TOWARDS XML ORIENTED INTERNET MANAGEMENT

Frank Strauß
Technical University Braunschweig, Germany
strauss@ibr.cs.tu-bs.de

Torsten Klie
Technical University Braunschweig, Germany
tklie@ibr.cs.tu-bs.de

Abstract: Internet Management is based on IETF specifications that have been developed and used during the past 14 years: There are multiple versions and options of the management protocol (SNMP), two versions of the language for specifying the structure of management information (SMI), and more than 160 Standard MIB modules. This altogether represents the most widely deployed management technology these days.

However, the SNMP centered management framework has some drawbacks especially related to efficient configuration management and efficient application development processes. Today to many people the whole family of XML technologies seems to promise a fancy way out of this trouble.

This paper presents an approach to automatically convert SMI MIB definitions and according SNMP agent data into XML Schema definitions and appropriate valid XML documents. Instead of a plain mapping of MIB trees to nested XML elements, we tried to adapt the XML philosophy of a rather flat element containment hierarchy and appropriate XML Schema type definitions. We also present an approach towards an SNMP/XML gateway and our thoughts on management applications based on XML technologies.

Keywords: Internet Management, Configuration Management, Information Model, Data Model, SNMP, SMI, MIB, XML, XML Schema, XSLT, SNMP/XML Gateway.

1. Introduction

Since 1988, when the first specifications of SNMP and the SMI and the first definitions of managed objects have been published by the IETF [1, 2, 3], the SNMP framework evolved dramatically in a way that problems have been identified and fixed, many details in specifications have been clarified and — probably the hardest part of the work — more than 160 MIB modules have been designed, refined and standardized by various IETF working groups. See the latest issue of the *Simple Times* [4] for an overview of all related IETF Standards documents.

The result of this solid standardization work is a fundamental technology upon which many hardware vendors, software manufacturers, network operators, and administrators have built software systems to manage a wide range of computer networks. These implementations also represent huge investments.

However, the SNMP framework has some drawbacks that cannot be solved without massive modifications or completely new technologies [5, 6]. Some people expect that at least two of the major problems can be solved by applying XML based technologies

to network management tasks: (a) efficient and atomic transfer of configuration data is problematic with SNMP, and (b) usual development processes of SNMP agents and other applications are quite slow and expensive, since the abstraction layer of MIB definitions and the SNMP protocol is much lower than the typical tasks that have to be fulfilled, so that well experienced staff is essential.

Section 2 explains how some of the core XML technologies are related and how they could be applied to typical network management tasks. In Section 3 we present some related work on XML based techniques for network management. In Section 4 we present and discuss our work on a MIB compiler that generates XML Schema definitions out of SMIv2 MIBs. Subsequently, in Section 5 we present some thoughts on potential XML based network management tasks. We also describe our approach to retrieve real-world SNMP data as XML documents through a simple SNMP-to-XML gateway that complies with our automatically generated XML Schema definitions. The last Section concludes with some experience statements and a rough outlook on future directions.

2. XML Technologies in Network Management

Clearly defined and standardized data structures, encoding schemes and access interfaces are a key issue for any kind of open communication systems. During the past in several areas different base technologies have been applied in a quite successful manner, e.g. management data definitions in the worlds of SNMP and CMIP as well as many protocol definitions are based on ASN.1, business data is exchanged by EDI-compliant messages, databases are often accessible through well defined ODBC or JDBC APIs, remote procedure calls and remote object access can be defined and realized through RPC and CORBA technologies, etc.

While these and many other technologies evolved over time and work quite well today, they have been developed independently. They do not build upon each other and they hardly use basic concepts in a common fashion, although many of them have to solve similar questions like byte ordering, data framing, data encryption, etc.

In contrast, XML [7] is a core building block upon which other related technologies are being developed. This "toolkit concept" allows for more efficient development processes of according applications. For example, existing XML parsers can be used to develop XSLT [8] processors and XML Schema [9, 10, 11] based validators, since XSL and XML Schema themselves are XML compliant. XML Schemas can be defined and used by validating XML parsers to ensure XML data integrity. Specific XML compilers can be built as XSL stylesheets using any already existing XSLT processor.

These advantages of existing XML-based specifications and their already existing implementations are applicable as building blocks to a wide range of network management aspects as well:

- **Management data can be represented as XML documents.** While the SNMP framework focuses on a simplistic management protocol and small items of management data which are appropriate for monitoring, it has severe disadvantages when larger chunks of configuration data has to be retrieved from an agent or stored to an agent: The performance of large chunks of GetNext or Set operations is quite poor [12]. Even more critical is the lack of a transaction model to ensure data integrity across a sequence of protocol operations. When a complete set of management data is represented as an XML document without any

formal length restrictions it could be moved efficiently and handled atomically. Of course, this requires an appropriate transport protocol.
Comparing XML to plain ASCII information that can be retrieved from many devices through a command line interface, parsing tagged XML data is much easier than the error-prone processing of hardly structured ASCII information that is intended for human reading.

