Entwurf und Implementierung eines SNMP-XML-Gateways

Jens Müller

26. März 2003

Technische Universität Braunschweig
Institut für Betriebssysteme und Rechnerverbund

Studienarbeit

Entwurf und Implementierung eines SNMP-XML-Gateways

von
cand. inform. Jens Müller

Aufgabenstellung und Betreuung:
Prof. Dr. S. Fischer und Dipl. Inform. F. Strauß

Braunschweig, den 26. März 2003

Erklärung

Ich versichere, die vorliegende Arbeit selbständig und nur unter Benutzung der angegebenen Hilfsmittel angefertigt zu haben.

Braunschweig, den 26. März 2003

Kurzfassung

Existierende Technologien zum Netzwerkmanagement stellen sich im immer höheren Maße als ungeeignet heraus, mit dem schnellen Wachstum des Internet Schritt zu halten. SNMP als weit verbreitetes Standard-Framework für das Management vernetzter Geräte erfordert eingearbeitete Spezialisten, um für das Management großer Netzwerke eingesetzt zu werden. Aus diesem Grund fokussiert sich die momentane Forschung auf Verbesserung oder Ersatz von SNMP durch weit verbreitete und gut verstandene Technologien. Diese Ansätze zielen darauf, bestehende Funktionalität zu erweitern und Entwicklungskosten zu verringern. Der Einsatz von XML-Technologien als einer dieser Ansätze erscheint als ein vielversprechendes Konzept, das momentan verwirklichte Netzwerkmanagement zu verbessern. Diese Arbeit gibt eine kurze Motivation für die Implementierung eines XML-basierten Netzwerkmanagement und beschreibt mögliche Konzepte eines SNMP-XML-Gateway als Verbindung dieser Technologien. Eine Implementierung dieses Gateways als HTTP-Servlet, eines der Konzepte eines solchen Dienstes realisierend, wird vorgestellt und bewertet.

Abstract

Existing network management technologies more and more turn out to be uncapable of keeping up with the fast growth of the Internet. The wide spread SNMP as the current standard framework for management of network attached devices requires qualified experts to be applied for management of large networks. Because of that actual research focuses on improving or replacing SNMP with wide spread and well understood technologies. These approaches aim at extending functionality and reducing development costs. Using XML-technologies as one of these approaches appears to be a promising concept to improve currently implemented network management. This work gives a brief motivation for implementing an XML-based network management and describes possible concepts of an SNMP-XML-Gateway as the connection between these technologies. An implementation of this gateway as an HTTP-servlet realizing one of the concepts for such a service is presented and reviewed.

Aufgabenstellung

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Das Simple Network Management Protocol (SNMP) ist das heutzutage gebräuchliche Protokoll, um TCP/IP-Netzwerke zu administrieren. Das SNMP-Framework wurde 1988 von der Internet Engineering Task Force (IETF) spezifiziert. SNMP ermöglicht ausgehend von einer zentralen Management-Anwendung die Administrierung einer großen Zahl an das Netzwerk angeschlossener Geräte, die mit SNMP-Agenten ausgestattet sind. Neben dem Kommunikationsprotokoll spezifiziert das Framework die zugrundeliegende Informationsbasis der in SNMP-Agenten zugreifbaren Objekte. Diese für das Management relevanten Objekten werden hierzu in Management Information Base-Modulen (MIB-Modulen) definiert. Die MIB-Module sind in einer datenorientierten Beschreibungssprache verfasst, der Structure of Management Information (SMI). Inzwischen existieren über 160 von der IETF standardisierte und veröffentlichte MIB-Module. Mit der Kenntnis der von einem bestimmten SNMP-Agenten implementierten Module kann eine Management-Anwendung auf die implementierten Objekte des Agenten zugreifen. Die hierfür notwendige Adressierung der Objekte wird im SNMP-Framework durch Object Identifier (OID) bereitgestellt, die eine Objektadressierung in einem globalen, hierarchisch aufgebauten Namensbaum ermöglichen. SNMP bietet wenige einfache Operationen zum Lesen und Setzen von Instanzwerten der Objekte.

SNMP besitzt aus dieser Spezifikation heraus einige Schwachstellen, die sowohl das Netzwerkmanagement selbst, wie auch die Entwicklung SNMP-basierter Anwendungen erschweren. SNMP ist leider ein Oxymoron, denn die zum Framework gehörenden Spezifikationen sind komplex und bedürfen zum Verständnis einige Einarbeitungszeit. Die verwendeten Konzepte unterscheiden sich deutlich zu den neueren, heutzutage in der Lehre anzutreffenden Techniken wie Objektorientiertheit, webbasierten Anwendungen und Datenrepräsentation via XML. Aus diesem Grund sind eingearbeitete Spezialisten erforderlich, um Netzwerkmanagement über SNMP zu betreiben und SNMP-basierte Anwendungen zu entwickeln. SNMP-Operationen ermöglichen es zudem nicht, einen Objektzustand bestehend aus allen Attributswerten in einer atomaren Anfrage zu verändern oder auszulesen, die Granularität der Operationen ist beschränkt auf einzelne Instanzwerte. Eine weitere Schwachstelle stellt die Informationsbasis in Form der MIB-Module dar: Die Moduldefinitionen sind nicht an zentraler Stelle im Internet zugänglich, ein Agent gibt wenn überhaupt nur in nicht generisch handhabbarer Weise Auskunft über die von ihm implementierten Module. SMI als Beschreibungssprache der MIB-Module unterstützt zudem keine Konzepte wie strukturierte Datentypen oder klassenbezogene Methoden.

All diese Schwachstellen legen den Übergang zu einer anderen Technologie nahe. Ein Ansatz unter Verwendung der Extensible Markup Language (XML) für das Netzwerkmanagement erscheint vielversprechend. Durch ihre inzwischen breite Verwendung stellt XML keine Nischentechnologie dar. Es wurden in den letzten Jahren mehrere Konzepte der Realisierung eines XML-basierenden Netzwerkmanagement vorgestellt ([1], [2]).

Diese Arbeit ist wie folgt gegliedert: Kapitel 2 *(SNMP vs XML)* erläutert die Bestandteile des SNMP-Framework und diskutiert mögliche Ansätze zur Spezifikation eines XML-basiertem Netzwerkmanagement. In Kapitel 3 *(Gateway-Entwurf)* werden Anforderungen an das SNMP-XML-Gateway vorgestellt. Anschließend werden denkbare Spezifikationen des Gateways erläutert. Die Auswahl einer Spezifikation als Grundlage der Implementierung wird in Kapitel 4 *(Spezifikation des Gateways)* vorgenommen. Dabei wird die Syntax der vom Gateway zu verarbeitenden Anfragen sowie seiner Antworten festgelegt. Kapitel 5 *(Implementierung des Gateways)* gibt einen groben Überblick über die Arbeitsweise des implementierten Gateways anhand geeigneter Darstellungen in UML. Eine genaue Dokumentation der Implementierung wird nicht innerhalb dieses Dokumentes gegeben, sondern ist beiliegend des veröffentlichten Quelltextes der Implementierung erhältlich. Ein Test der Implementierung erfolgt in Kapitel 6 *(Evaluierung)*. Das Kapitel 7 *(Bewertung und Ausblick)* diskutiert schließlich, inwiefern die Testergebnisse den Erwartungen gemäß der Spezifikation entsprechen und zeigt mögliche Verbesserungen auf.

Kapitel 2

SNMP vs XML

2.1 Managing mittels SNMP

2.1.1 Historie

Die Anzahl der über TCP/IP kommunizierenden, an das Internet angeschlossenen Hosts hat sich während der späten Achtziger und frühen Neunziger Jahre explosionsartig vergrößert. Während davor Netzwerkmanagement durch Protokoll-Experten mit grundlegenden Werkzeugen wie *ICMP (Internet Control Message Protocol)* und *PING (Packet Internet Groper)* möglich war, stellten sich diese Werkzeuge im schnell wachsenden Internet als nicht effizient genug heraus, jedes Gerät in einem Netzwerk zu beobachten und zu administrieren.

Aus dieser Erkenntnis heraus wurde 1987 das *Simple Gateway Monitoring Protocol (SGMP)* entwickelt. SGMP ermöglichte auf einfache Weise das Management von Gateways, war damit jedoch kein allgemein benutzbares Werkzeug zum Netzwerkmanagement. Das *Simple Network Management Protocol (SNMP)* setzte sich kurz darauf gegen Konkurrenzansätze als Werkzeug für allgemeines Netzwerkmanagement in TCP/IP Netzen durch. Von der *Open Systems Interconnection (OSI)* wurde SNMP zunächst als Interimslösung auf dem Weg zu einem komplett neu entwickelten Protokoll spezifiziert. Diverse Hersteller implementierten SNMP jedoch schnell in viele Produkte, und verschafften dem Protokoll somit eine breite Basis. SNMP ist durch seine schnelle Etablierung zum de facto-Standard für Netzwerkmanagement geworden, inzwischen hat es mit den Versionen SNMPv2 und SNMPv3 zwei größere Revisionen erfahren, wobei jedoch die Mehrzahl der Geräte noch SNMPv1 implementieren [3].

2.1.2 Agenten und Manager

Durch SNMP administrierbare Geräte werden als *managed nodes* bezeichnet. Jeder dieser an ein TCP/IP-Netz angeschlossenen Knoten enthält einen *SNMP-Agenten*. Diese Software ermöglicht die Administrierung ihres Knotens, indem sie auf entsprechende Anfragen reagiert. Die Anfragen werden an die Agenten von *Management-Anwendungen*, häufig auch einfach Manager genannt, gerichtet. Ein Manager ist dabei ein Softwareprozess laufend auf einer *Netzwerkmanagementstation (NMS)*. Zusätzlich zu dieser klaren Rollenverteilung kann es auch Anwendungen geben, die sowohl als Agent wie auch als Manager fungieren [4].

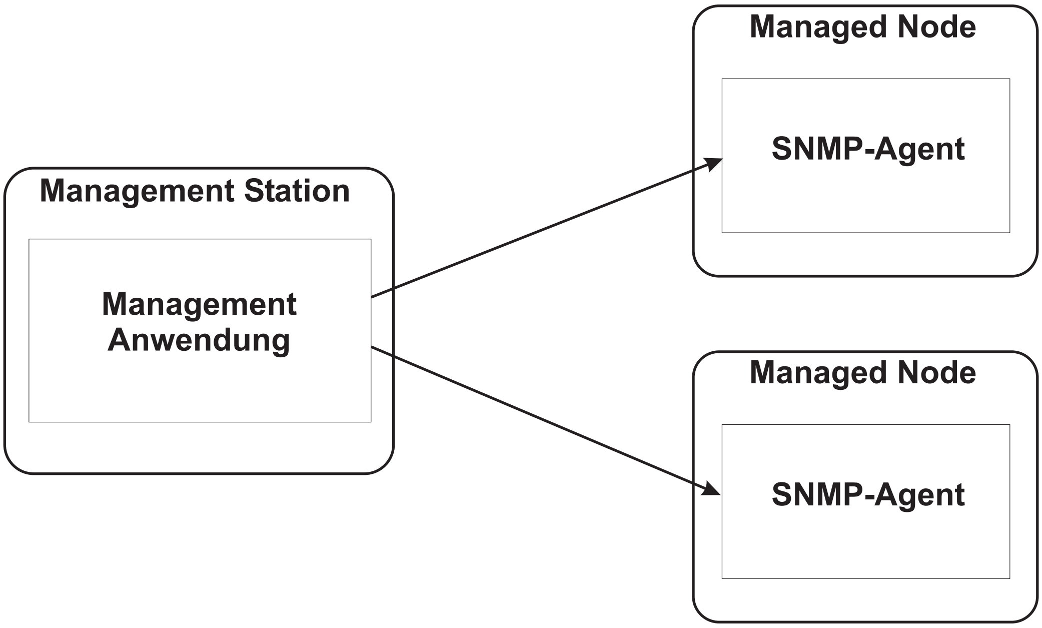

Abbildung 2.1: SNMP Architektur

2.1.3 Datenmodell

Als Kommunikationsgegenstand zwischen Managern und Agenten dienen in *MIB-Modulen* definierte Objekte. Typischerweise aggregiert ein MIB-Modul in einem bestimmtem Kontext zusammengehörende Objekte, so werden beispielsweise in der IF-MIB, dem Modul zur Repräsentation der Netzwerkinterfaces eines Netzwerkknoten, Objekte für einzelne Interface-Attribute wie gesendete und empfangene Pakete aber auch für skalare Werte wie die Anzahl der Interfaces definiert. Die Module werden in einer eigenen Sprache zur Datenspezifikation definiert, der *Structure of Management Information (SMI)*. Dabei gibt es zwei Möglichkeiten zur Angabe von managementrelevanten Objekten. Eine Objektgruppe kann als Gruppe von singulären Unterobjekten definiert sein, die alle nur aus einem einzelnen skalaren Attribut bestehen, zusätzlich nur versehen mit Metaangaben. Um Objektinstanzen in beliebiger Anzahl zu ermöglichen, können Tabellen von Objekten definiert werden. Dabei besteht eine Zeile der Tabelle aus mehreren Spaltenobjekten.

Als Beispiel wird an dieser Stelle ein Ausschnitt aus dem Modul `TCP-MIB` [5] gegeben. Dieses Modul beschreibt eine Objektgruppe `tcp`, die Objekte zum Management von TCP-Entitäten definiert.

```
TCP-MIB DEFINITIONS ::= BEGIN
[...]
tcp OBJECT IDENTIFIER ::= { mib-2 6 }

tcpRtoAlgorithm OBJECT-TYPE
    SYNTAX      INTEGER {
                    other(1),    -- none of the following
                    constant(2), -- a constant rto
                    rsre(3),     -- MIL-STD-1778, Appendix B
                    vanj(4)      -- Van Jacobson's Algorithm [5]
                }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "The algorithm used to determine the timeout value used for
         retransmitting unacknowledged octets"
    ::= { tcp 1 }

tcpRtoMin OBJECT-TYPE
    SYNTAX      Integer32
    UNITS       "milliseconds"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "The minimum value permitted by a TCP implementation for the
         retransmission timeout, measured in milliseconds. [...]"
    ::= { tcp 2 }

tcpRtoMax OBJECT-TYPE
    SYNTAX      Integer32
    UNITS       "milliseconds"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "The maximum value permitted by a TCP implementation for the
            retransmission timeout, measured in milliseconds. [...]"
    ::= { tcp 3 }

[...]
```

Um Objekte adressieren zu können, wird die Objektdefinition in einem MIB-Modul mit einem *Object Identifier* versehen. Dieser OID ist dabei ein eindeutiger Bezeichner eines Knotens in einem globalen OID-Baum, durch seine Angabe wird ein Objekt an eindeutiger Position in diesen Baum eingehängt. Im Beispiel des Modules TCP-MIB wird die Objektgruppe `tcp` durch die Zeile `OBJECT IDENTIFIER ::= { mib-2 6 }` als Kindknoten mit dem Index sechs unter den Knoten `mib-2` in den OID-Baum eingefügt. `mib-2` ist an dieser Stelle ein Name für den kompletten Pfad von der Baumwurzel zu dem entsprechenden

Knoten, ausgeschrieben hat er den Wert 1.3.6.1.2.1. Somit ergibt sich für die Gruppe `tcp` der komplette Pfad 1.3.6.1.2.1.6. Die einzelnen Objekte der Gruppe werden gemäß ihrer Definition als Kindelemente des `tcp`-Knotens in den Baum eingefügt, etwa das Objekt `tcpRtoAlgorithm` als erstes Kindelement mit der OID 1.3.6.1.2.1.6.1. `tcpRtoAlgorithm` ist dabei ein Objekt bestehend aus einem `INTEGER`, der die Werte 0 bis 4 annehmen kann. Die Angaben `MAX-ACCESS`, `STATUS` und `DESCRIPTION` sind Metaangaben.

Abbildung 2.2 stellt einen Ausschnitt des OID-Baumes dar. Die Objektgruppe `tcp` ist als sechstes Kindelement des Knotens `mib-2` mit der OID 1.3.6.1.2.1.6 eingefügt.

