Internet Engineering Task Force (IETF)
Request for Comments: 7758
Category: Experimental
ISSN: 2070-1721

T. Mizrahi
Y. Moses
Technion

February 2016

Time Capability in NETCONF

Abstract

This document defines a capability-based extension to the Network Configuration Protocol (NETCONF) that allows time-triggered configuration and management operations. This extension allows NETCONF clients to invoke configuration updates according to scheduled times and allows NETCONF servers to attach timestamps to the data they send to NETCONF clients.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet community. This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7758.

Table of Contents

1. Introduction

The Network Configuration Protocol (NETCONF), defined in [RFC6241], provides mechanisms to install, manipulate, and delete the configuration of network devices. NETCONF allows clients to configure and monitor NETCONF servers using remote procedure calls (RPCs).

NETCONF is asynchronous; when a client invokes an RPC, it has no control over the time at which the RPC is executed, nor does it have any feedback from the server about the execution time.

Time-based configuration ([OneClock] [Time4]) can be a useful tool that enables an entire class of coordinated and scheduled configuration procedures. Time-triggered configuration allows coordinated network updates in multiple devices; a client can invoke a coordinated configuration change by sending RPCs to multiple servers with the same scheduled execution time. A client can also invoke a time-based sequence of updates by sending n RPCs with n different update times, T1, T2, ..., Tn, determining the order in which the RPCs are executed.

This memo defines the :time capability in NETCONF. This extension allows clients to determine the scheduled execution time of RPCs they send. It also allows a server that receives an RPC to report its actual execution time to the client.

The NETCONF time capability is intended for scheduling RPCs that should be performed in the near future, allowing the coordination of simultaneous configuration changes or specification of an order of configuration updates. Time-of-day-based policies and far-future scheduling, e.g., [Cond], are outside the scope of this memo.

This memo is defined for experimental purposes and will allow the community to experiment with the NETCONF time capability. Based on the lessons learned from this experience, it is expected that the NETCONF working group will be able to consider whether to adopt the time capability.

2. Conventions Used in This Document

2.1. Key Words

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 [RFC2119].

2.2. Abbreviations

NETCONF Network Configuration Protocol

RPC Remote Procedure Call

2.3. Terminology

- Capability [RFC6241]: A functionality that supplements the base NETCONF specification.

- Client [RFC6241]: Invokes protocol operations on a server. In addition, a client can subscribe to receive notifications from a server.

- Execution time: The execution time of an RPC is defined as the time at which a server completes the execution of an RPC, before it sends the <rpc-reply> message.

- Scheduled RPC: an RPC that is scheduled to be performed at a predetermined time, which is included in the <rpc> message.

- Scheduled time: The scheduled time of an RPC is the time at which the RPC should be started, as determined by the client. It is the server's role to enforce the execution of the scheduled time.

- Server [RFC6241]: Executes protocol operations invoked by a client. In addition, a server can send notifications to a client.

3. Using Time in NETCONF

3.1. The Time Capability in a Nutshell

The :time capability provides two main functions:

- Scheduling:

  When a client sends an RPC to a server, the <rpc> message MAY include the scheduled-time element, denoted by Ts in Figure 1. The server then executes the RPC at the scheduled time Ts; once completed, the server can respond with an RPC reply message.

o Reporting:

When a client sends an RPC to a server, the <rpc> message MAY include a get-time element (see Figure 2), requesting the server to return the execution time of the RPC. In this case, after the server performs the RPC, it responds with an RPC reply that includes the execution time, Te.