- **Widely deployed protocols can be used to ship the data.** E.g. TCP and HTTP are implemented in almost any network device these days. These protocols can easily be used to transfer XML documents. URLs [13, 14] can be used to address the requested data [5].
- **The DOM and SAX APIs can be used to access management data from applications.** Most XML parsers implement these standard APIs to access the contents of XML documents ([15], et al). They can be used by individual management applications [6, 16, 5].
- **Items within management data documents can be addressed through XPath expressions** [17] [6]. This could be useful e.g. when only parts of very large data are being transfered between entities or when management data is processed by XML based applications, e.g. through XSLT:
- **XSLT can be used to process management data.** Although a little cryptic, XSL [8] is a quite powerful stylesheet language. E.g., it can be used to filter XML data, to correlate data from several documents, to generate statistics, or to create concise HTML pages or reports in other text-based formats.
- **The structure of management data can be expressed as XML Schemas** [9, 10, 11]. This would allow, e.g., to ensure the integrity of configuration data documents through the use of a usual XML parser that checks whether the document is well-formed and valid according to the XML Schema definition.
- **High-level management operations can be defined through WSDL and called via SOAP** [6]. E.g., row creation and deletion in SNMP through RowStatus objects can be quite complicated. This task can be achieved by higher-level operations in a more convenient way. However, these operations have to be well defined by WSDL [18] definitions for SOAP [19, 20, 21] functions.

Figure 1 illustrates how the XML technologies and according tools that are discussed above relate to each other. Specifications and tutorials for all of these technologies as well as pointers to detailed literature and implementations are available from the W3C [22].

3. Related Work

This section presents a vendor product, three research projects on Web and XML based management and an open source software project, as well as the starting points of related standardization work within the IETF and an industry consortium.

3.1 JUNOScript from Juniper Networks

Recent releases of routers from Juniper Networks are equipped with a JUNOS operating system that supports the JUNOScript [23] subsystem. JUNOScript allows client

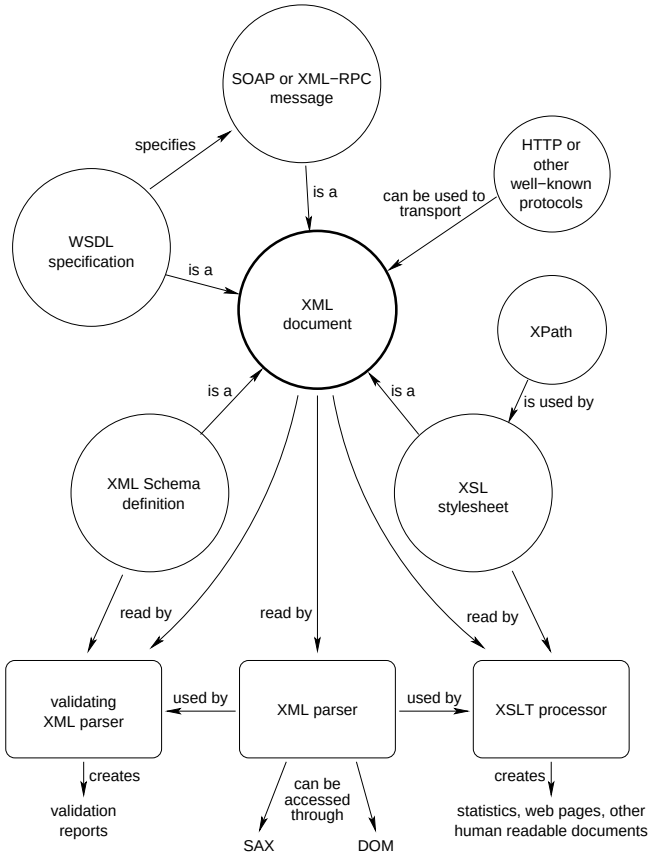

Figure 1. XML technologies and tools.

applications to connect to the Juniper router and exchange messages formed as XML documents. The grammars of these documents representing requests and responses are supplied by Juniper as DTDs and XML Schemas along with documentation that details the semantics of all message elements. A Perl module is supplied to ease the development of client applications communicating with the routers. Further processing of the XML documents can be achieved with third party XML tools that are available for Perl and many other programming languages.

