Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 21, 2012

V. Perelman
J. Schoenwaelder
Jacobs University
M. Ersue
Nokia Siemens Networks
K. Watsen
Juniper Networks
January 18, 2012

Network Configuration Protocol Light (NETCONF Light)
draft-schoenw-netconf-light-01.txt

Abstract

This document describes a profile of the NETCONF protocol called NETCONF Light. This profile modularizes the NETCONF protocol and allows devices to announce that they only support a subset of the NETCONF protocol operations. This is useful in situations where devices are either too resource constrained to support all NETCONF operations or where devices are gradually updated from proprietary configuration interfaces to support NETCONF.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 21, 2012.

Table of Contents

1. Introduction 3
2. Motivation 4
2.1. NETCONF on Constrained Devices 4
2.2. Gradually Adding NETCONF Support 4
3. NETCONF Light Overview 6
3.1. Reduced Protocol Operations 6
3.1.1. <get-config> 6
3.1.2. <edit-config> 7
3.1.3. <copy-config> 7
3.1.4. <delete-config> 7
3.1.5. <lock> and <unlock> 7
3.1.6. <get> 8
3.1.7. <close-session> 8
3.1.8. <kill-session> 8
3.2. Capability Negotiation 8
4. NETCONF Light YANG Module 10
5. IANA Consideration 14
6. Security Considerations 15
7. References 16
7.1. Normative References 16
7.2. Informative References 16
Appendix A. NETCONF Light on AVR Raven / Contiki 17
Appendix B. Experience at Juniper Networks 18

1. Introduction

The Network Configuration Protocol (NETCONF) [RFC6241] provides mechanisms to install, manipulate, and delete the configuration of network devices. This document modularizes the NETCONF protocol and allows devices to announce that they only support a subset of the NETCONF protocol operations. This is useful in situations where devices are either too resource constrained to support all NETCONF operations or where devices are gradually updated from proprietary configuration interfaces to support NETCONF.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Motivation

This section explains the motiviation for NETCONF Light.

2.1. NETCONF on Constrained Devices

The original target of NETCONF were network devices such as routers or switches that usually have plenty of resources for running a NETCONF server. However, there are a number of embedded systems where resources (most notably memory) are tight and hence such devices can only afford a subset of the NETCONF protocol operations. This document allows constrained devices to implement a subset of NETCONF and to communicate that subset to NETCONF management applications in an interoperable way.

The usage of NETCONF Light on resource constrained devices is attractive in environments where management applications have to deal with a wide range of different devices, for example ranging from very small embedded networked sensors over more powerful data aggregation servers up to highly complex control networks. Typical examples are Smart Grids or more general industrial control networks.

Constrained devices can be classified according to the memory they have. A recently proposed classification is the following:

- Class 0: too small to securely run on the Internet (too constrained).
- Class 1: about 10 KiB of data and 100 KiB of code (quite constrained, 10/100)
- Class 2: about 50 KiB of data and 250 KiB of code (not so constrained, 50/250)

According to these classes, NETCONF Light should be running fine in "not so constrained" Class 2 devices and it may be running in "quite constrained" Class 1 devices, with very little resources left for other application code.

2.2. Gradually Adding NETCONF Support

While the NETCONF protocol defines a number of capabilities that may be optionally implemented, the base protocol remains a significant effort to add for existing devices. For these devices, adding support for NETCONF is primarily driven by a specific integration target, thus the intrinsic goal is to have an initial release that satisfies the integration target and a subsequent release that implements the remainder of the NETCONF protocol.

Some scenarios where phasing in the imeplmentation would be helpful include:

- The device's primary goal is to implement a vendor-specific capability. In this case, the device is only using NETCONF for its "Messages" layer (i.e. RPC, RPC-reply, and Notification).

- The device's primary goal is to just support read-only access to its configuration. In this case, it only needs to implement <get-config> initially, leaving the remaining operations for a future release.

- The device's primary goal is to enable full configuration, but it doesn't have the time to implement all the <edit-config> operations. In this case, the device could implement just <copy-config>.

- The device's primary goal is to enable full configuration, but it is unable to implement <lock> or <unlock> due to its platform not having a locking mechanism yet.

Each of these cases is satisfied by NETCONF Light, as the device can adverise the specific subset of NETCONF operations it supports. The ability for development teams to incrementally implement NETCONF makes it a more appealing target for their short-term efforts.

3. NETCONF Light Overview

NETCONF Light uses the NETCONF message framing as defined in [RFC6241]. In particular, it uses the same XML encoding and XML namespace.