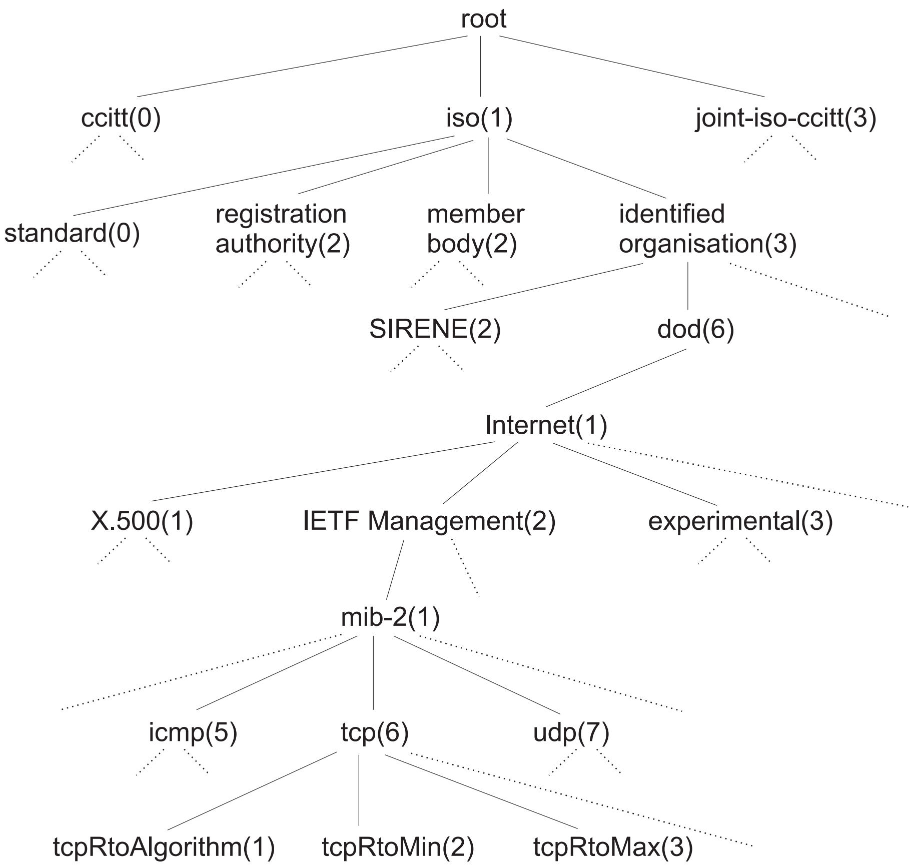

Abbildung 2.2: Auschnitt des OID-Baumes

Diese hierarchische Vorgehensweise beugt Namenskonflikten unter den Objekten vor. Zwar wurde ein Großteil der MIB-Module zentral durch die IETF spezifiziert, jedoch erstellen Hersteller von netzwerkfähigen Geräten ebenfalls Module, um durch Erweiterung bestehender Module spezielle Fähigkeiten ihrer Produkte mittels SNMP administrierbar zu gestalten. Durch Einfügen dieser Objekte in einen speziell für den Hersteller reservierten Teil des OID-Baumes wird eine Prüfung sämtlicher bereits definierter Objekte auf Namenskollisionen durch den Hersteller vermieden.

Die Definition des zugrundeliegenden Datenmodells ist nicht problemlos. Mitunter lassen sich einzelne Objekte nicht ausschließlich einem Kontext zusammengehörender Informationen zuordnen, auch besteht die Notwendigkeit für technisch weiterentwickelte Netzwerkknoten, zusätzliche Attribute zu den Objektgruppen der Standard-MIB spezifizieren zu können. Aus diesen Gründen hat sich ein kompliziertes Geflecht von gegenseitig erweiternden MIB-Modulen entwickelt, welches schwer zu durchschauen ist.

2.1.4 Kommunikation

Die Kommunikation zwischen Agenten und Manager erfolgt gemäß dem eigentlichen SNMP-Protokoll. Dabei stehen dem Manager einfache Operationen zur Verfügung, Instanzwerte der von Agenten implementierten Objekte auszulesen beziehungsweise zu setzen. Der Manager übermittelt in einer Request/Reply-Kommunikationsbeziehung die auszuführende Operation an einen Agenten. Dieser bearbeitet die gestellte Anforderung und sendet anschließend eine entsprechende Antwort an den Manager. Konnte der Agent die Anforderung nicht erfolgreich bearbeiten, enthält die Antwort einen die Fehlersituation beschreibenden Ausnahmewert, bei erfolgreicher Bearbeitung einen operationsspezifischen Rückgabewert. Die Datenübertragung erfolgt in Form einer *Protocol Data Unit (PDU)*, die nach ASN.1/BER kodiert ist [6].

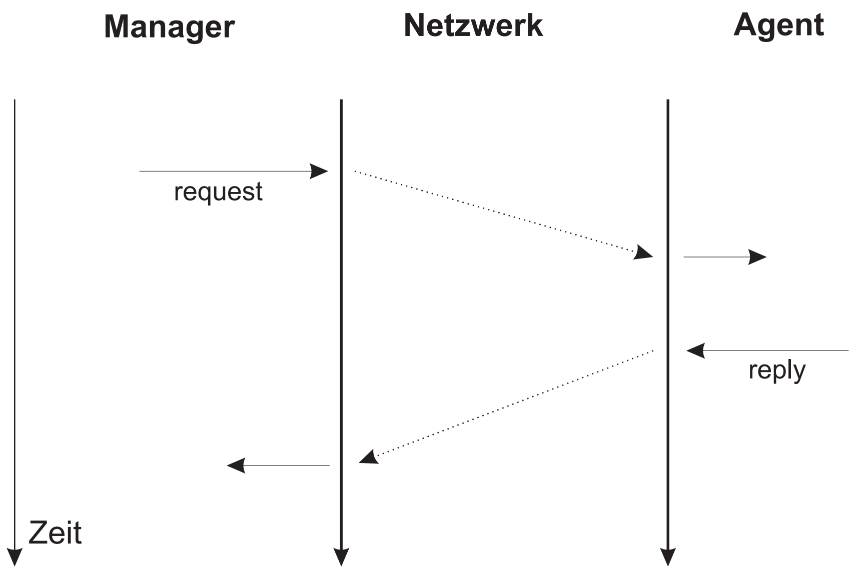

Abbildung 2.3: Request/Reply Kommunikation

SNMPv2 definiert im einzelnen sechs Operationen, die wie folgt kategorisiert werden können:

Leseanforderungen: get, get-next, get-bulk

get(Liste von OBJECT IDENTIFIER) dient zur Abfrage einzelner Objekte. Ist bekannt, welche Objektinstanzen eines Agenten, dargestellt durch ihren zugehörigen OBJECT IDENTIFIER, ermittelt werden sollen, kann der Manager mittels dieser Operation den Agenten zur Übermittlung dieser Werte auffordern.

get-next(Liste von OBJECT IDENTIFIER) bietet dem Manager die Möglichkeit, Instanzwerte eines Agenten iterativ abzufragen. Dies ist insbesondere dann nötig, wenn dem Manager nicht bekannt ist, welche Objekte ein Agent implementiert und wieviele Objektinstanzen innerhalb einer Tabelle vorhanden sind. Der Agent übermittelt die Identifier der direkt auf die Objekte der übergebenen Identifier folgenden Objekte zurück. Der Manager kann anhand dieser Identifier prüfen, ob eine weitere Objektinstanz existiert, und die Abfrage gegebenenfalls iterativ fortsetzen.

get-bulk(Integer, non-repeaters, Integer max-repetitions, Liste von OBJECT IDENTIFIER) stellt eine Erweiterung von get-next dar. Über *max-repetitions* teilt der Manager dem Agenten mit, wieviele nächste Instanzwerte ausgehend von denen in der übergebenen Liste spezifizierten Objekten zurückzugeben sind.

Schreibanforderung: set

Als Operation zur Modifikation von Objektinstanzen eines Agenten dient die *set(Liste von OBJECT IDENTIFIER, Liste von Werten)*-Operation. Der Manager muss die vorhandenen Instanzen innerhalb des Agenten kennen, und sie durch Übergabe des OID adressieren. Der Agent führt auf eine gestellte set-Anforderung die Modifikationen der Objektinstanzen unter Zuhilfenahme eines two-phase-commits durch. Auf diese Weise ist die Atomarität der gesamten Anforderung sichergestellt, eine einzelne fehlerhafte Modifikation führt zum Abbruch der gesamten Operation. Während in SNMPv1 lediglich ein Fehlerstatus `badValue` definiert ist, führt SNMPv2 eine ganze Reihe von Ausnahmenachrichten wie `noAccess`, `wrongType` oder `inconsistentName` ein. Auf diese Weise erhält der Manager eine differenziertere Begründung für einen etwaigen Fehlschlag der set-Operation.

trap-Kommunikation: snmpV2-trap

Traps geben den Agenten die Möglichkeit, Manager über bestimmte Ereignisse zu informieren. Während sämtliche andere Operationen des SNMP-Protokolls Manager-initiiert sind, wird trap-Kommunikation zur Benachrichtigung eines Managers von Agenten angeregt. Dabei übermittelt der initiierende Agent einem Manager den Zeitpunkt des Auftretens des Ereignisses als timestamp, das Ereignis sowie weitere Attribute, die das Ereignis näher beschreiben. Als weiteren Unterschied zu den übrigen Operationen handelt es sich nicht um eine Request/Reply-Kommunikationsbeziehung, der benachrichtigte Manager sendet keine Antwort auf den Erhalt einer trap-Benachrichtigung. Als Beispiel der trap-Kommunikation kann ein Agent einen Manager über Ereignisse wie `linkDown` oder `linkUp` bestimmter Netzwerkinterfaces informieren. Der Agent muss dabei für diese Benachrichtigung konfiguriert sein, also Attributswerte bestimmter dafür zuständiger Objekte besitzen. Anhand dieser Werte ist der Agent informiert, für welche Ereignisse welcher Objektinstanzen ein bestimmter Manager zu benachrichtigen ist. Die Konfiguration eines Agenten für trap-Ziele ist prinzipiell implementierungsspezifisch. Ist der Agent jedoch bezüglich der trap-Ziele über SNMP selbst konfigurierbar, so erfolgt diese Konfiguration anhand von Objekten, die durch die SNMP-TARGET-MIB [7] definiert sind.

Manager zu Manager Kommunikation: inform

Die *inform*-Operation dient zur Benachrichtigung über Ereignisse zwischen Management-Anwendungen. Die Kommunikation erfolgt hierbei ähnlich einer trap-Benachrichtigung, jedoch handelt es sich wieder um eine Request/Reply-Kommunikationsbeziehung, so dass der benachrichtige Manager eine Antwort auf den Erhalt einer inform-Nachricht sendet. Auch hier ist die Konfiguration der Management-Anwendung implementierungsspezifisch.

2.1.5 Sicherheit

SNMPv2 adaptiert im wesentlichen das schon in SNMPv1 benutzte Community-Konzept zur Zugriffsautorisation. Objektinstanzen innerhalb eines Agenten sind Gruppen, den *Communities*, zugeordnet. Jeder Anforderung eines Managers ist ein Community-String zugefügt, der Agent erlaubt vollen Zugriff auf die durch die übergebenen OIDs spezifizierten Objektinstanzen, falls der übergebene Name der Gruppe mit dem der Zuordnung der Instanzen übereinstimmt. Da die Community-Strings innerhalb einer Anforderung unverschlüsselt übertragen werden, ist durch dieses Konzept keine Sicherheit in Hinblick auf Authentizität, Vertrautheit und Autorisierung gewährleistet.

2.1.6 SNMP-Versionen

Die angesprochenen Aspekte des SNMP-Framework sind der Spezifikation der Version SNMPv2 entnommen. SNMPv1 als ursprüngliche Version enthält einige wichtige Operationen und Fehlertypen nicht, so ist beispielsweise die get-bulk Operation nicht definiert. Auf die Eigenheiten und den reduzierten Funktionsumfang von SNMPv1 muss beim Netzwerkmanagement via SNMP jedoch Rücksicht genommen werden, da viele Agenten in älteren noch benutzten managed nodes nur SNMPv1 unterstützen.

SNMPv3 als Weiterentwicklung von SNMPv2 wurde 1999 spezifiziert. Im wesentlichen führt es ein benutzerbasiertes Sicherheitskonzept und ein sichtbasiertes Zugriffsmodell ein.

2.2 XML basierter Management-Ansatz

2.2.1 Motivation

Das SNMP-Framework hat sich in nun über zehnjähriger Benutzung als nicht frei von Schwachstellen erwiesen. Das Datenmodell in Form der MIB-Module ist zu einem komplexen Gebilde gewachsen. Netzwerkadministratoren bezeichnen SNMP als unsicher, komplex und langsam, als Nischentechnologie ist das Framework uninteressant für Studenten, die stattdessen an generischen Protokollen interessiert sind. Eine weiterführende Motivation für den Übergang zu einem XML-basiertem Netzwerkmanagement ist in [1] zu finden.

2.2.2 XML-Technologien

XML ist als Technologie gut geeignet, Netzwerkmanagement auf einer neuen Basis zu ermöglichen. TCP und HTTP sind heutzutage in nahezu jedem Netzwerkgerät implementiert, und erlauben die Übertragung von XML-Dokumenten als Träger der managementrelevanten Daten. Für sämtliche gängige Programmiersprachen existieren APIs für den Umgang mit XML-Dokumenten, XPath und XSLT erlauben eine generische Verarbeitung der enthaltenen Daten. Zur Beschreibung des Datenmodells, welches für SNMP durch die MIB-Module vorgenommen wird, eignen sich XML Schemas, die zudem eine automatische Validierung der XML-Dokumente ermöglichen. Durch sie wird ein Mapping zwischen der SNMP-basierten und der XML-basierten Adressierung managementrelevanter Objektinstanzen spezifiziert.

Es existieren bereits XML-basierte Produkte für Netzwerkmanagement, auch die IETF hat sich inzwischen mit XML als Grundlage für Konfigurationsmanagement befasst. Eine Erläuterung des als XMLCONF bezeichnetem Ansatz ist unter [8] zu finden.

2.2.3 libsmi - mibdump

Eine umfassende Erörterung der Möglichkeiten eines XML-basierten Netzwerkmanagement leistet der Artikel *Towards XML oriented Internet Management* [2], dem die hier vorgestellten Ansätze entnommen sind. Desweiteren beschreibt der Artikel eine am *Institut für Betriebssysteme und Rechnerverbund, Abteilung Verteile Systeme* entwickelte Softwarebibliothek namens *libsmi*, die eine Übersetzung der in SMI geschriebenen MIB-Moduldefinitionen in XML Schema ermöglicht [9]. Auf dieser Bibliothek aufsetzend wurde dort *mibdump* entwickelt, ein Programm zur Abfrage sämtlicher existierender Objektinstanzen eines Agenten, die innerhalb eines MIB-Modules definiert sind. Die Ausgabe von mibdump erfolgt als XML-Dokument, welches sich durch das entsprechende durch libsmi erstellte XML Schema validieren lässt.

libsmi stellt durch die erzeugten XML Schema ein Mapping der Objektadressierungen bereit. Die durch OIDs vorgegebene Adressierung von Objektinstanzen innerhalb des SNMP-Frameworks wird durch Definition der Objektinstanzen als XML-Elemente in eine XML-basierte Identifizierung übersetzt. Beispielsweise resultiert die Übersetzung des Moduls `TCP-MIB` [10] in einem hier ausschnittsweise angegebenen XML Schema, der die Schema-Definition der Objekte `tcpRtoMin` und `tcpRtoMax` zeigt.

```
<?xml version="1.0"?>
<xsd:schema targetNamespace="http://www.ibr.cs.tu-bs.de/arbeiten/
                             snmp-xml-gw/xsd/TCP-MIB"
            xmlns="http://www.ibr.cs.tu-bs.de/arbeiten/
                   snmp-xml-gw/xsd/TCP-MIB" [...]>
 [...]
 <xsd:element name="snmp-data">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="context" [...]>
```

```
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element name="tcp" type="tcpType" minOccurs="0"/>
      <xsd:element name="tcpConnEntry" type="tcpConnEntryType" [...]/>
     </xsd:sequence>
     <xsd:attribute name="ipaddr" type="xsd:NMTOKEN" use="required"/>
     <xsd:attribute name="hostname" type="xsd:NMTOKEN"/>
     <xsd:attribute name="port" type="xsd:unsignedInt" use="required"/>
     <xsd:attribute name="community" type="xsd:NMTOKEN" use="required"/>
     <xsd:attribute name="caching" type="xsd:NMTOKEN"/>
     <xsd:attribute name="time" type="xsd:dateTime" use="required"/>
    </xsd:complexType>
   </xsd:element>
  </xsd:sequence>
 </xsd:complexType>
</xsd:element>

<xsd:complexType name="tcpType">
 <xsd:sequence>
  [...]
  <xsd:element name="tcpRtoMin" type="smi:Integer32" minOccurs="0">
   <xsd:annotation>
    <xsd:appinfo>
     <maxAccess>read-only</maxAccess>
     <status>current</status>
     <oid>1.3.6.1.2.1.6.2</oid>
     <units>milliseconds</units>
    </xsd:appinfo>
   <xsd:documentation>
    The minimum value permitted by a TCP implementation for the
    retransmission timeout, measured in milliseconds. [...]
   </xsd:documentation>
  </xsd:annotation>
 </xsd:element>
 <xsd:element name="tcpRtoMax" type="smi:Integer32" minOccurs="0">
  <xsd:annotation>
   <xsd:appinfo>
    <maxAccess>read-only</maxAccess>
    <status>current</status>
    <oid>1.3.6.1.2.1.6.3</oid>
    <units>milliseconds</units>
   </xsd:appinfo>
   <xsd:documentation>
    The maximum value permitted by a TCP implementation for the
    retransmission timeout, measured in milliseconds. [...]
   </xsd:documentation>
```

```
   </xsd:annotation>
  </xsd:element>
  [...]
 </xsd:complexType>
 [...]
</xsd:schema>
```

Die Schemata definieren die Datenbasis eines XML-basierten Netzwerkmanagement. Statt eine komplette Neudefinition der Objekte vorzunehmen, werden die durch SNMP bereits spezifizierten Objekte in XML-Elemente übersetzt. Instanzdaten eines Netzwerkknotens können nun durch ein XML-Dokument repräsentiert werden, dessen Aufbau durch ein Schema vorgegeben ist.

```
<?xml version="1.0"?>
<snmp-data xmlns:TCP-MIB="[...]">
  <context ipaddr="127.0.0.1" port=161"
           community="public" time="2003-01-03,14:43.0">
    <TCP-MIB:tcp>
      <TCP-MIB:tcpRtoMin>0</TCP-MIB:tcpRtoMin>
      <TCP-MIB:tcpRtoMax>100</TCP-MIB:tcpRtoMax>
    </TCP-MIB:tcp>
  </context>
</snmp-data>
```

Dieses Beispieldokument beinhaltet die Werte `0` beziehungsweise `100` für die Objektinstanzen `tcpRtoMin` und `tcpRtoMax` eines bestimmten Netzwerkknoten. Der Knoten wird identifiziert durch das `context`-Element, die SNMP-Verbindungsinformationen sind als Attribute angegeben. Dieses Dokument ist bezüglich des oben angegebenen XML-Schema des Moduls `TCP-MIB` valide.