Various protocols can be used to establish sessions between client applications and JUNOScript servers, e.g. TELNET, SSL or SSH. The messages sent by the client constitute RPC requests wrapped in `<rpc>...</rpc>` elements, upon which the server responds with `<rpc-reply>...</rpc-reply>` responses. The contained elements represent the requested actions, e.g. client authentication, configuration queries, configuration modifications, locking for exclusive access, etc. Collections of configuration data can be represented as nested XML elements within the messages as well as in the text format that is also used by the JUNOS CLI.

3.2 Avaya Labs Research on an XML based Management Interface for SNMP Enabled Devices

Concepts for an XML based interface to read and write management information of SNMP agents are being researched in a project at Avaya Labs [24]. To achieve these aims, a mapping from SMI MIB modules to XML Schema definitions has been defined and implemented. Furthermore, an XML-RPC based protocol is being defined and implemented for retrieving and modifying MIB information on SNMP agents. This protocol uses XPath to identify MIB variables within the agent.

The XML Schema definitions are derived from SMI MIB definitions in a straightforward way. For instance, the native XPath expression for addressing the first interface within an XML document derived from the IF-MIB would be `/IF-MIB/interfaces/ifTable/ifEntry[index='1']`. For one SMI MIB module the compiler generates a number of XML Schema files: one main file, one file containing all type definitions, one file per scalar object group, one file per additional MIB tree level and one file per MIB table. All these files are included by the main XML Schema file.

Much of the MIB information that is not necessarily required in XML Schema definitions is dropped, e.g. description clauses and display hints of type definitions. Some SMI types are mapped to more 'tolerant' XML Schema types, e.g. `IpAddress` is mapped to `xsd:string`. Elements representing scalar groups contain a redundant `index` attribute and, more seriously, tables with multiple index attributes are not yet supported. However, these problems could probably be fixed subsequently in this project.

The development and implementation of an appropriate SNMP-to-XML adapter based on the NET-SNMP and XML-RPC libraries has been underway during this writing.

3.3 POSTECH Research on XML-based Internet Management

Hong et al propose an XML-based Management (XBM) architecture [6] that is based on XML/HTTP as its management protocol. To allow the integration of SNMP managed devices they propose three methods to realize an XML/SNMP gateway [16].

The first approach is to implement a DOM interface that translates DOM function calls to SNMP operations. The results of the internally executed SNMP operations are then translated to XML nodes and passed back to the DOM based management application. In this approach the gateway is tightly bound to the management application through the DOM API.

In the second approach, a standalone gateway accepts HTTP requests from management stations that are based on URIs that may contain XPath or XQuery expressions to address specific agent object instances. The gateway parses these requests and translates them to SNMP operations issued to the SNMP agents. The SNMP responses are used to build a "response XML document" which is then sent back to the manager in response to its request. SNMP traps destined to the gateway can be processed in a similar way and forwarded as XML documents to managers through HTTP POST requests.

In the third approach, the gateway implements a SOAP RPC service. On receipt of a SOAP input message from a manager it is translated to SNMP operations. Vice

versa, the SNMP responses are used to construct the SOAP output message. While this approach introduces the most protocol and processing overhead, it also gives the best possibilities to extend the gateway with more powerful operations like scheduled polling.

Although this project carefully examines valuable methods to realize an XML/SNMP gateway, the mapping of SNMP management data to XML documents is quite simple and driven by SNMP practice, similar to the Avaya approach (Section 3.2).

3.4 WIMA

The Web-based Integrated Management Architecture (WIMA) proposed by Martin-Flatin [5] integrates multiple management data models, namely at least SMI and CIM, without introducing another data model. This is achieved by defined mappings of specific data models at two levels: In case of SMI, on the model-level an SMI MIB module is mapped to a DTD, i.e., an XML document that complies to such a DTD represents management instance data, e.g., retrieved from an SNMP agent. On the metamodel-level the items within an SMI module are mapped to element and attribute values within a document that complies to a generic WIMA-specified DTD.

WIMA's model-level SMI mapping is comparable to the mapping of SMI to XML Schema definitions presented in this paper: The resulting documents describe a formal grammar of management instance data. On the other hand, WIMA's metamodel-level mapping is also comparable to the XML Schema mapping presented in Section 4 in a way that its vocabulary is generic (WIMA-proprietary vs. XML Schema) while its content represents MIB data models.

As with the approaches presented in Section 3.2 and 3.3, the mapping from SNMP and SMI to XML and XML Schema is quite simple and bound to SNMP requirements and not driven by native XML concepts.