The NETCONF specification [RFC6241] defines a set of base operations and a number of optional capabilities. A NETCONF Light implementation may choose to not support all NETCONF base operations. The set of operations supported by a NETCONF Light server is announced to a NETCONF client as features, see the definition of the ietf-netconf-light YANG [RFC6020] module in Section 4.

A NETCONF Light implementation, like any NETCONF implementation, does not have to support any of the optional NETCONF capabilities. The normal NETCONF rules apply for the capability exchange with <hello> messages.

A NETCONF Light implementation may support only a limited number of concurrent sessions. On some devices, the number of concurrent sessions might be as low as one. A NETCONF Light implementation supporting only a limited number of sessions should reject the establishment of a new transport, i.e., it should not even start the NETCONF <hello> exchange.

3.1. Reduced Protocol Operations

The following sections describe the changes to the NETCONF base protocol operations.

3.1.1. <get-config>

A NETCONF Light implementation MAY choose to not support the <get-config> operation as defined in Section 7.1 of [RFC6241]. Furthermore, a NETCONF Light implementation MAY choose to support <get-config> without filtering. An implementation not supporting the <get-config> operation MUST return an <rpc-error> element with an <error-tag> value of "operation-not-supported" when an <edit-config> operation is invoked. If a <get-config> operation is invoked with a <filter> element and filtering is not supported, an <rpc-error> element MUST be returned with an <error-tag> value of "unknown-element".

Note that [RFC6241] only requires to support the <running> datastore as source parameter.

3.1.2. <edit-config>

A NETCONF Light implementation supporting only a small data model MAY choose to not support the <edit-config> operation defined in Section 7.2 of [RFC6241]. An implementation not supporting the <edit-config> operation MUST return an <rpc-error> element with an <error-tag> value of "operation-not-supported" when an <edit-config> operation is invoked.

3.1.3. <copy-config>

A NETCONF Light implementation MAY choose to not support the <copy-config> as defined in Section 7.3 of [RFC6241]. An implementation not supporting the <copy-config> operation MUST return an <rpc-error> element with an <error-tag> value of "operation-not-supported" when a <copy-config> operation is invoked.

Note that [RFC6241] only requires to support the <running> datastore as source parameter. If no other capabilities are announced, the source parameter of the <copy-config> operation will carry the <config> element containing the complete configuration to copy.

3.1.4. <delete-config>

A NETCONF Light implementation MAY choose to not support the <delete-config> operation as defined in Section 7.4 of [RFC6241]. An implementation not supporting the <delete-config> operation MUST return an <rpc-error> element with an <error-tag> value of "operation-not-supported" when a <delete-config> operation is invoked.

Note that NETCONF implementations only supporting the <running> datastore can trivially implement <delete-config> by always returning a suitable <rpc-error> since the <running> datastore cannot be deleted.

3.1.5. <lock> and <unlock>

A NETCONF Light implementation MAY choose to not support the <lock> and <unlock> operations as defined in Sections 7.5 and 7.6 of [RFC6241]. An implementation not supporting the <lock> operation or the <unlock> operation MUST return an <rpc-error> element with an <error-tag> value of "operation-not-supported" when a <lock> operation is invoked.

3.1.6. <get>

A NETCONF Light implementations MAY choose to not support the <get> operation as defined in Section 7.7 of [RFC6241]. An implementation not supporting the <get> operation MUST return an <rpc-error> element with an <error-tag> value of "operation-not-supported" when a <get> operation is invoked.

Some implementations MAY choose to support the <get> operation with the following restriction: A NETCONF Light implementation MAY choose to not support filtering. If a <get> operation is invoked with a <filter> element and filtering is not supported, an <rpc-error> element MUST be returned with an <error-tag> value of "unknown-element".

3.1.7. <close-session>

A NETCONF Light implementation MAY choose to not support the <close-session> operation as defined in Section 7.8 of [RFC6241]. An implementation not supporting the <close-session> operation MUST return an <rpc-error> element with an <error-tag> value of "operation-not-supported" when a <close-session> operation is invoked.

3.1.8. <kill-session>

A NETCONF Light implementation MAY choose to not support the <kill-session> operation as defined in Section 7.9 of [RFC6241]. An implementation not supporting the <kill-session> operation MUST return an <rpc-error> element with an <error-tag> value of "operation-not-supported" when a <kill-session> operation is invoked.

3.2. Capability Negotiation

NETCONF advertises the capabilities during the <hello> exchange (see Section 8.1 of [RFC6241]). The NETCONF base capability, "urn:ietf:params:netconf:base:1.1", indicates that the NETCONF peer supports all the base protocol operations. Since this is not the case for NETCONF Light implementations, a NETCONF Light peer MUST NOT announce the NETCONF base capability and instead announce the NETCONF light capability.