Kapitel 3

Diskussion verschiedener Gateway-Konzepte

Dieses Kapitel beschreibt die Konzeption des zu implementierenden SNMP-XML-Gateways. Verschiedene Möglichkeiten der Realisierung des Gateways werden vorgestellt und erörtert, ebenso wie mögliche Spezifikationen der Anfragen an das Gateway sowie entsprechender Rückgaben.

3.1 Ziel

Der SNMP-XML-Gateway soll es Netzwerkadministratoren und Entwicklern von Managementsoftware ermöglichen, über SNMP administrierbare Netzwerkknoten mittels XML-basierten Anfragen anzusprechen. Dazu stellt eine Client-Anwendung Anfragen auf XML-Basis an das Gateway, der daraufhin die entsprechende SNMP-Kommunikation mit den Agenten der Netzwerkknoten ausführt und das Ergebnis wiederum XML-basiert an den Client zurückübermittelt. Dabei soll das zugrundeliegende SNMP-Framework so weit wie möglich verborgen bleiben, im Idealfall soll eine Administrierung von Netzwerkknoten ohne jedwede Kenntnis SNMP-spezifischer Zusammenhänge rein auf XML-Basis ermöglicht werden. Inwiefern dieses Optimalziel in allen Aspekten erreichbar ist, ist Gegenstand der Diskussion einzelner Entwurfskonzepte.

Prinzipiell soll das Gateway jedoch Netzwerkmanagement im Funktionsumfang des SNMP-Framework ermöglichen, insbesondere also Operationen zur Abfrage und Modifikation managementrelevanter Daten der Netzwerkknoten zur Verfügung stellen. Nach Möglichkeit sollte vermieden werden, dass für bestimmte, für das Management nötige Operationen auf direkte SNMP-Kommunikation zurückgegriffen werden muss.

Die syntaktische und semantische Spezifikation der an das Gateway zu stellenden Anfragen sowie der entsprechenden Antworten ist der kritische Punkt des Gateway-Entwurfes. Diese Spezifikationen sind der Kern eines konsistenten XML-basierten Managementframeworks, somit ist eine detaillierte Diskussion der hierfür in Frage kommenden Alternativen geboten. Die Implementierung selbst dient anschließend als Machbarkeitsdemonstration, dementsprechend werden Aspekte wie Performance und Wartbarkeit der Implementierung nicht im Vordergrund stehen.

3.2 Modell

Abbildung 3.1 zeigt schematisch das Kommunikationsmodell der beteiligten Komponenten.

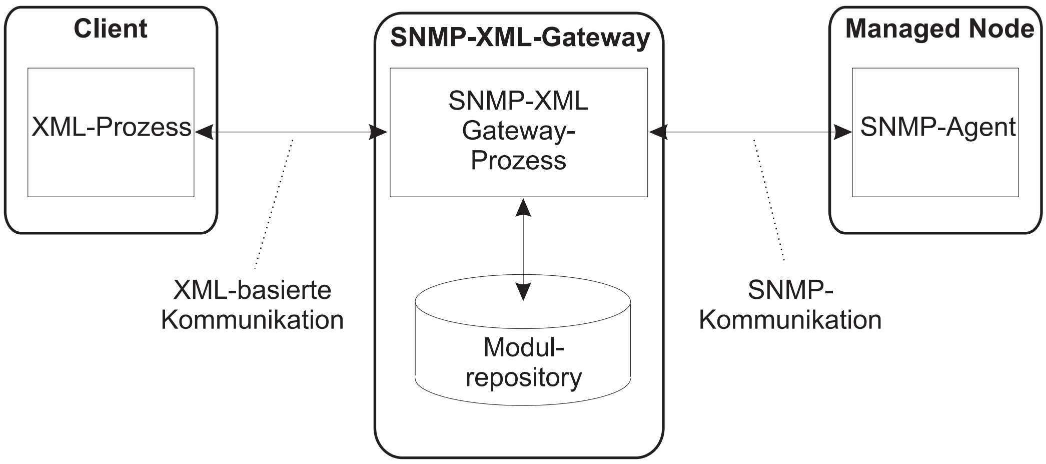

Abbildung 3.1: Gateway-Kommunikation

Das Gateway fungiert als Brücke zwischen den unterschiedlichen Management-Protokollen SNMP und XML-basiertem Protokoll. Für die Übersetzung von Anfragen in das jeweils andere Protokoll benötigt das Gateway eine Informationsbasis. Im Modul-Repository steht hierfür in geeigneter Weise ein Mapping als bijektive Abbildung zwischen Bezeichnern im SNMP-Framework (OIDs) und im XML-basiertem Ansatz zur Verfügung.

3.3 Anforderungen

Dieser Abschnitt beschreibt allgemeine Anforderungen an die Gateway-Implementierung, die als Grundlage der Diskussion verschiedener möglicher Entwürfe im nächsten Abschnitt dienen.

3.3.1 Protokoll

Das Gateway soll als ständig verfügbarer Dienst in heterogenen Netzwerken die spezifizierte Funktionalität zur Verfügung stellen können. Die Übertragung der XML-Daten zwischen Gateway und Client sollte daher in einem höheren Anwendungsprotokoll erfolgen, um keine Rücksicht auf unterschiedliche Interpretationen der übertragenen Daten durch verschiedene Hardwareplattformen nehmen zu müssen. Ein textbasiertes Protokoll ist vorzuziehen, da somit eine einfache Verifikation der textuellen XML-Daten während der Übertragung gegeben ist.

Die Nutzung eines generischen, weit verbreiteten Anwendungsprotokolls birgt zudem den Vorteil, Wahlfreiheit für Systemumgebung und Entwicklungssprache von einfachen Client-Implementierungen zu erhalten. Zwar wäre ein proprietäres Protokoll durch Einsatz einer Middleware oder direkter Socketprogrammierung ebenfalls auf unterschiedlichsten Plattformen nutzbar, dies jedoch nicht ohne die Komplexität der Implementierung einer auf dieser Weise erfolgenden Kommunikation beträchtlich zu erhöhen.

3.3.2 Systemumgebung

Als Minimalforderung gilt die Einsetzbarkeit der Gateway-Implementierung auf einem Unix/Linux-System. Sollte die Implementierung in einer zu kompilierenden Programmiersprache erfolgen, ist trotzdem auf eine leichte Portierbarkeit auf andere Plattformen zu achten. Alternativ steht eine Implementierung in einer interpretierten Programmiersprache zur Verfügung, die plattformübergreifend auf allen Systemen mit einer entsprechenden Laufzeitumgebung einsetzbar wäre.

3.3.3 Bereitgestellte Funktionalität

Die von der Implementierung bereitzustellende Funktionalität ergibt sich aus dem eingangs beschriebenen Ziel, durch den Gateway ein Management im ähnlichen Funktionsumfang des SNMP-Framework zu ermöglichen. Somit muss das Gateway Operationen zur Datenabfrage, Datenmodifikation und zur Behandlung durch SNMP-Agenten initiierter Kommunikation, den SNMP-traps, zur Verfügung stellen. Über diese notwendige Funktionalität hinaus sind erweiternde Operationen denkbar. So könnte das Gateway ein Caching der erfragten SNMP-Daten betreiben, um zeitlich eng zusammenliegende Lesezugriffe auf dieselben Daten eines SNMP-Agenten schneller und mit verringerter Netzbenutzung beantworten zu können. Zudem ist es denkbar, eine je nach Komplexität der Implementierung nötige Administrierung des Gateways selbst über das XML-basierte Management zu ermöglichen.

Wie jedoch ebenfalls im Abschnitt Ziele erwähnt, dient die Implementierung als Machbarkeitsdemonstration der getroffenen Spezifikationen. Aus diesem Grund wird das Hauptaugenmerk auf einer Implementierung der notwendigen Funktionalität gelegt werden, jedoch nicht ohne die Implementierung erweiternder Operationen zumindest konzeptionell zu skizzieren.

3.4 Konzepte

Die im vorherigen Abschnitt gestellten Anforderungen zugrundelegend, werden mögliche Gateway-Spezifikationen erörtert. Neben globalen Designfragen wie dem verwendeten Protokoll betrachtet dieser Abschnitt verschiedene Ansätze der syntaktischen und semantischen Spezifikation der Gateway-Anfragen und -Antworten. Während sich für einige Aspekte der späteren Implementierung bereits hier definitiv entschieden wird, dient ein Großteil der Überlegungen als Grundlage der im nächsten Kapitel endgültig erfolgenden Spezifikation.

3.4.1 Protokoll

Das HyperText Transport Protocol (HTTP) als weit verbreitetes, textbasiertes Anwendungsprotokoll wird zur Kommunikation zwischen Gateway und Client eingesetzt werden. Neben der Erfüllung der an das Protokoll gestellten Anforderungen bietet es zudem den Vorteil, die Wahlfreiheit der für die Implementierung auszuwählenden Systemumgebung zu erhalten. So besteht die Möglichkeit, das Gateway als eigenen HTTP-Server zu implementieren oder es als Servletprozess oder Skript-Anwendung in bestehende Webserver zu integrieren.

Neben der direkten Nutzung von HTTP als Kommunikationsprotokoll besteht die Möglichkeit, die Gateway-Funktionalität als Web-Service zur Verfügung zu stellen. Hierdurch würden Methoden definiert, die entfernt gemäß des *SOAP (Simple Objects Access Protokoll)* [11] aufgerufen werden könnten. Die XML-basierten Daten zum Management wären Parameter solcher Aufrufe und würden ebenfalls über HTTP übertragen werden. Die Bereitstellung der Gateway-Funktionalität als Web-Service ist weniger sinnvoll als die direkte Nutzung des HTTP, da wenige verschiedene Operationen zur Verfügung gestellt werden. Wie in den folgenden Abschnitten diskutiert, handelt es sich dabei um eine Leseanforderung, eine Schreibanforderung und um Operationen zur Übermittlung der traps. Für eine Differenzierung dieser Operationen in einer Anfrage bietet HTTP jedoch ausreichend Möglichkeiten.

Durch die Festlegung auf HTTP wird im Folgenden neben der eigentlichen Spezifikation der Anfragen und Antworten auch die Art des Transportes dieser zu erörtern sein. HTTP bietet neben den Anfragemethoden GET und POST verschiedene Möglichkeiten des Datentransportes. So können Daten in dem Header wie auch in dem Messagebody einer HTTP-Anforderung übermittelt werden.

3.4.2 Adressierung

Sowohl für Lese- wie auch für Schreibanforderungen muss ein Client dem Gateway mitteilen, auf welche Daten eines bestimmten Netzwerkknotens seine Anfrage bezogen ist. Da das Gateway die erwünschte Operation durch SNMP-Kommunikation mit dem Agenten des entsprechenden Gerätes ausführt, muss es in der Lage sein, die XML-basierten Adressierungen der Informationen in entsprechende Adressierungen durch OIDs zu übersetzen. Es ergeben sich im wesentlichen zwei Möglichkeiten eines Mappings der beiden Adressierungsarten. So kann auf ein bereits spezifiziertes Mapping zurückgegriffen oder eine Neuspezifikation vorgenommen werden.

Neuspezifikation der XML-Adressierung

Es gilt, eine eigene auf entsprechende SNMP-Adressierung abbildbare Adressierung managementrelevanter Daten durch XML zu definieren. Dabei muss diese Abbildung surjektiv sein, schließlich sollten in einer Anfrage an das Gateway alle Objektinstanzen innerhalb des SNMP-Framework adressierbar sein.

Sämtlichen Objektnamen der in den MIB-Modulen definierten Objekte sind durch die Angabe von OIDs Adressen zugewiesen. Diese Abbildung ist nicht bijektiv, in unterschiedlichen Modulen definierte Objekte können den selben Namen besitzen, solange verschiedene OIDs zugewiesen werden. Die OIDs hingegen sind eindeutig, eine einfache Adressierung in XML könnte somit durch die Nennung der OID erfolgen. Dieser Ansatz überträgt jedoch das SNMP-interne Konzept unverändert in die XML-Spezifikation und erfüllt somit nicht den gestellten Anspruch, das SNMP-Framework möglichst zu verbergen.

Um die streng hierarchische Adressierung der SNMP-Objekte durch den OID-Baum zu verbergen, ist es nötig, die XML-Adressierung auch aus XML-Sicht zu spezifizieren. Eine solche Spezifikation ist nicht trivial, bei vollem Gebrauch der in XML zur Verfügung stehenden Mittel wie Element-Hierarchie, Attribute und Namespaces ist es schwierig, bei größtmöglicher Einfachheit die Konsistenz zu waren. Als weitere Schwierigkeit stellt sich die Aufgabe dar, die schon bestehenden Inkonsistenzen der SNMP-Objektdefinitionen durch die XML-Spezifikation möglichst zu beseitigen.

Nutzung eines bereits spezifizierten Mappings

Es existieren verschiedene bereits spezifizierte Mappings zwischen MIB-Definitionen, XML und XML-Schema, die somit auch ein Mapping der Objektadressierungen bereitstellen. Beispielsweise hat die IRTF Network Management Research Group (IRTF-NRMG) ein Mapping zwischen MIB-Definitionen und XML-Adressierungen definiert [8], welches jedoch inzwischen als veraltet gilt. Allgemein sind solche Spezifikationen Bestandteil aktueller Forschungen auf diesem Gebiet, es existieren verschiedene Ansätze, die im Zuge der Weiterentwicklung verändert oder durch neue ersetzt werden.

Die im Abschnitt 2.2.2 vorgestellte Softwarebibliothek libsmi [12] stellt ebenfalls ein Mapping zur Verfügung. Durch die aus Übersetzung der Moduldefinitionen entstandenen XML-Schema ist eine bijektive Abbildung zwischen OIDs und benannten XML-Elementen gegeben. Über den Namen der XML-Elemente einer XML-Datei, die nach einem entsprechenden XML-Schema valide ist, lassen sich in dem Schema die OIDs dieser SNMP-Objekte wiederfinden. Durch mibdump ist dieses Mapping erprobt, ein Auslesen sämtlicher Objektinstanzen eines MIB-Moduls, die von einem SNMP-Agenten bereitgestellt werden, ist bereits möglich.

3.4.3 Leseanforderungen

Leseanforderungen an bestimmte Netzwerkknoten sind ein fundamentaler Bestandteil des Netzwerkmanagement. Ohne Kenntnis der aktuellen Konfigurationen im Netzwerk eingebundener Geräte ist ein gegenseitiges Abstimmen dieser nicht möglich. Clients muss demnach die Möglichkeit gegeben werden, Anfragen nach den Attributswerten bestimmter Objektinstanzen zu stellen. Die Adressierung der relevanten Objektinstanzen erfolgt dabei nach einem oben diskutiertem Ansatz. Dabei muss der Client dem Gateway die abzufragenden Objektinstanzen in Form der XML-Adressierung und den betreffenden Netzwerkknoten in Form eines DNS-Namens oder direkt durch die IP-Adresse übermitteln.

Für die genaue Syntax einer solchen durch HTTP übermittelten Anfrage bestehen dabei mehrere Möglichkeiten. So könnte ein Client ein XML-Dokument an das Gateway übermitteln, welches die XML-Elemente enthält, an denen der Client interessiert ist. Das Gateway ermittelt die Instanzwerte der entsprechenden Objekte per SNMP-Kommunikation und sendet das XML-Dokument mit den eingetragenen Werten an den Client zurück. Eine andere Möglichkeit besteht in der Übermittlung der XML-Adressierung der abzufragenden Objektinstanzen in geeigneter Form an das Gateway.

Leseanforderung durch Übermittlung eines XML-Dokumentes

Der Client könnte ein komplettes XML-Dokument gemäß der Mappingspezifikation per HTTP-Methode POST oder PUT an das Gateway übertragen, eine geeignete Antwort des Gateways bestünde in der Rückübermittlung des Dokumentes, in dem nun die entsprechenden Attributswerte des abgefragten Netzwerkknotens als XML-Elementwerte eingetragen sind. Die übermittelte Antwort ist dabei ein nach dem jeweiligen XML Schema des SNMP-XML Mappings validierbares XML-Dokument.

Interessieren einen Client die Attributswerte der Objekte `tcpRtoAlgorithm` und `tcpRtoMin` der Objektgruppe `tcp`, so kann das übermittelte Dokument gemäß des libsmi-Mappings wie folgt aussehen:

```
<?xml version="1.0"?>
<snmp-data xmlns:TCP-MIB="[...]">
  <context [...] hostname="test1.snmptest.net" community="public">
    <TCP-MIB:tcp>
      <TCP-MIB:tcpRtoAlgorithm></TCP-MIB:tcpRtoAlgorithm>
      <TCP-MIB:tcpRtoMin></TCP-MIB:tcpRtoMin>
   </TCP-MIB:tcp>
  <context>
</snmp-data>
```

Das Wurzelelement `snmp-data` definiert Namespaces für jedes MIB-Modul, aus dem Objekte zur Abfrage angegeben werden. Die Angabe des abzufragenden Netzwerkknoten erfolgt in `context`-Elementen durch Attribute. Auf diese Weise können durch ein Dokument Daten mehrerer Netzwerkknoten repräsentiert werden. Die Angaben der Objekte, an deren Instanzwerten der übermittelnde Client interessiert ist, erfolgen als Unterelemente des jeweiligen `context`-Elements. Sie werden dabei mit dem Namespace-Prefix ihres definierenden MIB-Moduls gekennzeichnet, dies erlaubt eine sofortige Zuordnung der angegebenen Objekte zu ihren jeweiligen Definitionen. Eine entsprechende Antwort des Gateways besteht wie erläutert aus dem mit Elementwerten versehenen Dokument:

```
<?xml version="1.0"?>
<snmp-data xmlns:TCP-MIB="[...]">
  <context [...] hostname="test1.snmptest.net" community="public">
    <TCP-MIB:tcp>
      <TCP-MIB:tcpRtoAlgorithm>other</TCP-MIB:tcpRtoAlgorithm>
      <TCP-MIB:tcpRtoMin>0</TCP-MIB:tcpRtoMin>
   </TCP-MIB:tcp>
  <context>
</snmp-data>
```

Dabei kann dieses Dokument im HTTP-Body der Gateway-Antwort übermittelt werden.

