Netzwerkmanagement mit NETCONF und YANG

Julian Sommerfeldt
Betreuer: Marc-Oliver Pahl
Seminar Future Internet WS09/10
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik
Technische Universität München
Email: sommerfe@in.tum.de

***Kurzfassung*— NETCONF hat mit dem Jahr 2009 einen Status erreicht, den man als weitestgehend vollständig bezeichnen kann. Die Entwicklung durch die IETF Working Group wurde motiviert durch die Tatsache, dass SNMP nicht wie ursprünglich gedacht, zur Erfüllung von Konfigurationsaufgaben für Netzwerkgeräte benutzt wurde, sondern lediglich für Monitoringzwecke. Das XML-basierte Protokoll reagiert darauf mit der Zielsetzung, die bisher üblichen Command-Line-Interfaces zu ersetzen und den Administratoren somit eine flexible, einfach und schnell zu erlernende Möglichkeit zur Verfügung zu stellen.**
Unterstützt wird dieser Reifegrad durch die Datenmodellsprache YANG, die für das Protokoll entworfen wurde und den Zugang noch weiter erleichtert.

***Schlüsselworte*— Netzwerk, Management, NETCONF, YANG**

I. Einleitung

Um die immer komplexeren Netzwerkstrukturen und die immer größer werdende Vielfalt an Geräten die darin interagieren, in den Griff zu bekommen, war eine Standardisierung notwendig, die die Netzwerkadministratoren unterstützt. Mit diesem Ziel wurde 2003 die IETF Working Group gegründet, deren Arbeit nun bereits durch einige Implementierungen, beispielsweise von Cisco, und einer stark wachsenden Anwendergruppe bestätigt wurde. Da vorher im Allgemeinen Kommandozeilenprogramme beziehungsweise sehr spezielle, auf bestimmte Geräte zugeschnittene, Lösungen verwendet wurden, kann das NETCONF Protokoll im Vergleich zu dem, was bisher eingesetzt wurde, als komplette Neuheit bezeichnet werden.

II. Anforderungen

Die Anforderungen und deren Motivation, die zu Beginn festgesetzt wurden, werden hier grob erläutert. [1], [2]

A. Differenzierung verschiedener Datenarten

Innerhalb der Netzwerkgeräte kann grob unterschieden werden zwischen Konfigurationsdaten und Statusdaten. Erste werden primär dazu verwendet, um Einstellungen zu manipulieren und das Gerät somit zu steuern. Zweitere geben beispielsweise Auskunft über Temperatur oder Laufzeit seit dem letzten Start. Da oftmals nur die eine oder die andere Art der Daten benötigt wird, ist es sinnvoll, dass man zwischen ihnen unterscheiden und somit gezielter arbeiten kann und eine kleinere Datentransfermenge hat.

B. Erweiterbarkeit

Die Fülle an unterschiedlichen Geräten bringt auch eine große Menge unterschiedlichster Daten mit sich, die alle ansteuerbar sein müssen. Hierzu ist es notwendig, dass ein hohes Maß an Flexibilität bereitgestellt wird, wodurch die Hersteller mit einem einzigen Protokoll Zugang zu allen Daten gewährleisten.

C. Programmierbare Schnittstelle

Dieser Punkt stellt den Schritt weg von der Kommandozeile hin zu einfacher steuerbaren Programmen dar. Die Administratorentätigkeit wird auf eine wesentlich abstraktere Ebene gehoben, wodurch die Konfiguration leichter fällt, wenn man nicht auf tiefster Ebene mit der entsprechenden Arbeitsweise vertraut ist. Des Weiteren wird durch unterschiedliche Implementierungsmöglichkeiten an Flexibilität gewonnen.

D. Textuelle Repräsentation

Die Konfigurationsdaten sollen rein durch „einfachen“ Text festgelegt werden, um zu verhindern, dass ein spezialisiertes Tool verwendet werden muss. Hierzu eignet sich XML, da dieses sowohl einfach zu erlernen und zu lesen sowie bereits weit verbreitet und anerkannt ist. Jeder kann somit ohne aufwendige Lernnotwendigkeiten relativ einfach Konfigurationen durchführen.

E. Authentifizierung

Da es sich bei der Konfiguration um höchst sensible Daten handelt, ist es notwendig, dass zu Beginn einer Session eine Authentifizierung stattfindet, wobei diese Aufgabe von bereits bestehenden Methoden, wie dem Transportprotokoll, unterstützt wird.