3.5 SCLI

`Scli` [25] is a tool that implements a command line interface to interact with SNMP agents in an interactive or batch-mode fashion. Despite many other SNMP tools, `scli` is not a generic tool to process arbitrary MIB data. Instead, it is designed in a way so that it is aware of the structure and semantics of a number of MIB modules. This way, it is capable to present and accept information to and from the user in a much more human friendly form that often significantly differs from the underlying MIB structures. E.g., when a user requests information on a specific interface, it can be specified by its human-friendly name instead of the agent's notion of an `ifIndex` variable. The data that is displayed is formatted in a human-readable tabular form where numbers are printed with appropriate units and the items are gathered from different tables of different MIBs like the `IF-MIB`, the `IP-MIB`, and the `ENTITY-MIB` [26, 27, 28]. It requires a well experienced person to develop new `scli` modes, but the resulting functionality facilitates the work of less experienced users significantly.

Besides the plain text form, `scli` is also capable to dump its information in an XML form. This way, it is possible, for example, to post-process the data by XSL stylesheets. The XSL programmer can benefit from `scli`'s gathering of all related information and its pre-processing. Currently, XML output is only supported for a

limited number of `scli` modes. XML Schema definitions for those XML dumps are not yet available.

3.6 The IETF XMLCONF BOF

During the 54th IETF meeting in Yokohama in July 2002, a BOF session concerned with XML configuration management (XMLCONF) has been held [29]. The goals were to discuss today's operator requirements for configuration management, to identify the disadvantages of today's IETF technologies to meet these requirements and to evaluate some XML related technologies with this respect, namely, SOAP/WSDL, SyncML, WBEM, and JUNOScript. There are some Internet-Drafts that represent a starting point to narrow down these requirements and evaluations. It has not yet been decided whether a new working group shall be formed to continue this work, but it looks very reasonable to work on the standardization of XML based network management within the IETF so that the gap between the wide range of existing SNMP environments and new XML based solutions can be bridged.

3.7 The OASIS Management Protocol Technical Committee

In July 2002, OASIS, a consortium that focuses on industry standards specifications based on XML, founded a technical committee [30] that intends to provide a web-based mechanism to monitor and control managed elements "based on industry accepted management models, methods, and operations, including, OMI, XML, SOAP, DMTF CIM, and DMTF CIM Operations".

4. Converting SMI MIBs to XML Schema Definitions

While the previous two Sections presented general concepts and actual efforts to use XML technologies for network management tasks, in this Section we will study how the large existing SNMP infrastructure can benefit from XML based management data processing. This obviously requires a representation of SNMP data as XML documents.

While the structure of SNMP data is formally described in SMI MIB modules, the structure of XML documents can be defined as a Document Type Definition (DTD) or an XML Schema definition. Whereas DTDs follow a rather simple grammar notation, XML Schema is more flexible to express characteristics of XML documents, e.g., there is a type system that allows to define derived types, and a powerful mechanism to express the format of string values as regular expressions. Furthermore, XML Schema definitions themselves are well-formed XML documents so that they can also be processed by XML parsers.

Consequently, a mapping from SMI MIB modules to XML Schema definitions is useful. While the approaches presented in Sections 3.2 – 3.4 use a straightforward way, we attempt to represent the data in a way that narrows the usual XML characteristics as closely as possible to make the XML instance documents as convenient for reading and processing as possible. For instance, we explicitly do not intend to stick with deep OID equivalent nesting hierarchies. On the other hand, the generated XML Schema definitions contain as much of the underlying SMI MIB module information as possible. The following list describes the most relevant characteristics of the XML documents and XML Schema definitions:

- The range of data that can be represented in an XML document shall be as flexible as possible. Hence, the root element is not bound to a specific MIB or agent or point in time.
- The root element may contain an arbitrary number of `<context>` elements at the second level that represent agent contexts which are identified by a tuple of an SNMP agent, a community string (in case of SNMPv1), and a time stamp to specify the point in time when a context has been created. This allows, for instance, to store data from multiple agents or time series of polled data in a single document.
- The third-level elements can either represent containers of scalar elements that appear at most once, or instances of objects that are derived from table entries and thus can appear multiple times. Note that the list of these elements is not limited to a single MIB module. While scalar container elements don't have any attributes, the table entry elements include one ore more index attributes to uniquely identify the instances. These attributes are derived from the MIB entry's `INDEX` clauses.
- Note that MIB modules are not represented by elements. Instead they are identified by namespaces in which the elements are defined. Since unique naming in SMI is based on modulename-descriptor pairs, we compile each MIB module into a separate XML Schema definition where each schema defines an according namespace.
- The fourth-level elements represent scalar objects or columnar objects. There is no deeper level of element containment (except for nested tables described below), since there is no need for a hierarchy such as with OIDs. This way the XML document hierarchy remains simple but powerful. Unique naming is purely based on namespaces, grouping names with indexing attributes, and leaf element names without any ambiguity.
- Augmentation tables and tables that share a common prefix list of index objects with another table are somewhat confusing SMI constructs to represent nested data structures. When these structures appear in a single MIB they are mapped to a native representation in XML: The columnar objects of augmentation tables are simply added as child elements of the element that represents the parent table. Similarly, "tables in tables" are represented by nesting the according elements.
- The text of leaf elements (and index attributes) is represented in a human readable fashion where possible: Integers are written as decimal numbers. Named number types, such as enumeration types and bit sets, are written as the according names. Strings are written in a human readable ASCII form, if the underlying MIB type has a display hint that 'suggests' an according representation of all octets of the value.
- Type and `TEXTUAL-CONVENTION` definitions in MIB modules are mapped to XML Schema types derived from base types with appropriate `<xsd:restriction>` clauses containing value restrictions for numbers (`<xsd:minInclusive>`, `<xsd:maxInclusive>`) or strings (`<xsd:minLength>`, `<xsd:maxLength>`). However, some limitations