Leseanforderung durch Angabe der abzufragenden Objektinstanzen

Mit *XPath* [13] und *XQuery* [14] existieren erprobte Konzepte, um bestimmte Elemente eines XML-Dokumentes zu adressieren. XPath wird in Style-Sheets gemäß der *Extensible Style Sheet Language (XSL)* eingesetzt, um bestimmte Elemente eines XML-Dokumentes zu identifizieren und eine Umwandlung der Dokumente in ein anderes Ausgabeformat zu ermöglichen [15]. XQuery stellt ein an der *Structured Query Language (SQL)* angelehntes Konstrukt dar, um ähnlich der Anfragesemantik bei relationalen Datenbanken eine gezielte Elementidentifizierung aus XML-Dokumenten zu ermöglichen.

Aus Client-Perspektive kann das Gateway als Vorhaltepunkt für ein einzelnes umfangreiches XML-Dokument angesehen werden, welches die Attributswerte aller Objektinstanzen in sämtlichen Netzwerkknoten beinhaltet. Unterhalb eines Wurzelelementes liegen die Knoten, die als Unterelemente über die Repräsentationen der einzelnen Module verfügen. In diesen Modulelementen werden die entsprechenden Instanzwerte der jeweiligen Objektdefinitionen gemäß einem bestimmten Mapping, zum Beispiel dem der libsmi, vorgehalten.

```
<snmp-data xmlns:TCP-MIB="[...]"
           xmlns:IF-MIB="[...]"
           [...]>

  <context hostname="test1.snmptest.net" [...]>
    <TCP-MIB:tcp>
      <TCP-MIB:tcpRtoAlgorithm>other</TCP-MIB:tcpRtoAlgorithm>
      <TCP-MIB:tcpRtoMin>0</TCP-MIB:tcpRtoMin>
      [...]
    </TCP-MIB:tcp>
    [...]

    <IF-MIB:interfaces>
      [...]
    </IF-MIB:interfaces>
    [...]
    <IF-MIB:ifEntry ifIndex="1">
```

```
      [...]
    </IF-MIB:ifEntry>
    [...]
  </context>

  <context hostname="test2.snmptest.net" [...]>
    <TCP-MIB:tcp>
      <TCP-MIB:tcpRtoAlgorithm>vanj</TCP-MIB:tcpRtoAlgorithm>
      <TCP-MIB:tcpRtoMin>0</TCP-MIB:tcpRtoMin>
      [...]
    </TCP-MIB:tcp>
    [...]
  </context>

</snmp-data>
```

Durch Übermittlung eines XPath-Ausdruckes kann ein Client nun gezielt Elemente dieses vorgestellten Dokumentes adressieren. Das Gateway ermittelt entsprechend der identifizierten Elemente die tatsächlichen Attributswerte der Netzwerkknoten durch SNMP-Kommunikation. Anschließend wird ein nach dem jeweiligen Schema valides XML-Dokument erstellt und dem Client übermittelt. Beispielsweise erhält der Client durch Angabe des XPath-Ausdruckes

```
/snmp-data/context[@name="test1.snmptest.net"]
 /tcp/(tcpRtoAlgorithm|tcpRtoMin)
```

das auch schon im Beispiel der indirekten Leseanforderung erhaltene XML-Dokument:

```
<?xml version="1.0"?>
<snmp-data xmlns:TCP-MIB="[...]">
  <context [...] hostname="test1.snmptest.net" community="public">
    <TCP-MIB:tcp>
      <TCP-MIB:tcpRtoAlgorithm>other</TCP-MIB:tcpRtoAlgorithm>
      <TCP-MIB:tcpRtoMin>0</TCP-MIB:tcpRtoMin>
   </TCP-MIB:tcp>
  <context>
</snmp-data>
```

Hier erscheint es ebenfalls sinnvoll, die Gateway-Antwort im HTTP-Body zu übermitteln. Die Anfrage durch Übermittlung des XPath-Ausdruckes kann neben der Verwendung der HTTP-Methode POST auch über die Methode GET direkt in einem URL erfolgen.

Zwar wird ein hierfür verwendeter XPath-Ausdruck nach den geltenden Syntaxregeln gebildet, jedoch ist zu beachten, dass die Rückgabe von dem üblichen XPath-Verständnis abweicht. Während das Auswertungsergebnis eines XPath-Ausdruckes auf ein XML-Dokument definitionsgemäß eine hierarchielose Liste sämtlicher durch den Ausdruck identifizierter XML-Elemente darstellt, wird hier dem Client ein komplettes XML-Dokument übermittelt. Die Semantik hier entspricht dem Verständnis einer URL, durch die ein XML-Dokument von einem Webserver angefordert werden kann. Diesem Verständnis entsprechend ist die Übermittlung der direkten Leseanforderung durch einen XPath-Ausdruck als HTTP-GET-Request naheliegend.

3.4.4 Schreibanforderungen

Für Modifikationen der Attributswerte eines Netzwerkknotens existiert nach den Überlegungen zu Leseanforderungen nur ein schlüssiges Konzept. Da dort bei den als möglich erachteten Konzepten die Rückgabe der angeforderten Daten in Form eines XML-Dokumentes erfolgt, ist es naheliegend, für Schreibanforderungen eben ein solches XML-Dokument an das Gateway zu übermitteln. Genau die enthaltenen Elementwerte werden vom Gateway als Bestandteil entsprechender set-Operationen einer SNMP-Kommunikation gesetzt. Zudem ist es nötig, die durch das vom Client übertragene Dokument gegebene XML-Adressierung in die jeweilige SNMP-Adressierung durch OIDs umzuwandeln, dies ist durch das bijektive Mapping der Adressierungen realisierbar.

Die Übertragung des Dokumentes an das Gateway kann wie schon bei der indirekten Leseanforderung durch die HTTP-Methoden POST oder PUT erfolgen. Eine Antwort des Gateways besteht in der Übermittlung von Informationen bezüglich möglicher aufgetretener Fehler während der Bearbeitung der Schreibanforderung. Die Übertragung von Fehlermeldungen wird im Abschnitt 3.4.6 behandelt.

Dieser Ansatz ermöglicht dem Client eine schlüssige Arbeitsweise für das XML-basierte Konfigurationsmanagement. Der Client erhält aktuelle Attributswerte eines Knotens durch direkte oder indirekte Adressierung der interessanten Daten in Form eines XML-Dokumentes vom Gateway. Die Werte können nun vom Client unter Nutzung sämtlicher für Bearbeitung von XML-Dokumenten zur Verfügung stehender Techniken betrachtet und gegebenenfalls modifiziert werden. Zur Übertragung der geänderten Werte in den Netzwerkknoten wird das modifizierte XML-Dokument an das Gateway zurückübermittelt.

Ebenso kann innerhalb der Client-Anwendung eine Bibliothek von XML-Dokumenten aufgebaut werden, um unterschiedliche Konfigurationen verschiedener Netzwerkknoten vorzuhalten. Durch Übertragung eines dieser Dokumente wird die enthaltene Konfiguration in den entsprechenden Netzwerkknoten übernommen.

Dieses Konzept der Schreibanforderungen beinhaltet jedoch Einschränkungen. So ist es auf diese Weise nur möglich, Werte schon bestehender Objektinstanzen zu verändern. Es ist nicht möglich, durch eigene Operationen neue Objektinstanzen durch Anlegen neuer Tabellenzeilen zu erzeugen beziehungsweise Instanzen durch Entfernen der entsprechenden Tabellenzeile zu löschen. Das SNMP-Framework stellt als einzige Methode zur Datenveränderung die *set*-Operation zur Verfügung, das Erzeugen und Löschen von Objektinstanzen erfolgt durch set-Anweisungen auf bestimmte *row status*-Objekte. Es besteht die Möglichkeit, für XML-basierte Anfragen neue Operationen zum direkten Erzeugen und Löschen von Objektinstanzen einzuführen, das Gateway müsste diese Operationen durch entsprechende set-Anweisungen innerhalb der SNMP-Kommunikation nachbilden. Wegen des zu erwartenden Aufwands der Implementierung werden im Rahmen dieser Arbeit solche Operationen der XML-basierten Kommunikation nicht eingeführt. Dem Benutzer des Gateways bleibt jedoch die Möglichkeit, durch set-Operationen auf den row status-Objekten Instanzen zu erzeugen beziehungsweise zu löschen.

3.4.5 Traps

Durch das Gateway muss ein Mechanismus zur Verfügung gestellt werden, die von SNMP-Agenten initiierten traps zu empfangen. Einem Gateway-Client muss es ermöglicht werden, die in den traps enthaltenen Informationen einzusehen. Dafür ist es zunächst erforderlich, die SNMP-Agenten dahingehend zu konfigurieren, trap-Benachrichtigungen an das Gateway zu übermitteln. Diese Konfiguration erfolgt durch Wertzuweisungen an Instanzen entsprechender SNMP-Objekte des Agenten, und kann somit durch Clients über den oben beschriebenen Weg der Schreibanforderungen vorgenommen werden. Für die Weiterübermittlung der Informationen in einem entsprechenden XML-Format an die Clients sind zwei Ansätze denkbar. So könnte das Gateway als Relay fungieren, die aufgelaufenen traps also für eine vom Client initiierte Abfrage vorhalten. Als weiteren Ansatz ist die sofortige Zustellung der trap-Informationen an interessierte Clients mittels eines geeigneten Protokolls denkbar.

Gateway als trap-Relay

Bei dem Betrieb des Gateways als trap-Relay verwaltet das Gateway aufgelaufene SNMP-traps. Die trap-Informationen können von Clients als XML-Dokument abgefragt werden. Die Informationen müssen dabei von den Clients adressiert werden können. Wie in 2.1.4 dargestellt tritt eine SNMP-trap immer für eine bestimmte Objektinstanz auf. Diese Instanz kann durch das Mapping der SNMP- und XML-Adressierung durch die Clients gezielt angegeben und das Gateway nach hierfür aufgelaufenen traps abgefragt werden. Als weitere Information übermittelt ein SNMP-Agent die Art des traps durch Angabe eines trap-OID in der SNMP-Kommunikation. Diese OID ist ebenfalls durch eine XML-Adressierung zugänglich, da durch das benutzte Mapping auch die im Modul SNMPv2-MIB spezifizierte Gruppe snmpTrap durch das XML-Schema dieses Moduls adressierbar ist.

Ähnlich zu den Überlegungen der Leseanforderung von Instanzwerten existieren für Leseanforderungen aufgelaufener traps zwei Möglichkeiten. So kann der Client eine Leseanforderung für trap-Daten durch Übermittlung eines XML-Dokumentes oder durch Übermittlung einer Adressierung der interessanten Objektinstanzen stellen. Im Beispiel wird hier die Anforderung durch Übermittlung einer Adressierung verdeutlicht.

Durch Angabe des XPath-Ausdruckes

```
/snmp-data/context[@name="test1.snmptest.net" and @trap="yes" and
                   @time>"2002-11-14T11:48"]/ifEntry[@ifIndex="7"]
```

fragt der Client an, sämtliche für das Netzwerk-Interface mit dem Index sieben aufgelaufene traps des Knotens `test1.snmptest.net` übermittelt zu bekommen. Das geforderte Attribut `@trap="yes"` zeigt dem Gateway dabei an, dass der Client bei dieser Anfrage trap-Informationen anfordert. Die Attributsangabe `@time>"2002-11-14T11:48"` schränkt die

Anfrage zusätzlich ein, angefordert werden nur die traps die nach 11:48 Uhr des 14.11.2002 aufgelaufen sind.

Die Antwort des Gateways besteht in einem XML-Dokument, welches die aufgelaufenen traps enthält.


```
<?xml version="1.0"?>
<snmp-data xmlns:IF-MIB="[...]"
           xmlns:SNMPv2-MIB="[...]">
  <context hostname="test1.snmptest.net" trap="yes" community="public">

    <IF-MIB:linkUp timestamp="2002-11-14T12:23">
      <IF-MIB:ifEntry ifIndex="7">
        <IF-MIB:ifAdminStatus>[...]
        <IF-MIB:OperStatus>[...]
        [...]
      </IF-MIB:ifEntry>
    </SNMPv2-MIB:snmpTrap>

    <IF-MIB:linkDown timestamp="2002-11-16T00:42">
      <IF-MIB:ifEntry ifIndex="7">
        <IF-MIB:ifAdminStatus>[...]
        <IF-MIB:OperStatus>[...]
        [...]
      </IF-MIB:ifEntry>
    </SNMPv2-MIB:snmpTrap>

    [...]
  </context>
</snmp-data>
```


Sofortige Zustellung der trap-Informationen

Um interessierten Clients bestimmte aufgelaufene traps ohne größere Zeitdifferenz zu übermitteln und dabei ein periodisches Anfragen (polling) der Clients zu verhindern, müssen die Clients durch eine vom Gateway initiierte Kommunikation über traps unterrichtet werden. Dafür müssen die Clients eine Schnittstelle für eingehende Kommunikation implementieren. Im wesentlichen bestehen für die vom Gateway übermittelte Information zwei Möglichkeiten:

Komplette Übermittlung eines XML-Dokumentes: Der Client stellt eine Schnittstelle für eingehende HTTP-Anfragen bereit, das Gateway übermittelt hierüber das XML-Dokument, welches die aufgelaufene SNMP-trap repräsentiert. Hinsichtlich der nötigen Kommunikation für Benachrichtigung und Übermittlung ist dies der effizienteste Ansatz, jedoch erhöht die Implementierung eines HTTP-Servers die Komplexität möglicher Clients beträchtlich.

Benachrichtigung der Clients: Das Gateway übermittelt dem Client eine Benachrichtigung, dass für ein beobachtetes Objekt eine SNMP-trap empfangen wurde. Der Client kann daraufhin die aufgelaufenen traps abfragen. Das Gateway fungiert bei diesem Ansatz als trap-Relay, besitzt jedoch zusätzlich die Möglichkeit zur Benachrichtigung der interessierten Clients. Die Benachrichtigung kann dabei auch weitere Informationen enthalten, beispielsweise die XML-Adressierung des trap-auslösenden SNMP-Objektes als XPath-Ausdruck. Der Aufwand für den Client verringert sich dadurch, statt die traps aller beobachteten Objekte abzufragen, reduziert sich die Abfrage auf traps eines Objektes. Die vom Client bereitzustellende Schnittstelle ist bei diesem Ansatz weniger komplex und kann auch durch direkte Socketkommunikation realisiert werden.

3.4.6 Fehlertoleranz - Fehlermeldungen

Eine hohe Verfügbarkeit der als Dienst vom Gateway bereitgestellten Funktionalität ist wünschenswert. Aus diesem Grund sollte die Implementierung möglichst robust bezüglich Kommunikationsfehlern und fehlerhaft übertragener Daten sein.

Um aussagekräftige Fehlermeldungen bei fehlerhaften Leseanfragen der Clients zu liefern, besteht die Möglichkeit, statt der via SNMP-Kommunikation erlangten Daten die vom SNMP-Agenten gelieferten Fehlermeldungen als Elementwerte in das zurückzugegebene XML-Dokument einzufügen. Dieses Vorgehen verbirgt dem Client gegenüber jedoch nicht die Aspekte des SNMP-Frameworks. Da allerdings kein Mapping von SNMP-Fehlermeldungen auf einen XML-konsistenten Ansatz existiert, besteht im Rahmen dieser Arbeit keine Möglichkeit im Fehlerfall das SNMP-Framework zu verbergen. Somit wird eine Implementierung dieser Art der Fehlermeldungen im weiteren Entwurf zu diskutieren sein.

Zumindest sollte das Gateway einem anfragenden Client aber überhaupt das Auftreten eines Fehlers während der Anfragebearbeitung mitteilen können. Hierfür bieten sich die HTTP-Status-Codes an, die eine grobe Differenzierung der möglichen Fehler in Fehlerklassen anhand definierter, dreistelliger Zahlencodes erlauben, ergänzt um die zusätzliche Möglichkeit, einen frei wählbaren Text als Fehlerbeschreibung zurückzuliefern. Tabelle 3.1 listet die fünf Klassen der Status-Codes mit einigen Beispielen auf.

Code	Beschreibung	Beispiele
1xx	Informational	100 Continue, 101 Switching Protocols
2xx	Successful	200 OK, 201 Created, 202 Accepted
3xx	Redirection	300 Multiple Choices, 301 Moved Permanently
4xx	Client Error	400 Bad Request, 401 Unauthorized
5xx	Server Error	500 Internal Server Error, 501 Not Implemented

Tabelle 3.1: Status Codes der HTTP Response

So wird das Gateway bei erfolgreich ausgeführten Operationen eine Antwort mit Status-Code 200 übermitteln. Für fehlerhafte Anfragen bietet sich ein Status-Code der Gruppe 4xx an.