F. Integration existierender Konfigurationsdatenbanksysteme

Um zu verhindern, dass ein völlig neuer Aufbau stattfinden muss, war eine Anforderung, die Fähigkeit der Zusammenarbeit mit bereits bestehenden Systemen, so dass mit diesen weitergearbeitet werden kann.

G. Transaktionen

Damit die Möglichkeit einer Konfiguration durch mehrere Stellen gegeben ist, müssen transaktionsbezogene Funktionen bereitgestellt werden. Hier sind Stichwörter wie *locking* und *rollback* zu nennen, die Komplikationen des Mehrbenutzerbetriebs verhindern sollen.

H. Transportunabhängigkeit

Ein weiterer Punkt, der unter Flexibilität einzuordnen ist, ist die Unabhängigkeit des Transportprotokolls. Das Ziel, alle Geräte, mit unterschiedlichen Protokollen zu unterstützen, erreicht NETCONF dadurch, dass es mit allen gängigen Transportprotokollen implementiert werden kann. Neben SSH im Vordergrund existieren bereits Implementierung die mit BEEP und SOAP arbeiten.

III. Architektur

Die Architektur des Protokolls ist in vier Ebenen aufgeteilt, wie sie in Tabelle I dargestellt ist. [2]

Tabelle I
ARCHITEKTUR DES NETCONF PROTOKOLLS [2, S. 8]

Ebene	Beispiel
Inhalt	Konfigurationsdaten
Operationen	<get-config>, <edit-config>
RPC	<rpc>, <rpc-reply>
Transportprotokoll	SSH, BEEP, SSL, Console

Im Folgenden wird der Aufbau näher erklärt, Beispiele werden erläutert und es wird aufgezeigt, wie auf die Anforderungen aus II eingegangen wird. Die Listings, die dabei verwendet werden, lehnen sich an [2] an.

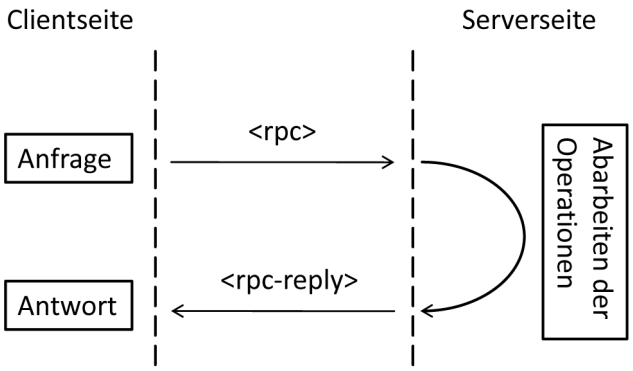
A. Transportprotokoll

Als unterliegendes Transportprotokoll kann jedes gewählt werden, das gewisse Anforderungen erfüllt, wobei eine Implementierung mindestens SSH unterstützen muss, wodurch II-H erfüllt ist. Die Verbindungsorientierung ist wichtig, damit eine persistente Kommunikation zwischen Client und Server stattfinden kann. Wichtig ist dies bei Authentifizierungsdaten, die während einer Session erhalten bleiben.
Das Aufgabenfeld der in II-E erwähnten Authentifizierungsanforderung wird auf das Transportprotokoll ausgelagert und muss somit davon, ebenso wie die Integrität und Vertraulichkeit der Daten, erfüllt beziehungsweise gewährleistet werden.

B. RPC

NETCONF benutzt Remote Procedure Call (RPC) um vom Client aus Anfragen an den Server zu stellen. Wie in II-D gefordert, wird hierzu XML verwendet, wobei alle NETCONF Nachrichten sich im Listing 1 angegebenen Namespace befinden müssen. Ebenso stellt es eine RPC-Anfrage in XML dar, welche wohlgeformt sein muss. Dieser Vorgang und das Verhalten der Akteure werden in Abbildung 1 dargestellt. Durch die Verwendung von XML besteht die Möglichkeit, beliebige Interfaces zu erstellen (II-C) und durch Umformung älterer Konfigurationen neue XML-basierte zu erhalten (II-F).

Abbildung 1. Ablauf des RPC-Requests



Listing 1. RPC

```
<rpc message-id="1337"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">

        <!-- Inhalt -->

</rpc>
```

Man sieht den Starttag <rpc>, der zwei Parameter aufweist: Der erste ist eine Id, mit deren Hilfe der Aufruf identifiziert werden kann, was unter anderem für den <rpc-reply>, also die Antwort, notwendig ist, um eine klare Zuordnung zu ermöglichen.
Im folgenden Beispiel (Listing 2) wird die Methode <get>, eine der Basisoperationen von NETCONF, ausgeführt und der gesamte Datensatz zurückgegeben, wobei der „Header" unverändert bleibt.