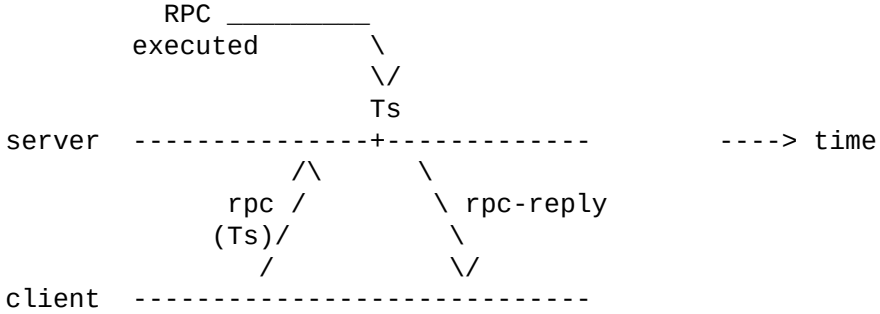

Figure 1: Scheduled RPC

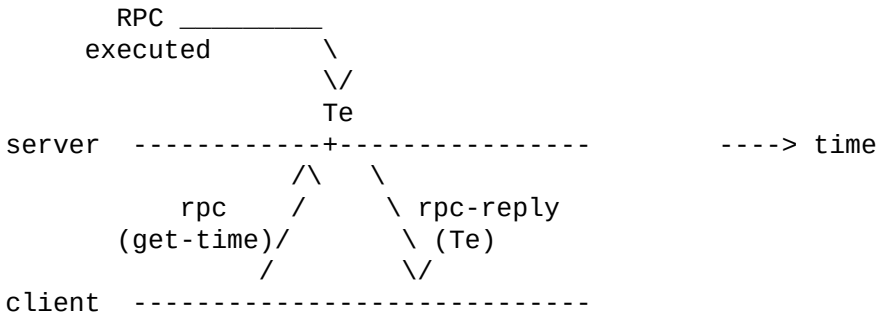

Figure 2: Reporting the Execution Time of an RPC

Example 1. A client needs to trigger a commit at n servers, so that the n servers perform the commit as close as possible to simultaneously. Without the time capability, the client sends a sequence of n commit messages; thus, each server performs the commit at a different time. By using the time capability, the client can send commit messages that are scheduled to take place at a chosen time Ts, for example, 5 seconds in the future, causing the servers to invoke the commit as close as possible to time Ts.

Example 2. In many applications, it is desirable to monitor events or collect statistics regarding a common time reference. A client can send a set of get-config messages that is scheduled to be executed at multiple servers at the same time, providing a

simultaneous system-wide view of the state of the servers. Moreover, a client can use the get-time element in its get-config messages, providing a time reference to the sampled element.

The scenarios of Figures 1 and 2 imply that a third scenario can also be supported (Figure 3), where the client invokes an RPC that includes a scheduled time, Ts, as well as the get-time element. This allows the client to receive feedback about the actual execution time, Te. Ideally, Ts=Te. However, the server may execute the RPC at a slightly different time than Ts, for example, if the server is tied up with other tasks at Ts.

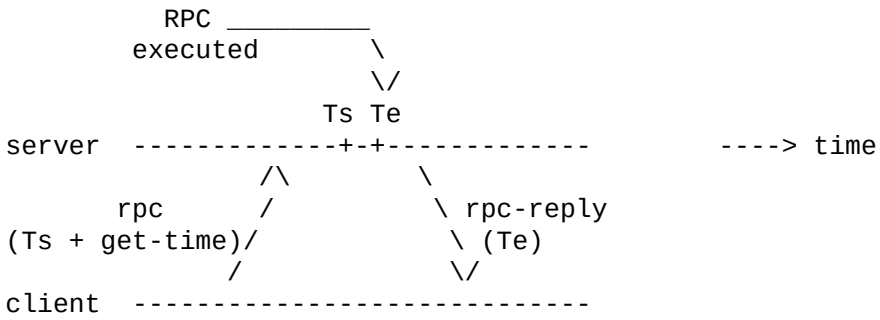

Figure 3: Scheduling and Reporting

3.2. Notifications and Cancellation Messages

Notifications

As illustrated in Figure 1, after a scheduled RPC is executed, the server sends an <rpc-reply>. The <rpc-reply> may arrive a long period of time after the RPC was sent by the client, leaving the client without a clear indication of whether the RPC was received.

