-Protokollen nicht vorgesehen ist. Die `look()` Funktion realisiert die SNMP-Zugriffe (lesend und schreibend) und die `set()` Funktion entspricht den `commit` Nachrichten der Multiplexing-Protokolle. Die `tick()` Funktion ist vorhanden, um den MIB-Servern gelegentlich die Kontrolle zu geben. Dies ist z.B. notwendig, um periodisch MIB-Objekte zu aktualisieren.

Ein Problem dieses Ansatzes ist die Regelung von Zugriffsrechten. Ein SNMP-Agent benötigt gewöhnlich spezielle Rechte zum Zugriff auf Kernel-Statistiken des Betriebssystems. Die MIB-Server können beispielsweise zum Zugriff auf Datenbanken weitere besondere Rechte benötigen. Das Laden von Objektmodulen zur Laufzeit kann durchaus Anlaß zur Sorge geben, da unerlaubte Zugriffe auf Daten anderer Module schwer zu unterbinden sind. Dieser Ansatz wird derzeit für einen erweiterbaren Windows SNMP-Agenten ausgearbeitet, wobei auch die Abbildung der Funktionsaufrufe auf Messages vorgesehen ist [6].

3 Anbindung an die Applikation

Nachdem im letzten Abschnitt Verfahren zur Anbindung an einen Agenten erläutert wurden, werden in diesem Abschnitt Möglichkeiten zur Anbindung eines Agenten an die Applikation vorgestellt. Die Arbeitsweisen der Agenten lassen sich in zwei grundlegend verschiedene Verfahren einteilen [7]:

- Beim request/response-Verfahren werden MIB-Objekte erst beim Eintreffen von SNMP-Nachrichten durch Daten der Applikation aktualisiert. Erfolgen keine SNMP-Zugriffe, so findet auch keine Interaktion mit der Applikation statt. Treffen hingegen sehr schnell viele SNMP-Nachrichten ein, so werden MIB-Objekte gegebenenfalls häufiger als unbedingt notwendig aktualisiert.
- Das cache-ahead-Verfahren beruht auf der Idee, die MIB-Objekte unabhängig von den tatsächlichen Zugriffen auf die Objekte entsprechend einer vorgegebenen Genauigkeit aktuell zu halten. Dies kann entweder durch die Applikation unterstützt werden, indem ein Mechanismus existiert, um relevante Zustandsänderungen zu propagieren, oder es kann ein periodischer Abgleich mit der Applikation erfolgen.

Abhängig von der Änderungshäufigkeit und dem erwarteten Zugriffsmuster auf ein MIB-Objekt ist entweder ein cache-ahead-Verfahren oder ein request/response-Verfahren besser geeignet. In praktischen Implementierungen wird man häufig beide Verfahren gemischt einsetzen. In den folgenden Abschnitten werden verschiedene Möglichkeiten zur Anbindung von Agenten an Applikationen vorgestellt.

3.1 Eingebettete Agenten

Die direkte Einbettung eines Agenten in die Applikation erlaubt den direkten, uneingeschränkten und effizienten Zugriff auf den aktuellen Status der Applikation. Dabei kann ein einfaches request/response-Verfahren benutzt werden, da der Zugriff auf den aktuellen Status nicht besonders aufwendig ist. Allerdings bereitet diese Lösung in vielen Fällen Probleme:

- Die Bearbeitung von eintreffenden SNMP-Nachrichten wird unter Umständen durch andere Aktivitäten der Applikation zu sehr verzögert.
- Der Umfang einer kompletten Implementation des SNMP-Protokolls kann insbesondere in der Version 2 einiges an Ressourcen kosten.
- Der Implementationsaufwand für SNMP Version 2 ist nicht unerheblich und kann zu neuen Fehlern in der Applikation führen.
- Änderungen und Erweiterungen sind nur durch die Anpaßung des Quelltextes und ein Neuübersetzen und Binden der Applikation möglich. Benutzerspezifische Anpassungen sind damit aufgrund von Lizenzbedingungen in vielen Fällen nicht möglich.

Einige der Probleme lassen sich lösen, indem man z.B. eine Thread-Laufzeitumgebung verwendet oder den applikations-spezifischen Agenten über Shared-Memory oder ähnliche Mechanismen auf den Status der Applikation zugreifen läßt. Außerdem kann man auch auf existierende SNMP-Bibliotheken zurückgreifen. Allerdings erfordert ein eingebetteter Agent normalerweise die Unterstützung durch den Hersteller der Applikation.

3.2 Erweiterbare Agenten

Ein anderer Ansatz ist die lose Anbindung des applikations-spezifischen Agenten an die Applikation, wobei die Applikation Management-Informationen an einer definierten Schnittstelle bereitstellt. Der Vorteil ist eine deutliche Trennung der zumeist optionalen Management-Funktionalität von der Applikation und die Möglichkeit zur Nutzung von einem applikations-spezifischen Agenten für eine ganze Reihe von verschiedenen Applikationen durch die Schaffung allgemein akzeptierter Schnittstellen.