Listing 2. RPC-Reply

```
<rpc message-id="1337"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <get\>
</rpc>

<rpc-reply message-id="1337"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <data>
              <!-- Kompletter Datensatz -->
        </data>
</rpc-reply>
```

Darüber hinaus gibt es noch den Tag <rpc-error>, der bei Fehlern zum Einsatz kommt und den Tag <ok>, der verwendet

wird, wenn kein Fehler aufgetreten ist, es aber auch keine Daten gibt, die zurückgegeben werden.

C. Operationen

Das NETCONF Protokoll stellt Basisoperationen zur Verfügung, mit denen einige Fähigkeiten realisiert werden.

1) Inhaltsbezogene Operationen:

- <get>
  Fordert alle, also Konfigurations- und Statusdaten, an.
- <get−config>
  In II-A wird eine Differenzierung zwischen den Daten gefordert, was unter anderem dadurch realisiert wird, dass hier nur die Konfigurationsdaten abgefragt werden.
- <edit−config>
  Hierbei wird die Konfiguration direkt geändert, wobei folgende unterschiedliche Operationen verfügbar sind: merge, replace, create, delete.
- <copy−config>
  Kopiert eine komplette Konfiguration, je nach Situation, über eine bereits Existierende oder erstellt diese.
- <delete−config>
  Löscht eine Konfiguration.

2) Transaktionsbezogene Operationen (II-G):

- <lock>
  Sperrt eine Konfiguration, um ein zwischenzeitliches Eingreifen zu verhindern.
- <unlock>
  Gibt eine Sperre wieder frei.

3) Sessionbezogene Operationen:

- <close−session>
  Schließt eine Session, wobei alle gehaltenen Locks entsperrt und alle Verbindung geschlossen werden.
- <kill−session>
  Der Operation <close−session> ähnlich, allerdings werden alle laufenden Operationen abgebrochen und es wird ein sofortiges Schließen erzwungen.

D. Inhalt

Der Inhalt ist nicht direkt Teil der Spezifikation des NETCONF Protokolls, da sie kein Datenmodell angibt, sondern der der jeweiligen Implementierung. Die dadurch gewonnene Flexibilität ermöglicht ein Erfüllen der Anforderung II-B. Mit YANG wurde für diesen Teil ein Datenmodell geschaffen, welches aufgrund seiner Eigenschaften für diese Ebene verwendet werden kann. Hierauf wird in V näher eingegangen.

IV. Weitere Fähigkeiten

Neben den bisher dargestellten Grundlagen bietet das Protokoll noch andere Möglichkeiten, welche genaueres Steuern (IV-A) beziehungsweise neue Fähigkeiten (IV-B) bereitstellen. Beide Gebiete werden in [2] näher erläutert. Die folgenden Codebeispiele wurden wieder grob daraus entnommen.

A. Subtree Filtering

Um die Daten besser zu kategorisieren, werden sie in Bäumen verwaltet, die die Datenstruktur bilden. Kinder eines Baumes können entweder Knoten, also Enden, in denen Daten gespeichert sind, sein, oder wiederum Bäume, die sogenannten Subtrees. Subtree Filtering wird darauf angewandt, um Daten expliziter abzufragen, da wir bisher nur die Möglichkeit kennen gelernt haben, beispielsweise durch <get−config> die gesamte Konfiguration abzufragen. Dies führt häufig zu Unübersichtlichkeit und es findet ein unnötig hoher Datentransfer statt, der nur selten notwendig ist. Es gibt einige Typen von Filtern, hier wird allerdings die Filterung, die direkt auf den Inhalt bezogen ist exemplarisch vorgestellt. Wie in III-B bereits erklärt, läuft die Kommunikation über RPC ab. So sieht man im folgenden Listing 3 den umschließenden <rpc> Tag und die darin befindliche Filterung.

Listing 3. Subtree Filtering

```
<rpc message-id="1337"
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get-config>
  <source>
   <running/>
  </source>
  <filter type="subtree">
   <top
    xmlns="http://example.com/schema/1.2/config">
    <users>
     <user>
      <name/>
     </user>
    </users>
   </top>
  </filter
 </get-config>
</rpc>
```

Zuerst wird festgelegt, dass die laufende Konfiguration angefordert wird. Im Filter wird dann durch das *top* Element ein Namespace definiert, innerhalb dessen sich die Daten befinden müssen. Hieraus wiederum werden erst alle User „selektiert“ und dann von jedem Einzelnen jeweils nur das Attribut *name*. Angenommen es gäbe drei User im entsprechenden Namespace, so gibt der Server folgende Daten zurück (Listing 4).