3.4.7 Caching und Performance

Das Gateway bietet die Möglichkeit, von Netzwerkknoten via SNMP-Kommunikation abgefragte Instanzwerte zu cachen. Somit kann eine erneute Anfrage auf dieselben Werte eines Knotens durch die vom Gateway gespeicherten Daten beantwortet werden, ohne dafür den SNMP-Agenten erneut abzufragen. Jedoch ist es wünschenswert überprüfen zu können, ob die zwischengespeicherten Werte inzwischen durch andere Management-Anwendungen in dem betreffenden SNMP-Agenten geändert wurden. Leider stellt SNMP hierfür keinen generisch nutzbaren Mechanismus, wie etwa abfragbare timestamps für Wertänderungen, zur Verfügung. Aus diesem Grund kommt als einziger Cachingansatz nur ein kurzzeitiges Zwischenspeichern der Daten in Betracht, eventuell ergänzt durch die Möglichkeit von Clients, das Gateway explizit zur Neuabfrage der erwünschten Attributswerte anzuweisen.

Es ist zu erwarten, dass die Antwortzeiten einer über das Gateway durchgeführten Operation durch die zwei Stufen der Netzkommunikation höher ausfallen werden als bei direkter SNMP-Kommunikation zwischen Manager und Agent. Obwohl es abzuwarten bleibt, in welchem Maße die Antwortzeit steigt und ob das Management hierdurch spürbar beeinträchtigt ist, sollten zumindest die gegebenen Möglichkeiten einer Performanceerhöhung frühzeitig diskutiert werden. So besteht die auch schon durch übliche Managementanwendungen genutzte Möglichkeit, die Bearbeitungszeit von SNMP-Operationen durch Parallelisierung zu verkürzen. So können in mehreren Threads gleichzeitig Operationen auf einem Agenten durchgeführt werden. Eingangs dieses Kapitels wurde erwähnt, dass die Implementierung auf Performanceaspekte weniger Rücksicht nehmen wird, dies schließt jedoch nicht aus, dass bei einfacher Implementierbarkeit performanceverbessernde Maßnahmen Eingang finden werden.

Kapitel 4

Spezifikation des Gateways

In diesem Kapitel werden aus den verschiedenen im Kapitel 3 diskutierten Konzepten zur Arbeitsweise des SNMP-XML-Gateways die der späteren Implementierung zugrundeliegenden Entwürfe ausgewählt und formal spezifiziert.

4.1 Adressierung

Zur Bereitstellung des Mappings zwischen SNMP- und XML-Adressierung managementrelevanter Objektinstanzen wird der xsd-Treiber des Programmes *smidump* benutzt. Somit stehen dem Gateway als Modul-Repository durch die libsmi erstellte XML-Schemadefinitionen der MIB-Module zur Verfügung. Die in Anfragen an das Gateway zu benutzende XML-Adressierung der Objektinstanzen ist ebenfalls durch diese Schema definiert.

4.2 Leseanforderungen

4.2.1 Syntaktische und semantische Spezifikation

Für Leseanforderungen wird der Ansatz der Leseanforderung durch Übermittlung einer Anforderungszeichenkette an den Gateway gewählt. Diese im Folgenden Request-String genannte Zeichenkette enthält einen XPath-Ausdruck, durch den ein Client die Objektinstanzen der von ihm gewünschten Werte adressiert. Die Leseanforderung erfolgt als HTTP-GET-Request, so kann die Leseanforderung in Form eines URL an den als Servlet zu implementierenden Gateway übertragen werden.

Als Syntaxdefinition des Request-Strings ergibt sich somit folgende EBNF:

```
REQUEST-STRING    := '/snmp-data' REQUESTS
REQUESTS          := CONTEXTREQUEST | '('CONTEXTS')'
CONTEXTS          := CONTEXTS '|' CONTEXTS | CONTEXTREQUEST
CONTEXTREQUEST    := '/context[' NODE OPTCONTEXTATTR ']'<XPATH>
NODE              := '@ipaddr="' <IPADDR> '"' |
```

```
                    '@hostname="' <DNS-NAME> '"' |
                    '@ipaddr="' <IPADDR> '"'
                     and @hostname="' <DNS-NAME> '"'

OPTCONTEXTATTR   := '' | OPTCONTEXTATTR  OPTCONTEXTATTR |
                    'and @community="' <SNMP-COMMUNITY-STRING> '"'
                    'and @port="' <NODE-SNMP-PORT> '"' |
                    'and @caching="' TRUTH-VALUE '"'   |
                    'and @trapdata="' TRUTH-VALUE '"'  |

TRUTH-VALUE      := 'yes' | 'no'
```

Die nicht weiter spezifizierten Ausdrücke <IPADDR>, <DNS-NAME> und <NODE-SNMP-PORT> beinhalten die Angaben IP-Adresse, Hostname und Port des SNMP-Agenten nach üblichem Verständnis. <SNMP-COMMUNITY-STRING> als nicht weiter spezifizierter Ausdruck dient zur Aufnahme des SNMP-Community-Strings als Zugriffsautorisation für die angegebenen Objektinstanzen (vgl. 2.1.5).

Durch den Ausdruck `CONTEXTS` erfolgt die Angabe der abzufragenden Kontexte. Hier können beliebig viele Kontexte innerhalb eines `REQUEST-STRING` angegeben werden, die Angabe mindestens eines Kontextes ist jedoch Pflicht. Durch `CONTEXTREQUEST` wird ein Kontext mit den nötigen Parametern zur SNMP-Kommunikation versehen und der die Objektinstanzen auswählende XPath-Ausdruck angehängt. Bei der gleichzeitigen Angabe von `@ipaddr` und `@hostname` ist zu beachten, dass widersprüchliche Werte gemäß der XPath-Vorstellung zu einem leeren Ergebnisdokument bezüglich diesen Kontextes führen. Die optionale Attributsangabe `@trapdata` drückt aus, ob die Standardwerte der Objektinstanzen (`@trapdata="no"`) oder Informationen gemäß der vom Gateway empfangenen notifications angefordert werden (`@trapdata="yes"`).

Standardwerte der optionalen Attribute sind:

```
@community : public
@port      : 161
@caching   : yes
@trapdata  : no
```

Dem Ansatz dieser Leseanforderung liegt nach 3.4.3 (Seite 19) das Verständnis des Gateways als Vorhaltepunkt eines umfassenden XML-Dokumentes zugrunde, aus dem durch den Request-String bestimmte Teile zur Übermittlung angefordert werden. Im Unterschied zur dort getroffenen Definition dieses Dokumentes werden nach dieser Spezifikation die Modulnamen nicht als Unterelemente der Netzwerkknoten repräsentiert. Das implizite durch den Gateway vorgehaltene Dokument ist demnach wie folgt definiert:

```
<snmp-data>

   <context hostname="test1.snmptest.net" [...]>
   [...]
      <IF-MIB:interfaces>
```

```
      <IF-MIB:ifNumber>2</IF-MIB:ifNumber>
   </IF-MIB:interfaces>
   <IF-MIB:ifEntry ifIndex="1">
      <IF-MIB:ifDescr>lo</IF-MIB:ifDescr>
      <IF-MIB:ifType>softwareLoopback</IF-MIB:ifType>
      <IF-MIB:ifMtu>16436</IF-MIB:ifMtu>
      [...]
   </IF-MIB:ifEntry>
   <IF-MIB:ifEntry ifIndex="2">
      <IF-MIB:ifDescr>eth0</IF-MIB:ifDescr>
      <IF-MIB:ifType>ethernetCsmacd</IF-MIB:ifType>
      <IF-MIB:ifMtu>1500</IF-MIB:ifMtu>
      [...]
   </IF-MIB:ifEntry>
   [...]
   <TCP-MIB:tcp>
      <TCP-MIB:tcpRtoAlgorithm>other</TCP-MIB:tcpRtoAlgorithm>
      <TCP-MIB:tcpRtoMin>0</TCP-MIB:tcpRtoMin>
      <TCP-MIB:tcpRtoMax>0</TCP-MIB:tcpRtoMax>
      [...]
   </TCP-MIB:tcp>
[...]
</context>

<context hostname="test2.snmptest.net" [...]>
[...]
   <TCP-MIB:tcp>
      <TCP-MIB:tcpRtoAlgorithm>constant</TCP-MIB:tcpRtoAlgorithm>
      <TCP-MIB:tcpRtoMin>0</TCP-MIB:tcpRtoMin>
      <TCP-MIB:tcpRtoMax>0</TCP-MIB:tcpRtoMax>
      [...]
   </TCP-MIB:tcp>
[...]
</context>

</snmp-data>
```

Objektgruppen sind als Unterelemente ihres Netzwerkknoten, repräsentiert durch das `context`-Element, angeordnet. Im gegebenen Beispiel ist die Gruppe `tcp` ein direktes Unterelement eines `context`-Elementes und beinhaltet die skalaren Attribute der in der Gruppe definierten Objekte. Instanzen von Objekten, die in einem MIB-Modul innerhalb einer Tabelle definiert sind, werden ebenfalls durch direkte Unterelemente eines context-Elementes repräsentiert. So stellt das Element `<ifEntry ifIndex="1">` die erste Zeile der in dem Modul IF-MIB definierten Tabelle `ifTable` dar, durch die eine variable Anzahl von Netzwerkinterfaces eines Netzwerkknotens dargestellt wird. Im Beispiel besitzt der Knoten zwei Interfaces, die beiden `ifEntry`-Elemente. Als Unterelemente sind die Attributswerte der in den Tabellenspalten definierten Unterobjekte angegeben.

SNMP-Objekte werden nur durch ihren Namen spezifiziert, die jeweilige OID spielt in der XML-Adressierung keine Rolle. Um trotz der Vernachlässigung der Module in der Spezifikation des umfassenden XML-Dokumentes Mehrdeutigkeiten durch Namenskollisionen zu vermeiden, werden die Module, in denen die jeweiligen angefragten Objekte definiert sind, in der Gateway-Antwort durch Angabe eines XML-Namespaces kenntlich gemacht. Die beispielhafte Anfrage nach dem zweiten Netzwerkinterface des Knotens test1.snmptest.net durch Angabe des Request-Strings

```
/snmp-data/context[@name="test1.snmptest.net"]/ifEntry[@ifIndex="2"]
```

wird vom Gateway mit der Übermittlung folgenden XML-Dokumentes beantwortet:

```
<?xml version="1.0"?>
<snmp-data xmlns:IF-MIB="http://www.ibr.cs.tu-bs.de/
                         snmp_xml_schema/IF-MIB.xsd>
  <context hostname="test1.snmptest.net">

    <IF-MIB:ifEntry ifIndex="2">
      <IF-MIB:ifDescr>eth0</IF-MIB:ifDescr>
      <IF-MIB:ifType>ethernetCsmacd</IF-MIB:ifType>
      <IF-MIB:ifMtu>1500</IF-MIB:ifMtu>
      [...]
    </IF-MIB:ifEntry>
  </context>
</snmp-data>
```

`IF-MIB` wird als XML-Namespace im entsprechenden XML-Schema definiert. Der angegebene URL ist beispielhaft, wird in der Implementierung jedoch einen tatsächlichen Bezugsort des Schema kennzeichnen. Durch den Namespace-Prefix `IF-MIB` wird `ifEntry` eindeutig, innerhalb einer MIB-Moduldefinition sind die Objektnamen eindeutig.

4.2.2 Behandlung von Leseanforderungen durch das Gateway

Das Gateway selbst wird als Servlet implementiert und in einen Webserver eingebunden. Der Request-String soll in Form eines HTTP-GET-Requests übertragen werden. Die komplette Leseanforderung, die im Folgenden als Request-URL bezeichnet wird, inklusive der Gateway-Adressierung hat also die Form eines URL mit der Syntax:

```
REQUEST-URL := 'http://' GW-DNS-NAME '/' GW-SERVLET '?get=' REQUEST-STRING
```

`GW-DNS-NAME` beinhaltet den Host-Namen des Webservers, `GW-SERVLET` den Namen des Servlets. Der übermittelte Request-String wird durch den festen Variablennamen `get` als Parameter angehängt. Das vom Gateway als Antwort übermittelte XML-Dokument wird im Message-Body der HTTP-Antwort übertragen. Dies ermöglicht die einfache Einbindung des Gateways in webbasierte Clients, die XML-Dokumente durch einen Link in ihre Ausgabe einbinden können.

Das Gateway muss die dem Request-String entsprechenden Attributswerte der Objektinstanzen als XML-Dokument zurückliefern. Dafür muss es durch SNMP-Kommunikation mit dem jeweiligen SNMP-Agenten die Attributswerte bestimmen. Die hierfür nötigen Informationen ermittelt der Gateway aus dem Request-String, wofür dieser in seine Bestandteile aufgespalten wird. Diese sind im einzelnen:

- Informationen über die Netzwerkknoten, dargestellt durch die `context`-Elemente.
- Die Angabe der Objektinstanzen, gegeben durch den kontextspezifischen XPath-Bestandteil.

Die Identifizierung der Verbindungsinformationen für die SNMP-Kommunikation ist leicht durchführbar. Die Ermittlung der abzufragenden OIDs aus dem XPath-Bestandteil ist jedoch nicht trivial, trotz des gegebenen Mappings nach den durch die libsmi erstellten XML-Schemadefinitionen. So sind komplexe XPath-Ausdrücke erstellbar, die Implementierung eines vollständigen Parsers für sämtliche möglichen Ausdrücke ist für diese Arbeit zu umfangreich. Um trotzdem volle Zugriffsflexibilität auf das umfassende, implizite XML-Dokument des Gateways zu gewährleisten, wird das Gateway zuungunsten der Effizienz der zu erfolgenden SNMP-Kommunikation mehr Instanzwerte abfragen, als für die Beantwortung der Client-Anfrage nötig ist. Insbesondere fragt der Gateway zur Beantwortung auf in Tabellen vorgehaltener Objekte die komplette Tabelle des SNMP-Agenten ab. Aus diesen Informationen erstellt der Gateway ein temporäres XML-Dokument, auf welches nun unter Zuhilfenahme eines XML-APIs für XPath-Ausdrücke der komplexe Ausdruck generisch angewendet werden kann. Anhand des Ergebnisses dieser Evaluierung des XPath-Ausdrucks auf das temporäre Dokument konstruiert der Gateway anschließend das an den Client zu übermittelnde XML-Dokument.

4.2.3 Zusammenfassung

Gemäß der getroffenen Spezifikation ergibt sich für die Arbeit der drei beteiligten Komponenten Client, Gateway und SNMP-Agent zur Durchführung einer Client-initiierten Leseanforderung der in Abbildung 4.1 dargestellte Ablauf.

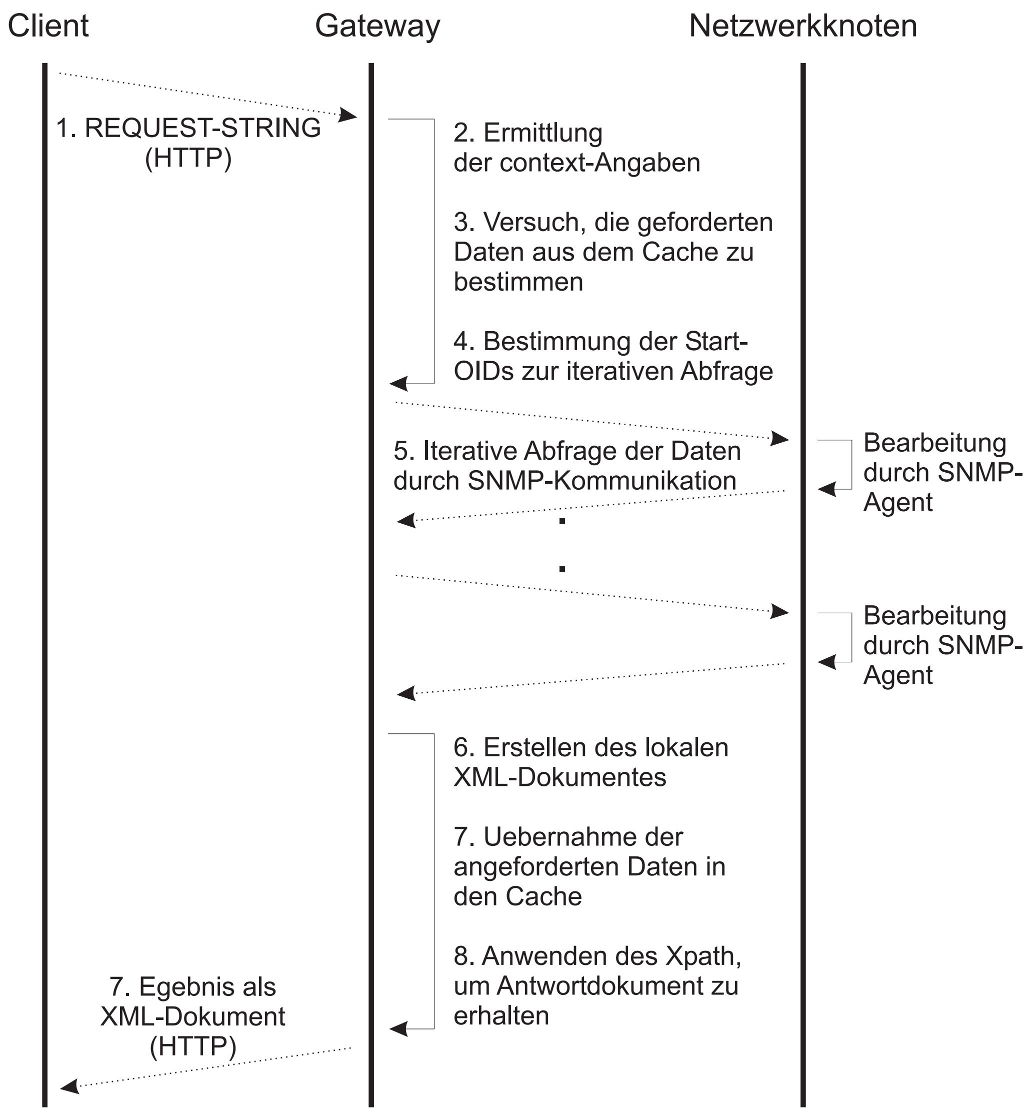

Abbildung 4.1: Ablauf einer Leseanforderung

Sollten die geforderten Daten bereits vor kurzem schon angefordert worden sein, befinden sie sich noch im Cache des Gateways. Falls es die Anfrage erlaubt, Daten des Caches zu verwenden, werden die Schritte `4` bis `7` übersprungen. Durch die Attributsangabe `@caching="no"` kann das Gateway angewiesen werden, Daten des Caches nicht zu verwenden und sämtliche Schritte zur Abfrage der Daten durchzuführen.