3.2.1 HPOV Extensible Agent

Der HP Open View Extensible Agent benutzt einen sehr einfachen Mechanismus, um zusätzliche MIB-Objekte zu realisieren. In die Beschreibung eines Objekts innerhalb der MIB-Definition können Betriebssystem-Kommandos eingebettet werden, die beim Zugriff auf das MIB-Objekt ausgeführt werden. Durch geeignete Kommandofolgen läßt sich der Zugriff auf Status- und Konfigurationsinformationen einer Applikationen nach dem request/response-Verfahren realisieren. Abbildung 5 zeigt ein einfaches Beispiel, in dem zwei MIB Variablen implementiert werden, die Auskunft über die Mail-Warteschlange geben[3].

Das Verfahren ist relativ einfach und oft ziemlich ineffizient, da jeder Zugriff unter Umständen recht aufwendige Befehlssequenzen auslöst. Es ist auch nicht direkt möglich, Tabellen entsprechend der Konventionen zur Erzeugung von Tabellenzeilen korrekt zu verwalten oder cache-ahead Strategien zu implementieren. Außerdem muß bei der Auswahl der Befehlssequenzen jeweils bedacht werden, unter welchen Rechten die Befehle ausgeführt werden und welche Sicherheitslücken dadurch eventuell entstehen können.

[3] Der Typ `DisplayString` ist auf 255 Zeichen begrenzt. Daher macht die Variable `fmailListMsgs` nur begrenzt Sinn.

```
fmailNumMsgs OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
    "Message count on the mail queue.
    READ-COMMAND: /usr/bin/mailq
      | fgrep -c -v Mail"
  ::= { fmail 1 }
```

```
fmailListMsgs OBJECT-TYPE
  SYNTAX DisplayString
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
    "Messages on the mail queue.
    READ-COMMAND: /usr/bin/mailq
    READ-COMMAND-TIMEOUT: 10"
  ::= { fmail 3 }
```

Abbildung 5: Implementation von MIB-Variablen mit dem HPOV Extensible Agent.

3.2.2 Programmierbare Agenten (Scotty)

Eine weitere Alternative sind Agenten, die über eine eingebettete Programmiersprache verfügen und zur Laufzeit Programme laden können, mit denen ein MIB-Objekt an eine Applikation gebunden werden kann. Ein Beispiel für diesen Ansatz ist der an der TU Braunschweig entwickelte SNMP-Agent Scotty [8], der auf der Tool Command Language (Tcl) [4] basiert.

Der Scotty SNMP-Agent bindet MIB-Objekte an globale Tcl-Variablen. Eintreffende SNMP-Nachrichten werden in der Event-Schleife des Agenten automatisch durch das Lesen und Setzen der betroffenen Tcl Variablen bearbeitet. Durch die Definition von sogenannten Bindings können bei der Auswertung einer SNMP-Nachricht beliebige Tcl-Kommandos ausgeführt werden. Zusätzlich zu den normalen SNMP-Zugriffen (`get`, `set`)[4] sind Bindings für `create`, `commit`, `rollback`, `begin` und `end` Ereignisse möglich. Ein `create` Ereignis entspricht einem SNMP Schreibzugriff auf eine nicht existierende aber erzeugbare Variable. Die `commit` und `rollback` Bindings sind zur Erhaltung der “as if simultaneous” Semantik notwendig. Die `begin` und `end` Ereignisse zeigen den Anfang und das Ende der Bearbeitung einer SNMP-PDU an und können dazu dienen, MIB-Objekte entsprechend zu aktualisieren.

Abbildung 6 zeigt die Implementation einer MIB Tabelle, die für jede Mail in der Warteschlange die Spalten `mailqID`, `mailqSize` und `mailqSender` mit einem Wert belegt. Die Informationen werden mit Hilfe der Prozedur `ReadMailQ` erfaßt. `UpdateMailQTable` trägt die Daten durch das Erzeugen entsprechender Instanzen in die Tabelle ein. Eine zusätzliche Skalare Variable `mailqLength` enthält die Länge der Warteschlange. Die Aktualisierung der Tabelle erfolgt durch ein `begin` Binding, das bei jeder eintreffenden SNMP Nachricht prüft, ob die letzte Aktualisierung mehr als 30 Sekunden zurück liegt.

Durch die Verwendung von einer eingebetteten Programmiersprache können beliebige Verfahren (request/response oder cache-ahead) zum Zugriff auf Applikationen implementiert werden. Da alle Daten als einfache Zeichenketten dargestellt werden, bleiben dem Programmierer lästige Details erspart und er kann sich auf die Interaktion mit der Applikation konzentrieren. Scotty bietet auch Schnittstellen zu TCP/IP Diensten, so daß Daten auch über applikations-spezifische Protokolle gesammelt werden können. Da mehrere Skripte gleichzeitig und dynamisch geladen werden können, kann man den Zugriff auf mehrere verschiedene Applikationen mit einem Scotty Prozeß realisieren.