In the following example, a NETCONF Light server advertises the NETCONF Light capability and support for the <get-config> and <copy-config> operation (whitespace has been added for readability).

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>
      urn:ietf:params:xml:ns:yang:ietf-netconf-light?
        module=ietf-netconf-light&amp;
        revision=2012-01-12&amp;
        features=get-config,copy-config
    </capability>
  </capabilities>
  <session-id>4</session-id>
</hello>
```

4. NETCONF Light YANG Module

This section defines the ietf-netconf-light YANG [RFC6020] module.

```
<CODE BEGINS> file "ietf-netconf-light@2012-01-12.yang"

module ietf-netconf-light {

  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-light";
  prefix "ncl";

  organization
   "IETF NETCONF (Network Configuration Protocol) Working Group";

  contact
   "WG Web:   <http://tools.ietf.org/wg/netconf/>
    WG List:  <mailto:netconf@ietf.org>

    WG Chair: Bert Wijnen
              <mailto:bertietf@bwijnen.net>

    WG Chair: Mehmet Ersue
              <mailto:mehmet.ersue@nsn.com>

    Editor: Vladislav Perelman
            <mailto:v.perelman@jacobs-university.de>

    Editor: Juergen Schoenwaelder
            <mailto:j.schoenwaelder@jacobs-university.de>

    Editor: Mehmet Ersue
            <mailto:mehmet.ersue@nsn.com>

    Editor: Kent Watsen
            <mailto:kwatsen@juniper.net>";

  description

   "This module defines the base NETCONF protocol defined in RFC 6241
    as a suite of optionally implemented features.  The NETCONF Light
    capability is expected to be advertized in the server's <hello>
    message in lieu of the traditional base NETCONF capability.  By
    advertizing this capability, servers can indentify which parts of
    the NETCONF protocol are supported.  For the most part, NETCONF
    Light defines a one-to-one mapping between base protocol
    operations and features enabling them; exceptions include
    <get-config>, which has one feature to enable the RPC and another
    to enable subtree-filtering, and <lock>/<unlock>, which share a
```

```
      feature called \"locking\".  Advertizing all NETCONF Light
      features is equivalent to advertizing the NETCONF base capability
      itself.

      Copyright (c) 2012 IETF Trust and the persons identified as
      authors of the code. All rights reserved.

      Redistribution and use in source and binary forms, with or
      without modification, is permitted pursuant to, and subject
      to the license terms contained in, the Simplified BSD
      License set forth in Section 4.c of the IETF Trust's
      Legal Provisions Relating to IETF Documents
      (http://trustee.ietf.org/license-info).

      This version of this YANG module is part of RFC XXXX; see
      the RFC itself for full legal notices.";
    // RFC Ed.: replace XXXX with actual RFC number and
    // remove this note

    // RFC Ed.: remove this note
    // Note: extracted from draft-schoenw-netconf-light-01.txt

    // RFC Ed.: please update the date to the date of publication

    revision 2012-01-12 {
      description "Initial version.";
      reference "RFC XXXX: Network Configuration Protocol for
                           Constrained Devices (NETCONF Light)";
    }
    // RFC Ed.: replace XXXX with actual
    // RFC number and remove this note

    feature get-config {
      description
       "This feature indicates that the server supports the
        <get-config> protocol operation, albeit without subtree
        filtering.   The server must additionally advertize
        the \"subtree-filtering\" feature to enable subtree
        filtering.  Alternatively, if the server only wants
        to support XPath filtering, it may just advertize
        the :xpath capability.";
    }

    feature subtree-filtering {
      description
       "This feature indicates that the server supports subtree
        filtering for the <get-config> operation.  This
        feature is only meaningful if the "get-config" feature
```

```
        is advertized; if "get-config" is not also advertized,
        this feature MUST be ignored.";
    }

    feature edit-config {
      description
       "This feature indicates that the server supports the
        <edit-config> protocol operation.  If the server is
        unable to support all the <edit-config> attributes
        (merge, replace, create, delete, remove), then it
        should advertize the \"copy-config\" feature instead.";
    }

    feature copy-config {
      description
       "This feature indicates that the server supports the
        <copy-config> protocol operation.";
    }

    feature delete-config {
      description
       "This feature indicates that the server supports the
        <delete-config> protocol operation.";
    }

    feature locking {
      description
       "This feature indicates that the server supports the
        <lock> and <unlock> protocol operations.";
    }

    feature get {
      description
       "This feature indicates that the server supports the
        <get> protocol operation.";
    }

    feature close-session {
      description
       "This feature indicates that the server supports the
        <close-session> protocol operation.  When this feature
        is not advertized, clients are expected to close the
        underlying transport directly.";
    }

    feature kill-session {
      description
       "This feature indicates that the server supports the
```

```
        <kill-session> protocol operation.";
    }
  }
```

<CODE ENDS>

5. IANA Consideration

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested.

URI: urn:ietf:params:xml:ns:yang:ietf-netconf-light

Registrant Contact: The NETCONF WG of the IETF.

XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [RFC6020].