4.3 Schreibanforderungen

4.3.1 Syntaktische und semantische Spezifikation

Schreibanforderungen an das Gateway sind durch die XML-Schemata syntaktisch spezifiziert. Ein valides XML-Dokument stellt somit eine gültige Schreibanforderung an das Gateway dar. Das Dokument wird innerhalb eines HTTP-POST-Request als Wert der Variable `set` übertragen. Die Antwort des Gateway besteht in einem HTTP-Status-Code. Falls Fehler während der Bearbeitung der Anfrage aufgetreten sind, ist dieser ergänzt um eine den Fehler beschreibende Nachricht.

Wie in 3.4.4 erwähnt sind durch die Übermittlung eines XML-Dokumentes nur Änderungen bestehender Objektinstanzen möglich. Somit initiiert das Gateway nur für bestimmte Wertangaben innerhalb des XML-Dokumentes entsprechende SNMP-set-Requests. Die entsprechenden Objektinstanzen müssen vorhanden sein und Änderungen erlauben, also in dem jeweiligen MIB-Modul mit Werten `read-write` oder `read-create` der Metaangabe `MAX-ACCESS` definiert sein.

4.3.2 Behandlung von Schreibanforderungen durch das Gateway

Das Gateway ermittelt zunächst die SNMP-Verbindungsinformationen der angegebenen Kontexte aus dem XML-Dokument. Für jeden Kontext werden anschließend Informationen über die angegebenen Objekte aus den Schemata ausgelesen. Da jedes ein Objekt repräsentierende XML-Element mit einem Namespace-Prefix versehen ist, kann hierüber das definierende MIB-Modul eindeutig bestimmt werden.

Aus den ermittelten Informationen muss nun eine Unterscheidung getroffen werden: Für Skalargruppen können die ausgelesenen OIDs direkt benutzt werden. Für Tabellen ist es jedoch zuerst erforderlich, den OID der betroffenen Tabellenzeile zu ermitteln. Dabei ist eine Tabellenzeile eindeutig bestimmt durch SNMP-Zeilenobjekte, die in den XML-Schemata in Attribute des entsprechenden Elements übersetzt sind. Somit ist es zwingend erforderlich, alle Attribute eines Zeilenobjektes in dem XML-Dokument anzugeben, um das entsprechende SNMP-Zeilenobjekt eindeutig bestimmen zu können. Zur Bildung der OID einer bestimmten Zeile werden die Instanzwerte der Zeilenobjekte der Zeile an den OID der Tabelle angehängt. In [16], Section 7.7(1) ist im Einzelnen festgelegt, wie Instanzwerte verschiedener Zeilenobjekttypen in das anzuhängende OID-Postfix umgewandelt werden. Die Anwendung dieser Umwandlungsregeln ist jedoch rein auf Basis der XML-Schemadefinitionen nicht trivial, so dass das Gateway OIDs von Zeilenobjekten nicht zusammensetzt, sondern durch Iteration mittels getnext-Requests ermittelt. In jedem Iterationsschritt werden dabei die Werte der abgefragten Zeilenobjekte mit denen der zu suchenden Zeile verglichen. Diese Vorgehensweise führt zu einem erhöhten Bedarf an SNMP-Kommunikation zur Ausführung eines set-Requests. Dieser Ansatz wurde jedoch trotzdem gewählt, da er unabhängig von SNMP-Objektdefinitionen für alle Tabellen anwendbar ist und bei einfacher Implementierung gut zu verifizieren ist.

Sind die OIDs der zu ändernden Objekte ermittelt, initiiert das Gateway iterativ entsprechende set-Requests an den jeweiligen, durch die Kontextangabe spezifizierten SNMP-

Agenten. Dabei werden set-Requests nur für solche Objekte durchgeführt, die gemäß ihrer Definition änderbar sind. Anschließend werden etwaige im Cache befindliche Daten der betroffenen Objektgruppe gelöscht, um den Cache konsistent zu halten. Hierfür wäre es auch möglich, die Daten zu aktualisieren statt sie zu löschen, unter Berücksichtigung der eher kurzen Vorhaltezeiten im Cache ist eine signifikante Performancesteigerung durch eine Aktualisierung aber nicht zu erwarten. Als letzten Schritt in der Bearbeitung eines set-Request übermittelt das Gateway eine entsprechende Antwort in einem HTTP-Status-Code.

4.3.3 Zusammenfassung

Bild 4.1 gibt zusammenfassend den Ablauf einer Client-initiierten Schreibanforderung wieder. Wie beschrieben, wird Schritt 4 nur für Tabellenobjekte durchgeführt.

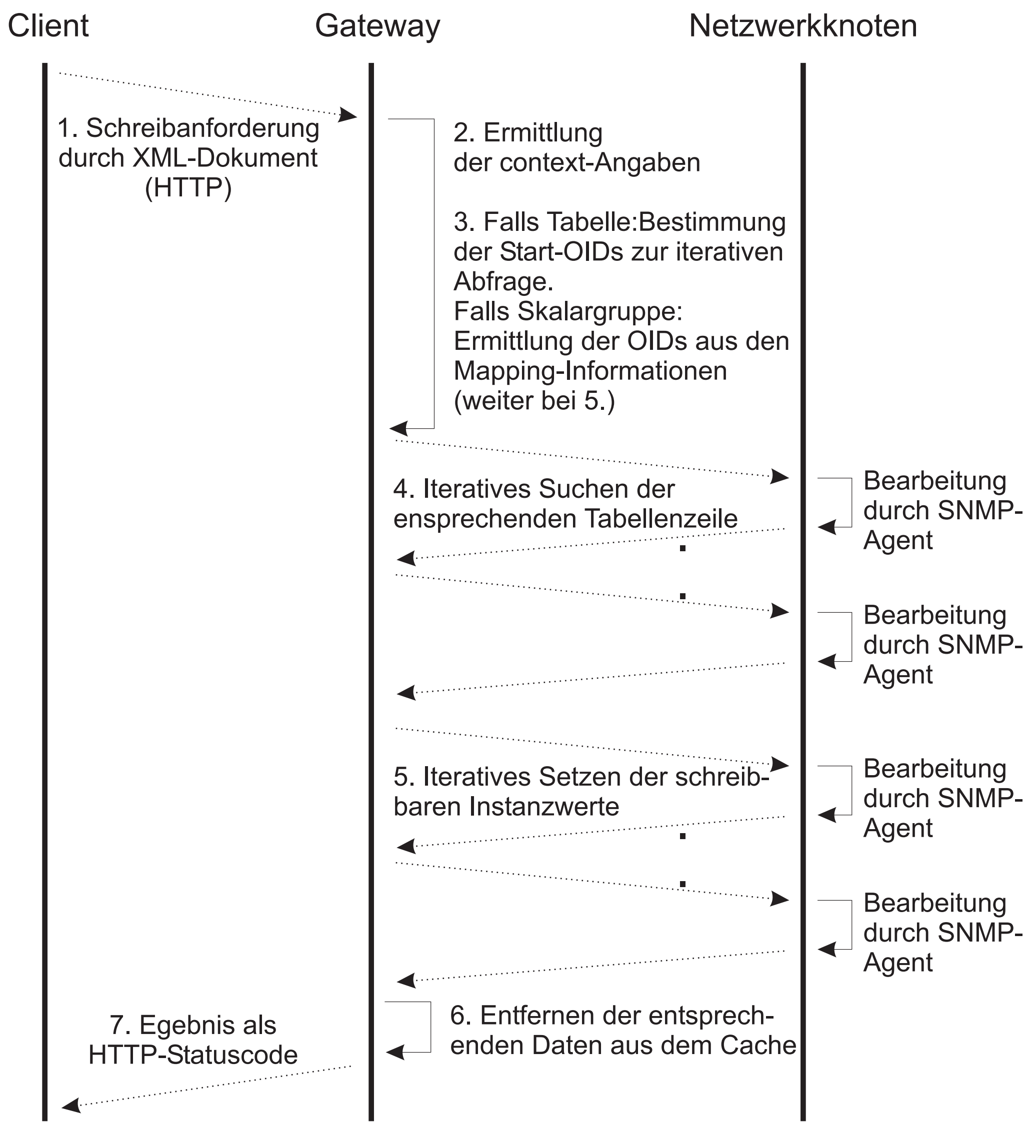

Abbildung 4.2: Ablauf einer Schreibanforderung

4.4 Fehlertoleranz

Neben den in einer verteilten Anwendung prinzipiell immer möglichen Kommunikationsfehlern ergeben sich als Fehlerquelle die Übermittlung inkorrekter Anforderungen an das Gateway. Als Hauptanforderung an das Gateway gilt es hier ein hohes Maß an Robustheit in der Implementierung zu verwirklichen. Kommunikationsfehler sowie fehlerhafte Client-Anforderungen sollen möglichst erkannt und in Verbindung mit einem entsprechenden HTTP-Status-Code natürlichsprachlich angezeigt werden.

Kapitel 5

Implementierung des Gateways

Die erstellte Implementierung des Gateways ist unter [17] verfügbar. Enthalten ist neben dem Quelltext des Gateways ein einfacher Kommandozeilenclient, der die Übermittlung von Anfragen an das Gateway erlaubt. Der Quelltext des Gateways wird unter der *GNU General Public License (GPL)* [18] veröffentlicht, die Benutzung steht somit jedem offen.

5.1 Verwendete Technologien

Die Implementierung des Gateways wurde als HTTP-Servlet in JAVA vorgenommen. Als JAVA-Laufzeitumgebungen wurden das *Software Development Kit (SDK) 1.3.1* und die *Enterprise Edition 1.3.1* benutzt.

Weiterhin wurden zwei komplett in JAVA implementierte *Application Programming Interfaces (API)* benutzt, *dom4j* [19] und *jmgmt* [20]. Dom4j stellt Zugriffsmöglichkeiten auf XML-Dokumente bereit und bietet die Möglichkeit, XPath-Ausdrücke anzuwenden. Jmgmt stellt ein API zur Durchführung von SNMP-Kommunikation dar.

Während der Entwicklung wurde zur Ausführung des Servlets der Webserver *Tomcat* [21] eingesetzt.

5.2 Design

Die Implementierung teilt sich in zwei Arten von Klassen. Zur Bereitstellung globaler Daten und Hilfsfunktionen zur Bearbeitung der Anfragen wurden einerseits Singleton- und Hilfsklassen implementiert. Zur Repräsentation der Daten einer speziellen Anfrage an das Gateway wurde andererseits eine Klassenstruktur realisiert, deren Objekte nach Bearbeitung der Anfrage verworfen werden und somit nur lokale Daten darstellen.

5.2.1 Globale Klassen und Hilfsklassen

Abbildung 5.1 zeigt die implementierten Hilfs- und globalen Klassen.

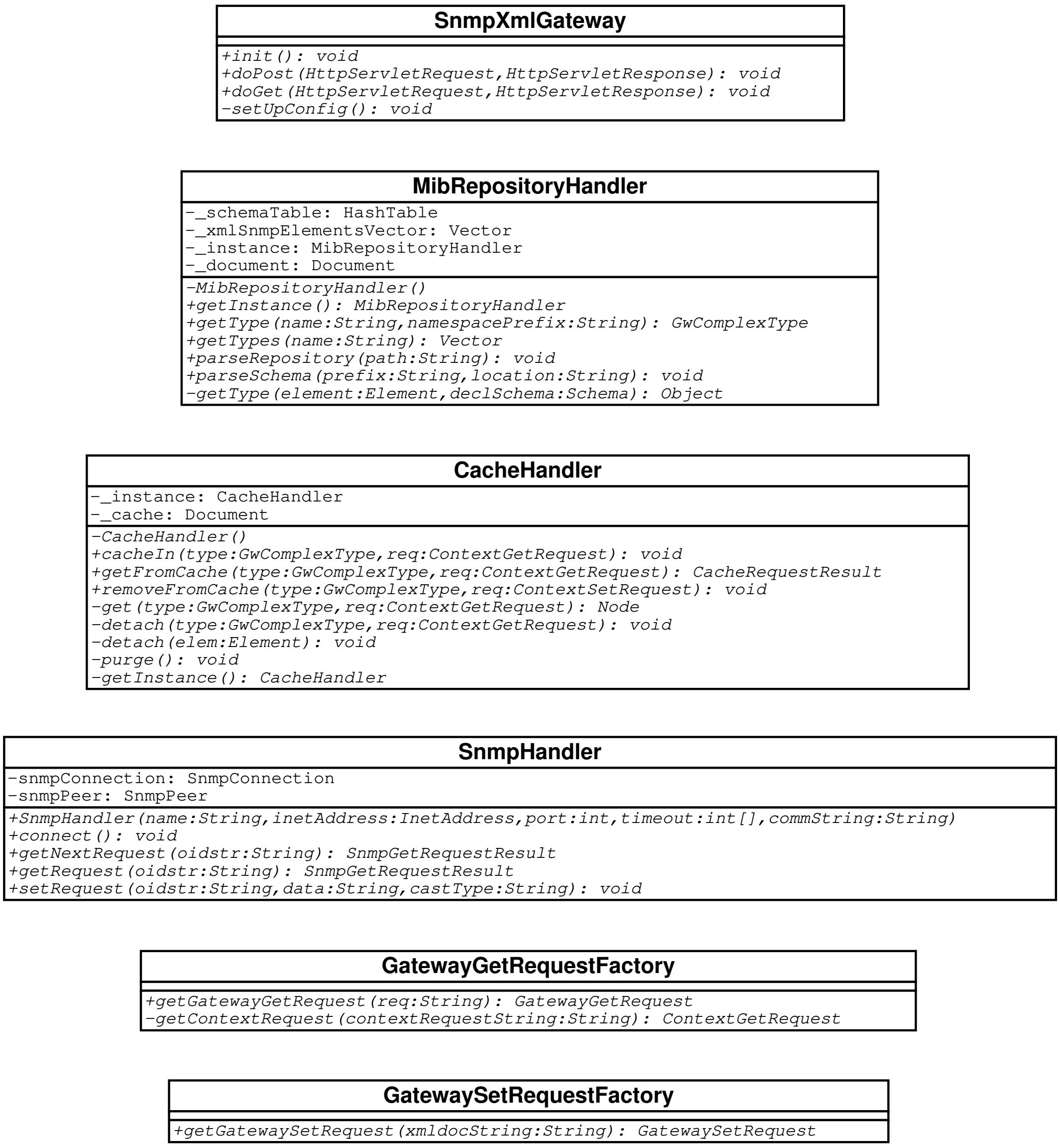

Abbildung 5.1: Globale Klassen und Hilfsklassen

Im einzelnen leisten die Klassen folgendes:

- `SnmpXmlGateway`: Servlet-Klasse, die mit den Methoden `doGet(..)` und `doPost(..)` den Eintrittspunkt der Anwendung für die jeweiligen HTTP-Requests definiert. Die Methode `init()` sorgt dafür, dass Klassen zur Repräsentation globaler Daten bei Aufruf durch den Webserver initialisiert werden. Insbesondere wird an dieser Stelle die Singleton-Klasse `MibRepositoryHandler` angewiesen, die XML-Schemata einzulesen.

- `MibRepositoryHandler`: Singleton-Klasse zur Repräsentation der XML-Schemata zur Übersetzung der SNMP- und XML-Adressierungen. Die Methoden `getType(..)` und `getTypes(..)` erstellen gemäß des übergebenen Objektnamens die Datenstruktur für eine spezifische Anfrage an das Gateway.

- `CacheHandler`: Stellt als Singleton-Klasse den Zugriff auf den globalen Cache bereit. über die Methoden `cacheIn(..)`, `getFromCache(..)` und `removeFromCache(..)` können während der Bearbeitung einer Anfrage Daten in den Cache übertragen, von dort ausgelesen oder aus dem Cache entfernt werden. Zusätzlich wird eine innere Klasse implementiert, die als eigenständiger Thread in bestimmten Zeitintervallen veraltete Daten aus dem Cache löscht, um den Speicherverbrauch des Cache zu verringern.

- `SnmpHandler`: Hilfsklasse für die Durchführung von SNMP-Kommunikation. Instanzen dieser Klasse werden während der Bearbeitung einer Anfrage als lokale Objekte erzeugt und stellen Methoden zur Ausführung von SNMP-Requests mit dem im Konstruktor spezifizierten Agenten bereit.

- `GatewayGetRequestFactory`: Factory-Klasse zur Erzeugung eines `GatewayGetRequest`-Objektes anhand des übergebenen Requests-Strings.