such as length alternatives (i.e., not length ranges) cannot be expressed completely in XML Schema.

- Even complex display hints can automatically be translated to `<xsd:pattern>` constructs with regular expressions that formally limit the value set so that, e.g., a number of typos or otherwise illegal values in XML documents can be prevented.
- SMI MIB module information that is not necessarily required in the XML Schema definition is contained in `<xsd:appinfo>` clauses. This way, it remains available for special applications, e.g., XSLT-based MIB compilers or MIB browsers.
- Despite the representation of the status of MIB object instances an XML document can also represent a sequence of notifications. Hence, the XML Schema for a MIB module contains a `<xsd:choice>` construct where the second alternative is intended to form the grammar rules for arbitrary lists of notifications conforming to a MIB's `NOTIFICATION-TYPE` definitions.

Figure 2 shows an example of an XML instance document, while Figure 3 illustrates some XML Schema constructs compiled from the SMIv2 `IF-MIB` module [26]. The compiler implementation is based on `libsmi`[31].

```
<?xml version="1.0"?>
<!-- This module has been generated by mibdump 0.1. Do not edit. -->
<snmp-data xmlns="http://www.ibr.cs.tu-bs.de/projects/libsmi/xsd/IF-MIB"
           xmlns:IF-MIB="http://www.ibr.cs.tu-bs.de/projects/libsmi/xsd/IF-MIB"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://www.ibr.cs.tu-bs.de/projects/libsmi/xsd/IF-MIB
                               http://www.ibr.cs.tu-bs.de/projects/libsmi/xsd/IF-MIB.xsd">
  <context agent="ciscobs.rz.tu-bs.de" community="public" port="161"
           time="2002-12-12T23:42+0100">
    <IF-MIB:interfaces>
      <IF-MIB:ifNumber>7</IF-MIB:ifNumber>
    </IF-MIB:interfaces>
[...]
    <IF-MIB:ifEntry ifIndex="2">
      <IF-MIB:ifDescr>FastEthernet0/0</IF-MIB:ifDescr>
      <IF-MIB:ifType>ethernetCsmacd</IF-MIB:ifType>
      <IF-MIB:ifMtu>1500</IF-MIB:ifMtu>
      <IF-MIB:ifSpeed>100000000</IF-MIB:ifSpeed>
      <IF-MIB:ifPhysAddress enc="hex">00:03:fd:32:e4:00</IF-MIB:ifPhysAddress>
      <IF-MIB:ifAdminStatus>up</IF-MIB:ifAdminStatus>
[...]
      <IF-MIB:ifName>Gi1/0</IF-MIB:ifName>
      <IF-MIB:ifLinkUpDownTrapEnable>enabled</IF-MIB:ifLinkUpDownTrapEnable>
[...]
    </IF-MIB:ifEntry>
[...]
    <IF-MIB:ifStackEntry ifStackHigherLayer="2" ifStackLowerLayer="0">
      <IF-MIB:ifStackStatus>active</IF-MIB:ifStackStatus>
    </IF-MIB:ifStackEntry>
  </context>
</snmp-data>
```