[4]Da die Instanzen intern verwaltet werden, werden `getnext` oder `getbulk` Zugriffe automatisch zu `get` Zugriffen auf das lexikographisch nächste MIB Objekt.

```
proc ReadMailQ {} {
    set f [open "| mailq" r]
    while {[gets $f line] >= 0} {
        if {[scan $line "%s %d %*s %*s %*d %*s %s" id size sender] == 6} {
            set m(id,$id) $id
            set m(size,$id) $size
            set m(sender,$id) [string trim $sender "<>()"]
        }
    }
    close $f
    return [array get m]
}

proc UpdateMailQTable {s} {
    global mailq
    foreach n [array names mailq] { unset mailq($n) }
    array set m [ReadMailQ]
    foreach n [array names m id,*] {
        set idx [mib scan sysDescr [set id $m($n)]]
        $s instance mailqID.$idx     mailq(id,$id)     $id
        $s instance mailqSize.$idx   mailq(size,$id)   $m(size,$id)
        $s instance mailqSender.$idx mailq(sender,$id) $m(sender,$id)
    }
    $s instance mailqLength.0 mailq(length) [llength [array names m id,*]]
}

proc InitMailQMIB {s} {
    $s bind "" begin {
        if {! [info exists lastUpdate]} { set lastUpdate 0 }
        if {[getclock] - $lastUpdate > 30} {
            UpdateMailQTable %S
            set lastUpdate [getclock]
        }
    }
}
```

Abbildung 6: Implementation einer Mail-MIB-Tabelle mit Scotty.

4 Zusammenfassung

In diesem Beitrag haben wir einige Möglichkeiten zur Einbindung von Applikationen in das SNMP-Management vorgestellt. Die wichtigsten Eigenschaften der hier vorgestellten Verfahren zur Anbindung an einen SNMP-Agenten sind in der Tabelle 1 zusammengefaßt.

Die Anbindung an die Applikation kann entweder direkt durch eingebettete Agenten oder aber durch erweiterbare oder programmierbare Agenten erfolgen. Eingebettete Agenten erfordern im allgemeinen die Unterstützung des Herstellers einer Applikation, während eine lose Kopplung oftmals durch die Benutzung bereits vorhandener Programme/Schnittstellen realisiert werden kann.

Merkmal	SMUX	DPI	SNMP Proxy	Open Agent
Transparenz	vollständig	vollständig	unvollständig	vollständig
Schreibzugriffe	2-Phase-Commit	3-Phase-Commit	trivial	2-Phase-Commit
Kommunikation	SNMP & SMUX PDUs	DPI PDUs	SNMP PDUs	Funktionsaufrufe
Datendarstellung	ASN.1/BER	applikationsnah	ASN.1/BER	applikationsnah
Paketgrenzen	nicht angezeigt	nicht angezeigt	trivial	angezeigt
Status	historic	proprietary	standard	proprietary

Tabelle 1: Möglichkeiten zur Anbindung an SNMP-Agenten.

Literatur

[1] D. Brower. Relational Database Management System (RDBMS) Management Information Base (MIB) using SMIv2. RFC 1697, The ASK Group, INGRES DBMS Development, Oracle Corporation, Informix Software, Inc., August 1994.

[2] J.D. Case, J. Galvin, K. McCloghrie, M.T. Rose und S. Waldbusser. Administrative Infrastructure for Version 2 of the Simple Network Management Protocol (SNMPv2). Internet Draft 01, SNMP Research Inc, Trusted Information Systems, Cisco Systems Inc, Dover Beach Consulting Inc, Carnegie Mellon University, März 1995.

[3] K. McCloghrie und M. Rose. Management Information Base for Network Management of TCP/IP-based Internets: MIB-II. RFC 1213, Hughes LAN Systems, Performance Systems International, März 1991.

[4] J.K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, April 1994.

[5] A.Y. Romanov, W.H. White und P. Wilson. The Open SNMP Agent. The Interface for Programming the Extensible SNMP V1/V2 Agent. Release Notes, Paul Freeman Associates, Digital Equipment Corporation, 1994.

[6] A.Y. Romanov, W.H. White und P. Wilson. Windows SNMP Agent. The Open Interface for Programming the Extensible SNMP V1 and V2 Agent under Under Microsoft Windows. Version 0.3, Paul Freeman Associates, Digital Equipment Corporation, Dezember 1994.

[7] M. Rose. SNMP MUX Protocol and MIB. RFC 1227, Performance Systems International, Mai 1991.

[8] J. Schönwälder und H. Langendörfer. Tcl Extensions for Network Management Applications. In *Proc. 3rd Tcl/Tk Workshop*, Toronto (Canada), Juli 1995.

[9] W. Stallings. *SNMP, SNMPv2 and CMIP: The Practical Guide to Network Management Standards*. Addison-Wesley, 1993.

[10] D. Steedman. *Abstract Syntax Notation One (ASN.1): The Tutorial and Reference*. Technology Appraisals, 1990.

[11] B. Wijnen, G. Carpenter, K. Curran, A. Sehgal und G. Waters. Simple Network Management Protocol Distributed Protocol Interface Version 2.0. RFC 1592, IBM T.J. Watson Research Center, Bell Northern Research Ltd., März 1994.