Listing 4. Subtree Filtering Reply

```
<rpc-reply message-id="1337"
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <data>
  <top
   xmlns="http://example.com/schema/1.2/config">
   <users>
    <user>
     <name>root</name>
    </user>
    <user>
     <name>Alfred</name>
    </user>
    <user>
```

```
      <name>Heidi</name>
     </user>
    </users>
   </top>
  </data>
</rpc-reply>
```

Wie man sieht, lassen sich mit der Anwendung von Subtree Filtern so exakte Datenabfragen bzw. -manipulationen durchführen und effizient arbeiten.

B. Capabilities

Capabilities sind Implementierungsmöglichkeiten für bestimmte Einsatzzwecke, wobei man sie als eine Art Erweiterung der Basisoperationen verstehen kann. Für viele Capabilities ist eine entsprechende Kommunikation zwischen Client und Server notwendig. Deshalb sendet jedes der beiden Enden am Beginn einer Session eine *hello*-Message, in der neben einer Id alle unterstützten Capabilities eingetragen sind. In Listing 5 sieht man eine solche Nachricht.

Listing 5. Hello Message

```
<hello
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <capabilities>
  <capability>
   urn:ietf:params:netconf:base:1.0
  </capability>
  <capability>
   urn:ietf:params:netconf:capability:startup:1.0
  </capability>
  <capability>
   http://example.net/router/2.3/myfeature
  </capability>
 </capabilities>
 <session-id>4</session-id>
</hello>
```

Als ein Beispiel sei hier die *Rollback on Error Capability* beschrieben, welche die Fähigkeit, beim Auftreten von Fehlern ein Rollback durchzuführen, zur Verfügung stellt. Sollte beispielsweise im Listing 6 beim Setzen des *mtu* Parameters ein Fehler auftreten, so werden alle Änderungen auf den Stand zurückgesetzt, der vor der Ausführung dieser Anfrage aktuell war, also wird der *name* wieder auf den Ursprünglichen gesetzt.

Listing 6. Rollback on Error

```
<rpc message-id="1337"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <running/>
  </target>
  <error-option>rollback-on-error</error-option>
  <config>
   <top
   xmlns="http://example.com/schema/1.2/config">
    <interface>
     <name>Ethernet0/0</name>
     <mtu>100000</mtu>
    </interface>
   </top>
  </config>
 </edit-config>
</rpc>
```

Darüber hinaus kann man noch die *With-defaults Capability* nennen, welche die Mitübergabe von Standardwerten der Konfigurationsdaten steuert. Dies ist bei der Eröffnung einer Session wichtig, da eine zu große Datenmenge und die damit verbundenen notwendigen Operationen vermieden werden. Nähere Informationen sind in [3] zu finden.

C. Monitoring Schema

NETCONF definiert ein Schema, mit dem der Client sich über die Funktionalitäten des Servers informieren kann. Es beinhaltet alle Subtrees, also nicht nur die Capabilities, die Informationen liefern können und stellt diese mit Hilfe eines XSD Schemas dar. Mehr wird durch den Internet-Draft [4] spezifiziert.

V. YANG

Als ein Datenmodell abstrahiert YANG (Yet Another Next Generation) die eigentlichen Daten und bildet diese allgemeingültiger ab. Hier beschreibt es die Inhaltsebene der Architektur, also die Konfigurations- bzw. Statusdaten an sich, aber auch die RPCs und Notifications. Es füllt die Lücke, die vom NETCONF Protokoll offen gelassen wird und ermöglicht so eine standardisierte Sprache. Im Folgenden werden nur Ausschnitte aus allen Möglichkeiten, die Yang bietet präsentiert, da dies sonst zu weit führen würde. Weitere Informationen können in [5] nachgelesen werden.

A. Warum YANG?

Da bereits unzählige Datenmodellierungssprachen existieren, stellt sich die Frage, warum eine neue benötigt wird. Es gab einige Überlegungen XSD (XML Schema Definition) oder andere Sprachen zu erweitern, was aber nie wirklich zufrieden stellen konnte. Eines der größten Hindernisse für XSD war, dass es zu schwer zu erlernen und zu lesen ist. So wird es wie RelaxNG (Regular Language Description for XML New Generation) zu komplex und würde damit die Akzeptanz und somit den gesamten Erfolg gefährden. Zusammengefasst ist zu sagen, dass es bisher keine Sprache gibt, die alle Bedürfnisse, die durch die NETCONF Spezifikation entstehen, abdeckt. Weitere Punkte werden in [6] diskutiert.