Figure 2. An XML instance document conforming to the IF-MIB XML Schema.

```
<?xml version="1.0"?>
<!-- This module has been generated by smidump 0.4.2-pre. Do not edit. -->
<xsd:schema
```

```
    targetNamespace="http://www.ibr.cs.tu-bs.de/projects/libsmi/xsd/IF-MIB"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:smi="http://www.ibr.cs.tu-bs.de/projects/libsmi/xsd/smi"
    xmlns:SNMPv2-SMI="http://www.ibr.cs.tu-bs.de/projects/libsmi/xsd/SNMPv2-SMI"
    xmlns:SNMPv2-TC="http://www.ibr.cs.tu-bs.de/projects/libsmi/xsd/SNMPv2-TC"
    xmlns:SNMPv2-CONF="http://www.ibr.cs.tu-bs.de/projects/libsmi/xsd/SNMPv2-CONF"
    xmlns:SNMPv2-MIB="http://www.ibr.cs.tu-bs.de/projects/libsmi/xsd/SNMPv2-MIB"
    xmlns:IANAifType-MIB="http://www.ibr.cs.tu-bs.de/projects/libsmi/xsd/IANAifType-MIB"
    xml:lang="en"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">
  <xsd:annotation>
    <xsd:documentation>
        The MIB module to describe generic objects for network
        interface sub-layers.  This MIB is an updated version of
        MIB-II's ifTable, and incorporates the extensions defined in
        RFC 1229.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:import namespace="http://www.ibr.cs.tu-bs.de/projects/libsmi/xsd/SNMPv2-SMI"
         schemaLocation="http://www.ibr.cs.tu-bs.de/projects/libsmi/xsd/SNMPv2-SMI.xsd"/>
[...]
  <xsd:element name="snmp-data">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="context" minOccurs="0" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="interfaces" type="interfacesType" minOccurs="0"/>
              <xsd:element name="ifEntry" type="ifEntryType" minOccurs="0" maxOccurs="unbounded"/>
[...]
            </xsd:sequence>
            <xsd:attribute name="agent" type="xsd:NMTOKEN" use="required"/>
            <xsd:attribute name="community" type="xsd:NMTOKEN" use="required"/>
            <xsd:attribute name="port" type="xsd:unsignedInt" use="required"/>
            <xsd:attribute name="time" type="xsd:dateTime" use="required"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
[...]
  <xsd:complexType name="ifEntryType">
    <xsd:annotation>
      <xsd:appinfo>
        <maxAccess>not-accessible</maxAccess>
        <status>current</status>
        <oid>1.3.6.1.2.1.2.2.1</oid>
      </xsd:appinfo>
      <xsd:documentation>
        An entry containing management information applicable to a
        particular interface.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="ifDescr" minOccurs="0">
        <xsd:annotation>
          <xsd:appinfo>
            <maxAccess>read-only</maxAccess>
            <status>current</status>
            <oid>1.3.6.1.2.1.2.2.1.2</oid>
          </xsd:appinfo>
          <xsd:documentation>
            A textual string containing information about the
            interface.  This string should include the name of the
            manufacturer, the product name and the version of the
            interface hardware/software.
          </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
```

```
            <xsd:pattern value=".{0,255}"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="ifType" type="IANAifType-MIB:IANAifType" minOccurs="0">
        <xsd:annotation>
[...]
  <xsd:simpleType name="InterfaceIndex">
    <xsd:annotation>
      <xsd:documentation>
          A unique value, greater than zero, for each interface or
          interface sub-layer in the managed system.  It is
          recommended that values are assigned contiguously starting
          from 1.  The value for each interface sub-layer must remain
          constant at least from one re-initialization of the entity's
          network management system to the next re-initialization.
      </xsd:documentation>
      <xsd:appinfo>
        <displayHint>d</displayHint>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:restriction base="smi:Integer32">
      <xsd:minInclusive value="1"/>
      <xsd:maxInclusive value="2147483647"/>
[...]
</xsd:schema>
```

Figure 3. An XML Schema file automatically generated from IF-MIB.

5. Applications

The previous Section presented an approach to represent management data as XML documents and their structure as XML Schema definitions. So, in this Section we will discuss some applications that could make use of these representations. Note that at this point in time these applications, except the one presented in Section 5.4, are just ideas that have not yet been implemented by the authors of this paper.

5.1 Configuration Management

SNMP is not very suitable for configuration management, since moving configuration data between devices and managers is inefficient and cannot be done in an atomic way without additional MIB support. Putting XML technologies on top of it cannot do any better. Furthermore, data dumped from arbitrary MIBs cannot be regarded as a configuration: To restore a configuration, e.g., if a broken device has been replaced by a new one, additional information like ordering of the required SNMP Set operations and the semantics of many MIB objects is required. However, it could be an advantage if such additional information on a specific MIB is provided in a DOM or XSLT based application and the dumped data is available as an XML document. Accessing data through the DOM API or addressing items through XPath expressions would be more flexible and more familiar to many developers than using any proprietary management toolkit.

5.2 Notification Processing

Not only the values of agent objects that can be read or written, i.e., instances of the MIBs' `OBJECT-TYPE` definitions, can be represented as XML elements. Instances

of notifications emitted by an agent can be dumped as XML documents as well. This way, they can be filtered, searched, and post-processed by XPath and XSLT means, for example. Even event correlation tasks could be realized as XSLT applications and produce HTML output for a concise presentation to the operator.