- `GatewaySetRequestFactory`: Factory-Klasse zur Erzeugung eines `GatewaySetRequest`-Objektes anhand des übergebenen XML-Dokumentes.

5.2.2 Klassenstruktur zur Repräsentation von Anfragen

Abbildung 5.2 zeigt die implementierte Klassenstruktur zur Anfragerepräsentation. Aus Übersichtsgründen wird auf eine vollständige Darstellung der Klassen verzichtet, insbesondere sind einfache Accessor-Methoden privater Instanzvariablen nicht dargestellt.

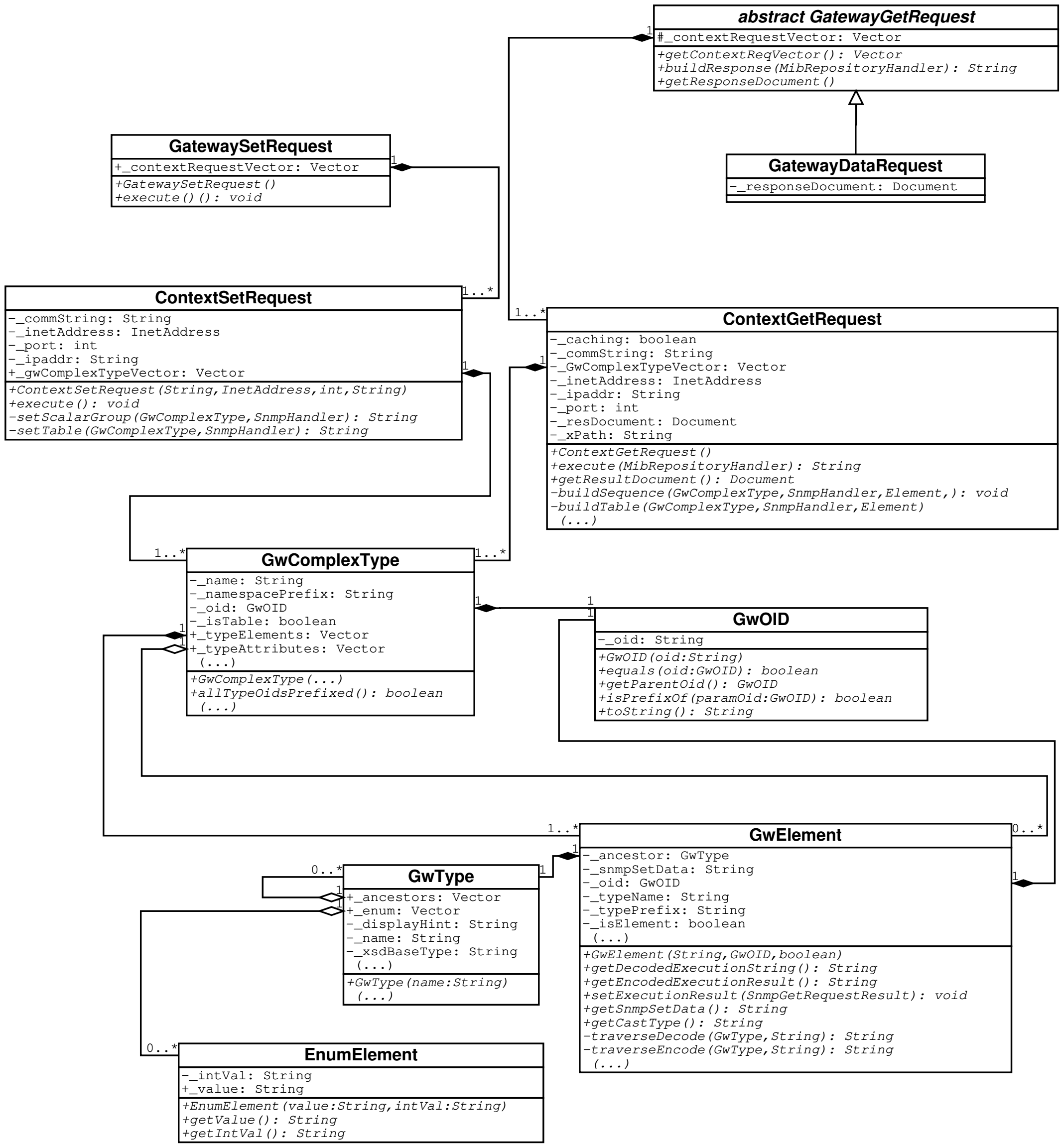

Abbildung 5.2: Klassenstruktur zur Anfragerepräsentation

Die dargestellte Klassenstruktur repräsentiert an das Gateway gestellte Anforderungen zum Auslesen oder zur Modifikation in jeweiligen Kontexten vorgehaltener Daten. Im einzelnen stellen die Klassen folgende Repräsentationen dar:

- `GatewayDataRequest`: Stellt eine Leseanforderung dar, bestehend aus mehreren kontextspezifischen Leseanforderungen.

- `ContextGetRequest`: Eine einzelne kontextspezifische Leseanforderung, beinhaltet die Verbindungsdaten der SNMP-Kommunikation und die angeforderten Tabellen beziehungsweise Skalargruppen.

- `GwComplexType`: Repräsentiert die aus dem <`xsd:ComplexType`>-Element des entsprechenden XML-Schema ausgelesenen Daten einer Tabelle oder Skalargruppe.

- `GwElement`: Daten eines <`xsd:Element`>-Elements, zugehörig zu einem bestimmten <`xsd:ComplexType`>-Element. Repräsentiert somit einzelne SNMP-Objektinstanzen.
- `GwType`: Dient zum Aufbau der in den XML-Schemata vorgegebenen Typhierarchie.
- `EnumElement`: Repräsentiert einen Enumerationswert innerhalb eines <`xsd:enumeration`>-Elements.
- `GatewaySetRequest`: Stellt eine Anforderung zur Modifikation dar, bestehend aus mehreren kontextspezifischen Schreibanforderungen.
- `ContextSetRequest`: Eine einzelne kontextspezifische Schreibanforderung, beinhaltet die Verbindungsdaten der SNMP-Kommunikation und die zu modifizierenden Tabellen beziehungsweise Skalargruppen.

5.3 Ausführung gestellter Anforderungen

Die Sequenzdiagramme 5.3 und 5.4 verdeutlichen qualitativ den Programmablauf zur Ausführung der an das Gateway gestellten Requests. Zur Ausführung der SNMP-Kommunikation werden innerhalb eines `ContextSetRequest` beziehungsweise eines `ContextGetRequest` weitere Schleifen durchlaufen, auf deren Darstellung zugunsten einer besseren Lesbarkeit der Diagramme verzichtet wurde.

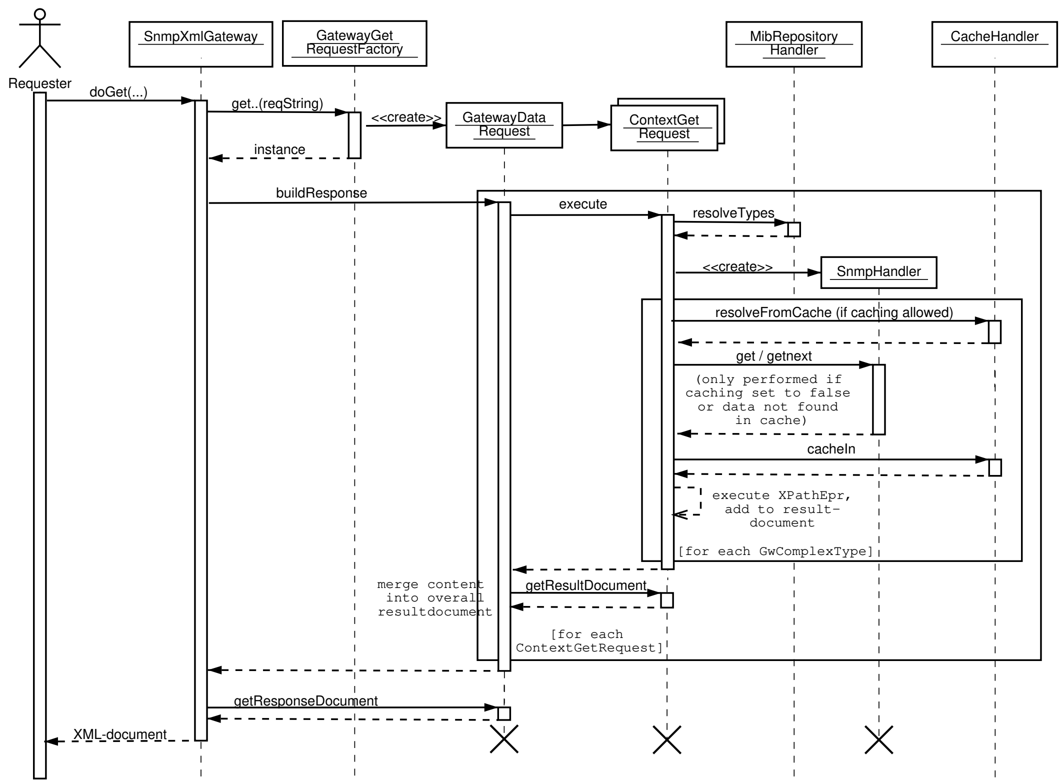

Abbildung 5.3: Sequenzdiagramm: get-Requests

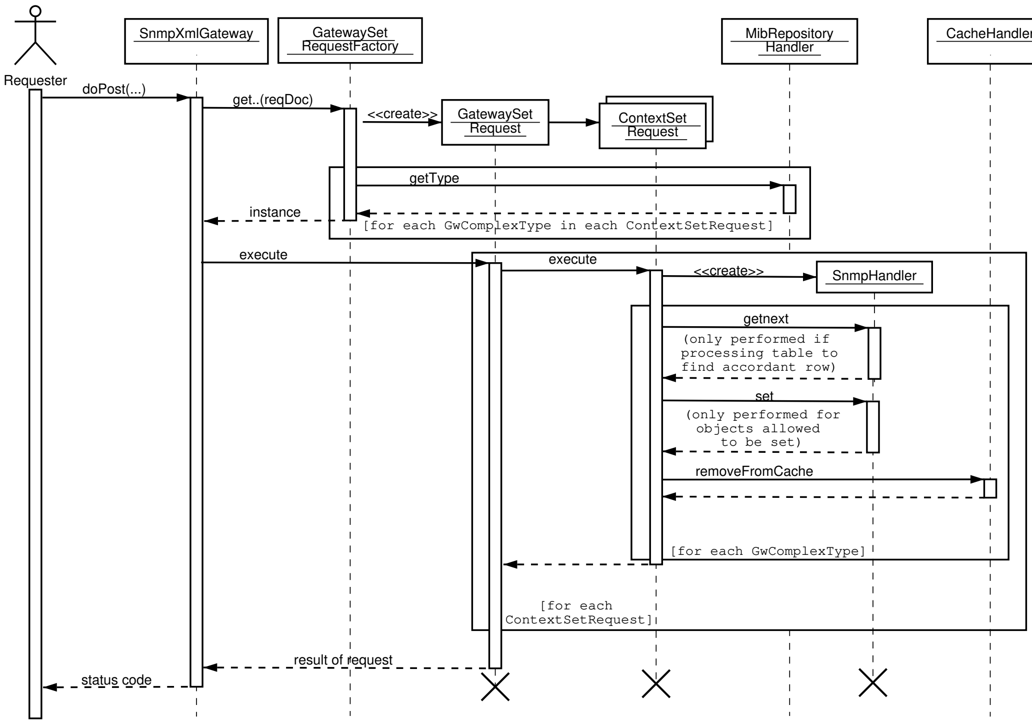

Abbildung 5.4: Sequenzdiagramm: set-Requests

5.4 Konfiguration des Gateways

5.4.1 deployment descriptor

JAVA-Servlets werden üblicherweise durch einen *deployment descriptor* konfiguriert. Dieser besteht in einer XML-Datei, welche im Servletverzeichnis gespeichert ist und standardmäßig den Namen *web.xml* trägt. Durch diese Datei können einem Servlet Startparameter übergeben werden.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>SNMP-XML Gateway Application
            </web-resource-name>
            <url-pattern>/servlet/SnmpXmlGateway</url-pattern>
        </web-resource-collection>
    </security-constraint>
```

```
    <context-param>
        <param-name>repositoryURL</param-name>
        <param-value>http://www.ibr.cs.tu-bs.de/arbeiten/snmp-xml-gw/xsd/
                     container.xsd
        </param-value>
        <description>URL of toplevel container-schema used for mapping
        </description>
    </context-param>
    <context-param>
        <param-name>cachingTimeSecs</param-name>
        <param-value>40</param-value>
        <description>determines how long (in secs) from agents retrieved
                     data will be handled as up-to-date
        </description>
    </context-param>
    <context-param>
        <param-name>cachePurgeIntervalSecs</param-name>
        <param-value>60</param-value>
        <description>determines the interval (in secs) of cache-cleanup
        </description>
    </context-param>
    <context-param>
        <param-name>setAllowed</param-name>
        <param-value>yes</param-value>
        <description>(yes|no) determines if the gateway will perform
                     set-operations.
        </description>
    </context-param>
</web-app>
```

Durch <`web-resource-collection`> kann dem Servlet ein Name gegeben werden und eine Abkürzung des URL eingerichtet werden, diese Angaben sind nicht zwingend für den Betrieb des Gateway. Über die <`context-param`>-Elemente werden dem Servlet Startparameter mitgeteilt, die hier vorgenommenen Angaben beeinflussen die Arbeitsweise des Gateways.

- `repositoryURL`: Der URL des verwendeten XML-Schema als Grundlage der Adressierungsübersetzung.
- `cachingTimeSecs`: Angabe in Sekunden, wie lange im Cache befindliche Daten als aktuell angesehen werden.
- `cachePurgeIntervalSecs`: Angabe in Sekunden, in welchem Intervall der Thread zum Entfernen alter Einträge innerhalb der Klasse `CacheHandler` aktiv wird.
- `setAllowed`: Bestimmt, ob das Gateway Schreibanforderungen ausführt. Mögliche Werte hierfür sind `yes` und `no`.

5.4.2 XML-Schema Repository

Das Gateway benötigt zur Übersetzung zwischen der SNMP- und XML-Adressierung der SNMP-Objekte ein Modulrepository in Form von XML-Schemata. Diese werden durch den XSD-Treiber des Programms *smidump* unter Nutzung der *libsmi* [12] erzeugt. Dabei stellt ein *container-Schema* die oberste Zugriffsebene auf die Schemata durch das Gateway dar. Antworten auf Leseanforderungen und die XML-Dokumente der Schreibanforderungen müssen bezüglich dieses container-Schema valide sein. Damit das Gateway auf die Schemata zugreifen kann, müssen diese über HTTP zugänglich gemacht gemacht werden und der URL des container-Schema als Wert des Parameters `repositoryURL` im deployment descriptor des Servlets eingetragen werden.

5.5 Nutzung des Clients

Neben der Implementierung des Gateways wurde ein einfacher kommandozeilenbasierter Client zur Übermittlung von Lese- und Schreibanforderungen erstellt. Die Clientanwendung wurde ebenfalls in JAVA als Klasse *GatewayClient* erstellt und benötigt zur Ausführung das API dom4j. Das folgende Beispiel verdeutlicht die Benutzung des Clients zur Übermittlung einer Leseanforderung an das Gateway.