B. Struktur

YANG arbeitet mit einer Aufteilung in Module und Submodule, welche hierarchisch gesehen die obersten Elemente bilden. Solche Module enthalten viele Informationen, vor allem aber die Moduldefinitionen, die den eigentlichen Inhalt bereitstellen und von denen einige hier vorgestellt werden.

1) Typen: YANG enthält einige Standarddatentypen, wie beispielsweise int32, wobei deren Wertebereich flexibel definiert werden kann (Listing 7).

Listing 7. type int32

```
type int32{
        range "1|1328..1337|max";
}
```

Mit Hilfe des *typedef* Befehls lassen sich aus den Stammtypen weitere Typen ableiten. In Listing 8 ist zum Beispiel der Code dargestellt, mit dem man Prozente realisieren kann.

Listing 8. typedef Prozent

```
typedef percent {
        type uint8 {
                range "0..100";
        }
        description "Percentage";
}
```

2) Leaf: Leafs stellen im Datenbaum die unterste Einheit dar und repräsentieren die Daten an sich. Man definiert also den Hintergrund der XML-tags, die in der tiefsten Ebene stehen. Im folgenden Listing 9 wird ein Element Port beschrieben.

Listing 9. leaf port

```
leaf port {
 type inet:port-number;
 default 22;
 description
 "The port which the SSH server listens to"
}
```

Im NETCONF Protokoll, könnte dieser Datenknoten dann wie folgt verwendet werden (Listing 10):

Listing 10. leaf port Einsatz

```
<rpc message-id="1337"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc=
 "urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <running/>
  </target>
  <config>
   <system
    xmlns="http://example.com/schema/config">
    <services>
     <ssh>
      <port>2022</port>
     </ssh>
    </services>
   </system>
  </config>
 </edit-config>
</rpc>
```

Darüber hinaus existiert noch die *leaf-list*, welche eine Liste gleichartiger Elemente beschreibt und mit einem Array von Werten verglichen werden kann.

3) Container: Container können leafs oder andere Container enthalten und bilden somit zusammenfassende Elemente. Die Arbeitsweise ist wieder an einem Beispiel verdeutlicht (Listing 11):

Listing 11. Container

```
container system {
 container services {
  container "ssh" {
   leaf port {
    type inet:port-number;
    default 22;
    description
    "The port which the SSH server listens to"
   }
  }
 }
}
```

Umgesetzt in XML ergibt sich Listing 12.

Listing 12. Container XML

```
<system>
 <services>
  <ssh>
   <port>2022</port>
   </ssh
 </services>
</system>
```

VI. Zusammenfassung und Ausblick

NETCONF versucht, eine bisher existierende Lücke zu füllen und arbeitet dabei mit flexiblen und dem Nativen zugewandten Methoden. Angestrebt wird dadurch eine breite Unterstützung von Netzwerkgeräten, wie sie in größeren Netzarchitekturen zu finden sind. YANG vervollständigt als Datenmodell das Protokoll und bietet so die notwendig Reife für eine Verbreitung.
Da NETCONF relativ einfach zu erlernen ist, besteht auf der Anwenderseite bereits ein großer Zuspruch, von dem noch mehr zu erwarten ist. Notwendig ist jetzt eine Vielzahl von Implementierungen auf Herstellerseite, damit der Gebrauch vermehrt in die Praxis übergehen kann.

Literatur

[1] http://www.ietf.org/dyn/wg/charter/netconf-charter.html
Stand 10.09.2009.

[2] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "NETCONF configuration protocol," Internet-Draft, draft-ietf-netconf-4741bis-01, July 2009.

[3] A. Bierman and B. Lengyel, "With-defaults capability for NETCONF," Internet-Draft, draft-ietf-netconf-with-defaults-03, August 2009.

[4] M. Scott, M. Bjorklund, and S. Chisholm, “NETCONF monitoring schema,” Internet-Draft, draft-ietf-netconf-monitoring-07, July 2009.
[5] M. Bjorklund, “YANG - a data modeling language for NETCONF,” Internet-Draft, draft-bjorklund-netconf-yang-02, February 2008.
[6] B. Lengyel, “Why we need a NETCONF-specific modeling language - YANG,” Internet-Draft, draft-lengyel-why-yang-00, November 2007.