```
name:         ietf-netconf-light
namespace:    urn:ietf:params:xml:ns:yang:ietf-netconf-light
prefix:       ncl
reference:    RFC XXXX
```

6. Security Considerations

NETCONF requires every implementation to support the SSH transport (Section 2.3 of [RFC6241]). On resource constrained devices, it is crucial that a single security protocol can be shared between different application protocols. While SSH tends to be popular for remote login services, it seems that TLS [RFC5246] and its datagram cousin DTLS [RFC4347] are enjoying much greater support on small embedded devices. Hence it might be necessary to choose a different mandatory to implement secure transport protocol for NETCONF Light.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

7.2. Informative References

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

Appendix A. NETCONF Light on AVR Raven / Contiki

An implementation of NETCONF Light on Contiki operating system has been created. It is running on AVR Raven motes, which are Class 1 devices. The implementation is compliant with this Internet-Draft. It does not support filtering, <edit-config> or any other optional NETCONF capabilities. NETCONF messages are currently transported over plain TCP connections.

Together with the Contiki operating system (which weighs about 10 KiB RAM) and the System Manager application (0.4 KiB RAM), which is used for retrieval of the operational state of the device, NETCONF Light takes 13 KiB RAM out of 16 KiB RAM available. The operating system together with the NETCONF Light implementation uses 78 KiB out of 128 KiB flash memory. This means that the current implementation of the protocol itself takes 2.6 KiB RAM - a value, that can be lowered by further code optimizations. 12 KiB out of the used 78 KiB of flash memory are reserved for the four files in the Coffee File System. These files are used for input / output manipulations in order to avoid using more RAM than needed. The size of the files can be changed if needed, however, it is not advisable to make the files larger since this will constrain usage of the flash memory by other applications. After installing NETCONF Light the device has 3.5 KiB of RAM free, which can be used by other applications.

Appendix B. Experience at Juniper Networks

Following are three case studies where NETCONF Light would have helped.

As a disclaimer, please note that in accordance with the RFC, Juniper does not claim its devices implement NETCONF or let them listen on port 830 until the entire NETCONF protocol has been implemented.

- Juniper Networks' device management strategy depends on NETCONF or, more precisely, on NETCONF plus capabilities we've defined to support various aspects of the device (system, hardware, software, licenses, etc.). In order to integrate with Juniper's NMS system, a device must support a number of these capabilities before the NMS will even attempt to configure the device. Thus it is common that devices new to Juniper, either OEM'ed from a partner or obtained through acquisistion, to initially support just the RPCs needed for the most basic support by Juniper's NMS and then implement the rest of the NETCONF protocol in a subsequent release.

- As an extension to the previous example, it's not uncommon for a device to intially support configuration using its native configuration format, which is typically not XML. However, since much of NETCONF still applies (getting/setting/locking datastores, etc), some NETCONF operations are extended to allow passing the device's native configuration format. Specifically, the <copy-config> RPC and <get-config> RPC-reply are extended to support passing an opaque payload. Naturally, subtree-filtering is disabled for the <get-config> operation.

- One of the devices that Juniper acquired has no notion of locking. The goal of the acquisition was to import the device's technology into Junos, but customers owning the original device wanted to continue using their existing hardware. However, since it was deemed impossible to add NETCONF to this device (not enough resources), the NSM team was forced to develop a device adapter that could mediate between the NETCONF interface the NMS system requires and the CLI interface the device provided. The adapter project was successful with one exception, the adapter had to implement the <lock>/<unlock> RPCs as null operations that blindly returned <ok>.

Authors' Addresses

Vladislav Perelman
Jacobs University Bremen

EMail: v.perelman@jacobs-university.de

Juergen Schoenwaelder
Jacobs University Bremen

EMail: j.schoenwaelder@jacobs-university.de

Mehmet Ersue
Nokia Siemens Networks

EMail: mehmet.ersue@nsn.com

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net