```
$ java -cp <PATH> GatewayClient http://www.ibr.cs.tu-bs.de/SnmpXmlGateway
  get '/snmp-data/context[@hostname="testhost"]/tcp'
```

Das erste Argument teilt der Anwendung den URL des Gateways mit, das zweite Argument `get` weist den Client an, eine Leseanfrage zu initiieren. Das dritte Argument schließlich stellt den Request-String dar, der an das Gateway übermittelt werden soll. In diesem Fall wird die Skalargruppe `tcp` abgefragt. Es ist sicherzustellen, dass der Klassenpfad <`PATH`> das API dom4j enthält, falls dieses API nicht im systemweiten JAVA-Klassenpfad enthalten ist.

Zur Benutzung des Clients zur Übermittlung von Schreibanforderungen wird vorausgesetzt, dass eine gespeicherte Datei das zu übermittelnde XML-Dokument enthält. Folgendes Beispiel zeigt den Aufruf des Clients zur Übermittlung einer Schreibanforderung:

```
$ java -cp <PATH> GatewayClient http://www.ibr.cs.tu-bs.de/SnmpXmlGateway
  set document.xml
```

Während das erste Argument wiederum den URL des Gateways festlegt, weist das zweite Argument `set` den Client an, eine Schreibanforderung zu übermitteln. Das entsprechende XML-Dokument ist dabei in der Datei `document.xml` gespeichert.

5.6 Antworten des Gateways

Wie in 3.4.6 und 4.4 gefordert, übermittelt das Gateway in der HTTP-Antwort einen HTTP-Statuscode zusammen mit einer natürlichsprachlichen Nachricht, um eventuell aufgetretene Fehler anzuzeigen. Die Tabellen 5.1 und 5.2 zeigen die möglichen HTTP-Statuscodes der Antworten.

Code	Erläuterung
200	Fehlerfreie Bearbeitung der Anfrage, Antwortdokument übermittelt
400	Syntaktischer Fehler in Anfrage, weitere Erklärung in Nachricht
500	Interner Fehler während der Bearbeitung oder falsche Konfiguration im deployment descriptor. Weitere Erklärung in Nachricht

Tabelle 5.1: Übermittelte HTTP-Statuscodes bei Leseanforderungen

Code	Erläuterung
200	Fehlerfreie Bearbeitung der Anfrage, allen schreibbaren Objektinstanzen die übermittelten Werte zugewiesen
400	Syntaktischer Fehler in Anfrage oder Fehler während der SNMP-Kommunikation. Weitere Erklärung in Nachricht
405	Gemäß deployment descriptor sind Schreibanforderungen nicht erlaubt
500	Interner Fehler während der Bearbeitung oder falsche Konfiguration im deployment descriptor. Weitere Erklärung in Nachricht

Tabelle 5.2: Übermittelte HTTP-Statuscodes bei Schreibanforderungen

5.7 Beschränkungen der Implementierung

Die Haupteinschränkung der hier vorgestellten Implementierung bezüglich der vorgenommenen Spezifikation eines Snmp-Xml-Gateways besteht darin, SNMP-Traps nicht verarbeiten zu können. Das für die SNMP-Kommunikation verwendete API jmgmt stellt keine Möglichkeit bereit, Agenten-initiierte Kommunikation zu empfangen. Aus diesem Grund bietet die Implementierung keine Funktionalität als trap-relay. Das zur Abfrage vorgehaltener Trap-Daten gedachte context-Attribut `@trapdata` wird in dieser Implementierung ignoriert.

Während der Implementierung des Gateways wurde im Zug der Weiterentwicklung von smidump ein neues Konstrukt in den XML-Schemata verwirklicht, welches die Repräsentation von SNMP-Tabellen innerhalb anderer SNMP-Tabellen ermöglicht. Diese Erweiterung wird in der Implementierung des Gateways nicht berücksichtigt.

Eine weitere während der Implementierung vorgenommene Änderung der XML-Schemata betrifft die Darstellung von SNMP-Bitsequenzen. In solchen Sequenzen dient ein einzelnes

Bit als Wahrheitswert für ein durch den Index des Bits bestimmtes Attribut. Ein Beispiel einer solchen Sequenz ist das SNMP-Objekt `schedMinute` eines Tabelleneintrages `schedEntry` der Tabelle `schedTable`, die Definition dieser Objekte wird in der DISMAN-SCHEDULE-MIB [22] vorgenommen. `schedMinute` besteht dabei aus einer 60 Bit langen Sequenz, ist das Bit an Position i gesetzt, wird der durch den Tabelleneintrag geplante Task an zutreffenden Stunden in der i-ten Minute ausgeführt. Die entsprechende Objektdefinition innerhalb des XML-Schema wurde während der Implementierung geändert, das Gateway verarbeitet nur die alte Objektdefinition zu einer korrekten Darstellung.

Zudem scheint das für die SNMP-Kommunikation verwendete API jmgmt nicht in der Lage zu sein, SNMP-set-requests für solche als Bitsequenzen definierte SNMP-Objekte korrekt durchzuführen. Modifikationen solcher Objekte werden aus diesem Grund von der Implementierung des Gateways nicht unterstützt.

Bei einem Betrieb des Servlets durch einen Tomcat-Webserver mit einer niedrigeren Version als 4.0.1 werden nicht die gatewayspezifischen Nachrichten in der Antwort übermittelt, sondern der Standardtext der entsprechenden HTTP-Statuscodes. Ab Version 4.0.1 werden hingegen korrekt die vom Gateway eingebrachten Nachrichten übermittelt. Grund hierfür ist eine geänderte Behandlung der JAVA-Methode `sendError(..)` eines `HTTPServlet`, welche zur Übermittlung des HTTP-Statuscodes genutzt wird.

Kapitel 6

Evaluierung

6.1 Betriebsumgebung

Während der Implementierung und des Performancetests wurde das Gateway unter der Tomcat Servlet Engine in der Version 4.0.1 auf einem Linux-System betrieben. Als Hardware wurde ein PC mit einem AMD Athlon Prozessor mit 1.2 GHz und 512 MB RAM eingesetzt. Während der Performanceuntersuchungen wurden keine weiteren Prozesse mit großer Prozessor- und Speicherlast ausgeführt.

Zudem wurde eine zweite Installation des Gateways auf dem Webserver des Instituts für Betriebssysteme und Rechnerverbund vorgenommen. Auf dieser Installation war es möglich, verschiedene SNMP-fähige Netzwerkknoten des Institutsnetzwerkes für Korrektheitstests der Implementierung anzusprechen.

Das Repository bestand für beide Installationen aus einem container-Schema, welches Zugriff auf Objekte der folgender MIB-Module ermöglicht [23].

- SNMPv2-MIB
- SNMP-FRAMEWORK-MIB
- HOST-RESOURCES-MIB
- ENTITY-MIB
- DISMAN-SCRIPT-MIB
- DISMAN-SCHEDULE-MIB
- IF-MIB
- TCP-MIB

6.2 Test

Korrektheitstests erfolgten während der Implementierung des Gateways. Die Klassen der Datenstruktur zur Anfragerepräsentation besitzen eine Debug-Methode `dump(..)`, mit deren Hilfe nach Änderungen im Quellcode eine textuelle Darstellung der repräsentierten Daten durch das Gateway ausgegeben werden kann. Anhand dieser Darstellung wurden implementierungsbegleitend Korrektheitstests für unterschiedliche Anfragen vorgenommen.

6.3 Performance

Wie in 3.4.7 angemerkt, wurde nicht erwartet, bei der Beantwortung der Anfragen durch das Gateway die Ausführungsgeschwindigkeit direkter SNMP-Kommunikation zu erreichen. Interessant ist es jedoch zu untersuchen, in welchem Maße die Bearbeitungsdauer höher ist und inwiefern das Caching die Beantwortungsdauer beeinflusst.

6.3.1 Performanceuntersuchung

Für die Performanceuntersuchung wurden die Skalargruppen `tcp`, `system` und die Tabellen `ifTable`, `hrFSTable` abgefragt. Beide Tabellen besaßen dabei fünf Einträge. Die Bearbeitungszeit wurde unter Verwendung des Unix-Programms *time* ermittelt.

Zugriff	tcp	system	hrFSTable	ifTable
snmpwalk	0,17	0,13	0,16	0,23
Gateway, ohne Cache	1,2	1,1	1,8	6,9
Gateway, mit Cache	1,1	1,1	1,6	6,4

Tabelle 6.1: Ca.-Werte der Bearbeitungsdauer in Sekunden

6.3.2 Ergebnisdiskussion der Performanceuntersuchung

Wie erwartet ist die Bearbeitungszeit der Anfragen unter Verwendung des Gateways höher. Beachtenswert ist der geringe Einfluss des Caches und die im Vergleich der anderen Abfragen nochmals höhere Bearbeitungszeit bei Abfrage der `ifTable`. Beides lässt sich jedoch durch die Arbeitsweise des Gateways erklären: Für eine übermittelte Anfrage wird die anfragelokale Objektstruktur zur Repräsentation der Daten während ihrer Bearbeitung erstellt. Dies geschieht durch den Aufruf der Methode `parseTypes(..)` der Klasse `MibRepositoryHandler` (Abbildungen 5.1, 5.3). Diese Erstellung macht den Hauptteil der Bearbeitungszeit aus und erfolgt unabhängig davon, ob die Daten später aus dem Cache gelesen oder per SNMP-Kommunikation abgefragt werden. Durch den Cache wird zwar die SNMP-Kommunikation und somit Netzwerkverkehr eingespart, die Bearbeitungszeit jedoch nur um den geringen Zeitanteil dieser Kommunikation verringert. Dies lässt sich auch aus der Tabelle ablesen, die Differenz der benötigten Zeit zur Bearbeitung unter

Ausschluss beziehungsweise Verwendung des Caches entspricht größenordnungsmäßig der Zeit, die zur Abfrage der Daten per `snmpwalk` benötigt wird.

Die nochmals höhere Bearbeitungszeit zur Abfrage der `ifTable` hat ihre Ursache ebenfalls in der Erstellung der anfragelokalen Objektstruktur zur Bearbeitungszeit. Die Definition eines Zeilenobjektes `ifEntry` enthält ein Objekt `ifType` vom Typ `IANAifType`, welches in dem MIB-Modul `IANAifType-MIB` [24] definiert ist. `IANAifType` besteht aus 219 Enumerations-Objekten, die Namen von Interface-Typen Integer-Werten zuordnen. Bei Erstellung der lokalen Objektstruktur erfolgt ein Parsen dieser Objektdefinition, welches aufgrund der hohen Anzahl der Enumerations-Werte vergleichsweise viel Zeit erfordert und die Bearbeitungszeit zur Abfrage der `ifTable` in starkem Maße erhöht.

6.3.3 Diskussion performanceerhöhender Maßnahmen

Die Performanceuntersuchung zeigt, dass eine Parallelisierung der SNMP-Kommunikation durch mehrere Threads wie in 3.4.7 vorgestellt die Bearbeitungszeit nicht wesentlich verringern würde. Die SNMP-Kommunikation hat an der Gesamtzeit zur Bearbeitung einer Anfrage durch das Gateway nur einen vergleichsweise geringen Anteil.

Vielmehr gilt es, die Erstellung der anfragelokalen Datenstruktur zu beschleunigen. Eine Verbesserung kann hier erzielt werden, indem bereits während der Initialisierung des Servlet sämtliche im container-Schema angegebenen Objekte in eine globale Datenstruktur überführt werden und lokale Teilkopien dieser Struktur für bestimmte Anfragen erstellt werden. Somit entfällt das aufwendige Parsen der XML-Schemata während der Bearbeitung einer Anfrage. Dieser Ansatz birgt jedoch einen Nachteil. Bei großer Anzahl der im container-Schema angegebenen Objekte wird für eine solche globale Struktur ein hohes Maß an Speicherplatz benötigt. Zwar kann der benötigte Speicher reduziert werden, indem für jede `SimpleType`-Definition nur ein entsprechendes `GwType`-Objekt erstellt wird, auf welches dann mehrfach referenziert werden muss. Jedoch ist selbst bei Minimierung des verwendeten Speicherplatzes auf diese Weise fraglich, ob durch diesen Ansatz ein großer Teil der ca. 100000 verschiedenen MIB-Objekte darstellbar ist.

Kapitel 7

Bewertung und Ausblick

Die erfolgte Implementierung zeigt, dass ein XML-basiertes Netzwerkmanagement praktisch nutzbar ist. Nach kurzer Einarbeitungszeit in die durch XML-Schemata gegebenen Objektdefinitionen und in die Erstellung von XPath-Ausdrücken ermöglicht das SNMP-XML-Gateway eine intuitive Abfrage und Modifikation managementrelevanter Daten ohne tiefgreifende Kenntnis des zugrundeliegenden SNMP-Frameworks. Die bestehenden Beschränkungen der Implementierung sind nicht prinzipieller Natur: Unter Verwendung eines anderen JAVA-APIs zur SNMP-Kommunikation können auch Informationen der SNMP-traps durch das Gateway zur Verfügung gestellt werden. Die Spezifikation zur Abfrage solcher Daten wurde in 4.2.1 bereits vorgenommen.

Die Hauptbeschränkung der Spezifikation selbst ist die fehlende Möglichkeit, Tabellenobjekte erzeugen beziehungsweise löschen zu können. Das SNMP-Framework bietet hierfür keinen allgemeingültigen Ansatz. Aus diesem Grund ist es schwierig, hierfür eine generisch handhabbare Lösung auf XML-Basis zu finden, es müsste eine spezielle Semantik losgelöst von bestehenden MIB-Definitionen definiert werden, die erlaubt, solche Anfragen auf XML-Basis auszudrücken. Da die Möglichkeiten der XML-Technologien für das Netzwerkmanagement erst seit vergleichsweise kurzer Zeit erforscht werden, ist es nachvollziehbar, dass Lösungen hierfür eher Vorschlagscharakter haben als eine genaue Definition zu treffen. Es ist erst einmal zu erforschen, welche Ansätze sich gut eignen, änderbare Tabellen auf XML-Basis darzustellen.

Die in 6.3.2 herausgestellten Performancenachteile der Implementierung im Vergleich zu direkter SNMP-Kommunikation scheinen verringerbar, ein möglicher Ansatz zur Performanceverbesserung wurde in 6.3.3 vorgestellt. In vielen Fällen erlaubt jedoch bereits die hier vorgestellte Implementierung des Gateways eine zügige Abfrage und Modifikation bestehender Objektinstanzen.

Ein weiterer Schritt zur Verbesserung des Gateways wäre die Nutzung von SNMPv3 zur Kommunikation mit Agenten, die dieses Protokoll unterstützen. Das Sicherheitskonzept SNMPv1 und SNMPv2 implementierender SNMP-Agenten beruht in vielen Fällen auf dem Community-String als einer Art Passwort in Verbindung mit der IP-Adresse des Netzwerkknotens, von dem aus eine Anfrage initiiert wird. Das SNMP-XML-Gateway hebelt diese Konzept in dem Sinne aus, dass Anfragen unterschiedlichster Clients über einen physischen Netzwerkknoten, dem Rechner der als Webserver das Gateway ausführt, gebündelt werden.

Somit ist eine Erkennung, von welchem Netzwerkknoten eine Anfrage gestellt wird und eine entsprechende Differenzierung durch den SNMP-Agenten nicht mehr möglich. Der Community-String alleine ist in diesem Szenario für die Zugriffssteuerung zu benutzen, dies erfordert eine durchdachte Konfiguration über das Gateway ansprechbarer SNMP-Agenten. Prinzipiell wird das Management unter Benutzung des Gateways jedoch nicht unsicherer, ein Angreifer könnte bei Kenntnis eines Community-Strings zur Schreibautorisierung auch mit direkter SNMP-Kommunikation und *IP-Spoofing* von nicht autorisierten Netzwerkknoten Modifikationen der Daten vornehmen. Diese prinzipielle Unsicherheit hat zur Definition des nutzerbasierten Sicherheitskonzeptes in SNMPv3 geführt, eine Integration dieser Protokollversion in das Gateway wäre wünschenswert.

Anstelle des einfachen Testclients sind weitaus komplexere Anwendungen denkbar, die auf Basis der übermittelten XML-Dokumente vielfältige Funktionen zum Netzwerkmanagement realisieren. Die Daten könnten einem Benutzer in verschiedenen Darstellungen präsentiert werden, komplexe Konfigurationsänderungen als XML-Dokumente gespeichert werden. Zur Realisierung solcher Clients ist keine detaillierte Kenntnis des SNMP-Frameworks mehr erforderlich.

Literaturverzeichnis

[1] Juergen Schoenwaelder, Aiko Pras, Jean-Philippe Martin-Flatin: *On the Future of Internet Management Technologies (not yet published)*, 2002.

[2] Frank Strauss, Torsten Klie: *Towards XML oriented Internet Management. submitted to appear in 8th IFIP/IEEE International Symposium on Integrated Network Management*, March 2003.

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

[4] Rose, Marshall T.: *The Simple Book: An Introduction to Networking Management.* Prentice Hall, Second Edition , 1996.

[5] McCloghrie, Keith: *RFC 2012: SNMPv2 Management Information Base for the Transmission Control Protocol using SMIv2 (Status: PROPOSED STANDARD)*, November 1996.

[6] McCloghrie, Keith: *RFC 1905: Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2) (Status: DRAFT STANDARD)*, January 1996.

[7] Levi, D.: *RFC 2573: SNMP Applications (Status: DRAFT STANDARD)*, April 1999.

[8] *Internet Drafts der IRTF Network Management Research Group (NRMG): <http://www.ibr.cs.tu-bs.de/projects/nmrg/>.*

[9] Strauss, Frank: *SMIng - A new Structure of Management Information. TU Braunschweig*, 1999.

[10] *TCP-MIB Schema: <http://www.ibr.cs.tu-bs.de/arbeiten/snmp-xml-gw/xsd/TCP-MIB.xsd>.*

[11] D. Box, et al.: *Simple Object Access Protocol (SOAP) 1.1, W3C Note: <http://www.w3.org/TR/SOAP/>*, May 2000.

[12] *libsmi: <http://www.ibr.cs.tu-bs.de/projects/libsmi/>.*

[13] J. Clark, S. DeRose: *XML Path Language (XPath) Version 1.0, W3C Recommendation: <http://www.w3.org/TR/XPath/>*, November 1999.

[14] S. Boag, et al.: *XQuery 1.0: An XML Query Language, W3C Working Draft: <http://www.w3.org/TR/XQuery/>*, November 2002.

[15] Clark, James: *XSL Transformations (XSLT) Version 1.0, W3C Recommendation: <http://www.w3.org/TR/XSLT/>*, November 1999.

[16] K. McCloghrie, D. Perkins, J. Schoenwaelder: *RFC 2578: Structure of Management Information Version 2 (SMIv2) (Status: STANDARD)*, April 1999.

[17] *Implementierung des Gateways: <http://www.ibr.cs.tu-bs.de/arbeiten/snmp-xml-gw/sxg-1.0.tar.gz>.*

[18] *GNU General Public License: <http://www.gnu.org/copyleft/gpl.html/>.*

[19] *dom4j JAVA XML-API: <http://www.dom4j.org/>.*

[20] *Jmgmt JAVA SNMP-API: <http://i31www.ira.uka.de/ sd/mgmt/jmgmt/>.*

[21] *Tomcat Webserver: <http://jakarta.apache.org/tomcat/>.*

[22] D. Levi, J. Schoenwaelder: *RFC 3231: Definitions of Managed Objects for Scheduling Management Operations (Status: PROPOSED STANDARD)*, January 2002.

[23] *Container-Schema der Gateway-Evaluierung: <http://www.ibr.cs.tu-bs.de/arbeiten/snmp-xml-gw/xsd/container.xsd>.*

[24] *XML-Schema IANAifType-MIB.xsd: <http://www.ibr.cs.tu-bs.de/arbeiten/snmp-xml-gw/xsd/IANAifType-MIB.xsd>.*