5.3 Agent Validation

Today, there are many agent implementations of MIBs that do not conform to the underlying SNMP and SMI MIB module specifications. Typical cases are objects that have another base type than that in the authoritative MIB definition, string objects of illegal length or object values that the agent is not aware, but which are served as unspecified 'zero' values instead of just skipping the object.

To some degree such flawed implementations can be checked automatically based on a comparison of an agent's dump against the MIB specification. If the data is represented as an XML document and the MIB structure is represented as an XML Schema definition, this comparison can be done by any validating XML parser. Of course, this would require a very 'verbose' representation of agent data that keeps a lot of SNMP specific information in the XML document which is not necessarily required for other XML based post-processing.

5.4 An SNMP-to-XML Gateway

An obvious approach to make management data supplied by SNMP agents available as XML documents is the use of an appropriate translator or gateway. We have implemented a prototype of such a translator that conforms to the XML Schema characteristics described in Section 4. The example of an XML document seen in Figure 2 has actually been generated by this translator named `mibdump`. It works as follows: The SNMP agent (address, port, SNMPv1 community string) and the MIB module to be dumped are passed to `mibdump`. Then the SNMP session is initiated, the structure of the MIB is analyzed through `libsmi` means, and then sequences of SNMP GetNext operations are issued to retrieve all subtrees of the MIB from the agent. `Mibdump` collects the retrieved data in internal data structures, first. When the gathering phase has been finished, the contents of these data structures are dumped in the form of appropriately nested XML elements forming a valid document with respect to the XML Schema.

`Mibdump` has been developed as a first prototype of a translator to generate XML instance documents that comply to the XML Schema definitions we proposed in Section 4. Hence, for simplicity reasons it supports no other granularity than the level of MIB modules: neither single objects or tables of a MIB, nor the whole set of all MIBs implemented by an agent can be retrieved through `mibdump` at once.

Work is underway to develop a real gateway that translates HTTP requests for XML documents to SNMP operations and forms the resulting XML document out of the SNMP response messages. This approach is very similar to the second method proposed by POSTECH (Section 3.3, [16]). The gateway accepts HTTP GET requests for URIs that contain XPath expressions to address regions of the schema compliant XML documents with the full complexity of XPath. Generally, location paths of the XPath expressions can be interpreted in the gateway before sending SNMP requests in order limit SNMP operations, while the predicate parts have to be applied when the SNMP data has been received at the gateway to filter out the parts that the XML

requestor wishes to receive. DOM is used to represent and access the XML documents at runtime in the gateway. In case of HTTP GET requests the DOM is built by the core translator based on the received SNMP response messages. In case of HTTP POST requests the applied document is parsed to build the DOM so that the translator can access it to issue appropriate SNMP Set operations. Traps can be logged (for later access by managers) and result in short XML documents sent to registered managers that listen as HTTP POST receivers. To enhance performance in case of subsequent XML operations on related objects, a short-term cache can be used. Problems related by caching and write operations to create or delete objects are subject for future work.

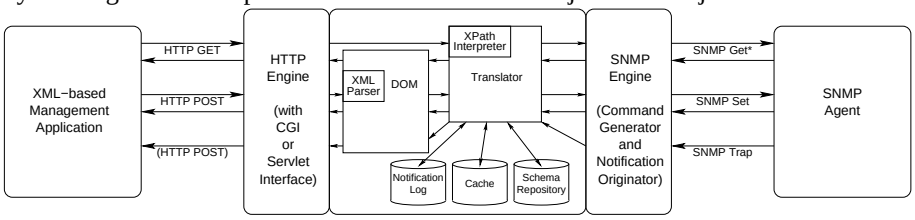

Figure 4. Architecture of an SNMP/XML gateway.

6. Conclusions and Outlook

This paper gave a condensed overview of some XML technologies and how they can be used to solve network management tasks that are partly difficult to address with the current SNMP framework as it is. A number of related projects and standardization efforts that are underway have been presented. However, it has been argued that a smooth bridging between the widely deployed SNMP infrastructure and the 'new generation' of XML based solutions is essential.

The presented approach to represent management data received from and stored to SNMP agents as XML documents and to represent their structure as automatically generated XML Schema definitions based on SMI MIB modules constitutes one step in this direction. In contrast to other attempts that define a simple straightforward mapping of SMI data models to DTDs or schemas the presented approach is driven by the goal to tap the full potential of XML and XML Schema leaving drawbacks of SNMP behind where possible. Furthermore, this paper presents some visions on other XML applications. Within the presented project, an SNMP-to-XML gateway is under development that represents management data conforming to the presented XML Schema model.

The standardization work done so far in the area of XML/SNMP integration is not much more than the discussion of requirements and the evaluation of some technologies. In the future, more work on specific schemas and protocol support has to be done to support the higher-level tasks in XML based network management and configuration management.

References

[1] M. Rose and K. McCloghrie. Structure and Identification of Management Information for TCP/IP-based internets. RFC 1065, TWG, August 1988.

[2] K. McCloghrie and M. Rose. Management Information Base for Network Management of TCP/IP-based internets. RFC 1066, TWG, August 1988.
[3] J. Case, M. Fedor, M. Stoffstall, and J. Davin. A Simple Network Management Protocol. RFC 1067, University of Tennessee at Knoxville, NYSERNet, Rensselaer Polytechnic Institute, Proteon, August 1988.
[4] J. Schönwälder and A. Pras. The Simple Times. An openly-available online publication on SNMP, http://www.simple-times.org/.
[5] J.-P. Martin-Flatin. *Web-Based Management of IP Networks and Systems*. Wiley, 2002.
[6] H. Ju, M. Choi, S. Han, Y. Oh, J. Yoon, H. Lee, and J. W. Hong. An Embedded Web Server Architecture for XML-Based Network Management. In *Proc. 2002 IEEE/IFIP Network Operations and Management Symposium*, April 2002.
[7] T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler. Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation, October 2000.
[8] J. Clark. XSL Transformations (XSLT) Version 1.0. W3C Recommendation, November 1999.
[9] D. C. Fallside. XML Schema Part 0: Primer. W3C Recommendation, IBM, May 2001.
[10] H. S. Thompson, D. Beech, M. Maloney, and N. Mendelsohn. XML Schema Part 1: Structures. W3C Recommendation, University of Edinburgh, Oracle, Lotus, May 2001.
[11] P. V. Biron and A. Malhotra. XML Schema Part 2: Datatypes. W3C Recommendation, Kaiser Permanente, Microsoft, May 2001.
[12] R. Sprenkels and J. P. Martin-Flatin. Bulk Transfer of MIB Data. *Simple Times*, 7(1), March 1999.
[13] T. Berners-Lee, L. Masinter, and M. McCahill. Uniform Resource Locators (URL). RFC 1738, CERN, Xerox Corporation, University of Minnesota, December 1994.
[14] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifiers (URI): Generic Syntax. RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998.
[15] L. Wood, et al. Document Object Model (DOM) Level 1 Specification Version 1.0. W3C Recommendation, Soft Quad, October 1998.
[16] Y. Oh, H. Ju, M. Choi, and J. W. Hong. Interaction Translation Methods for XML/SNMP Gateway. In *Proc. 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, October 2002.
[17] J. Clark and S. DeRose. XML Path Language (XPath) Version 1.0. W3C Recommendation, Inso Corp., November 1999.
[18] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1. W3C Note, Microsoft, IBM, March 2001.
[19] N. Mitra. SOAP Version 1.2 Part 0: Primer. W3C Working Draft, Ericsson, June 2002.
[20] M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, and H. F. Nielsen. SOAP Version 1.2 Part 1: Messaging Framework. W3C Working Draft, DevelopMentor, Sun, IBM, Canon, Microsoft, June 2002.
[21] M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, and H. F. Nielsen. SOAP Version 1.2 Part 2: Adjuncts. W3C Working Draft, DevelopMentor, Sun, IBM, Canon, Microsoft, June 2002.
[22] W3C. W3C – The World Wide Web Consortium. WWW Page, 2002. http://www.w3c.org/.
[23] Juniper Networks. JUNOScript API Software. WWW Page, November 2002. http://www.juniper.net/support/junoscript/.
[24] Avaya Labs Research. XML-Based Management Interface for SNMP Enabled Devices. WWW Page, 2001. http://www.research.avayalabs.com/user/mazum/Projects/XML/.
[25] J. Schönwälder. Specific Simple Network Management Tools. In *Proc. LISA 2001*, pages 109–119, December 2001. http://www.ibr.cs.tu-bs.de/projects/scli/.
[26] K. McCloghrie and F. Kastenholz. The Interfaces Group MIB. RFC 2863, Cisco Systems, Argon Networks, June 2000.
[27] K. McCloghrie. SNMPv2 Management Information Base for the Internet Protocol using SMIv2. RFC 2011, Cisco Systems, November 1996.
[28] K. McCloghrie and A. Bierman. Entity MIB (Version 2). RFC 2737, Cisco Systems, December 1999.
[29] M. Wasserman. XML Configuration BOF. WWW Page, July 2002. http://www.ietf.org/ietf/02jul/xmlconf.txt.
[30] OASIS. Management Protocol TC. WWW Page, December 2002. http://www.oasis-open.org/committees/mgmtprotocol/.
[31] F. Strauß. Libsmi – A Library to Access SMI MIB Information. WWW Page, December 2002. http://www.ibr.cs.tu-bs.de/projects/libsmi/.