This document defines a new notification, the netconf-scheduled-message notification, which provides an immediate acknowledgement of the scheduled RPC.

The <netconf-scheduled-message> notification is sent to the client if it is subscribed to the NETCONF notifications [RFC6470]; as illustrated in Figure 4, when the server receives a scheduled RPC, it sends a notification to the client.

The <netconf-scheduled-message> notification includes a <schedule-id> element. The <schedule-id> is a unique identifier that the server assigns to every scheduled RPC it receives. Thus, a client can keep track of all the pending scheduled RPCs; a client can uniquely identify a scheduled RPC by the tuple {server, schedule-id}.

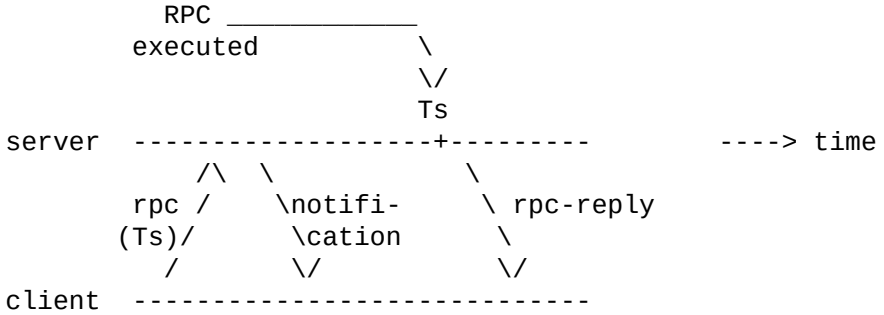

Figure 4: Scheduled RPC with Notification

Cancellation Messages

A client can cancel a scheduled RPC by sending a <cancel-schedule> RPC. The <cancel-schedule> RPC includes the <schedule-id> of the scheduled RPC that needs to be cancelled.

The <cancel-schedule> RPC, defined in this document, can be used to perform a coordinated all-or-none procedure, where either all the servers perform the operation on schedule or the operation is aborted.

Example 3. A client sends scheduled <rpc> messages to server 1 and server 2, both scheduled to be performed at time Ts. Server 1 sends a notification indicating that it has successfully scheduled the RPC, while server 2 replies with an unknown-element error [RFC6241] that indicates that it does not support the time capability. The client sends a <cancel-schedule> RPC to server 1 and receives an <rpc-reply>. The message exchange between the client and server 1 in this example is illustrated in Figure 5.

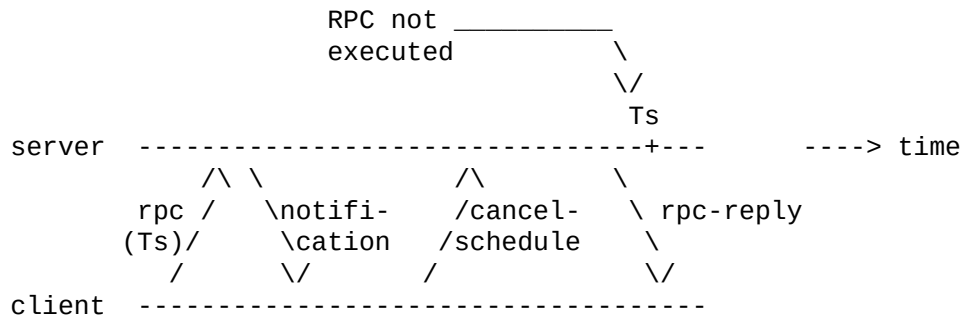

Figure 5: Cancellation Message

A <cancel-schedule> RPC MUST NOT include the scheduled-time parameter. A server that receives a <cancel-schedule> RPC should try to cancel the schedule as soon as possible. If the server is unable to cancel the scheduled RPC, for example, because it has already been executed, it should respond with an <rpc-error> [RFC6241], in which the error-type is 'protocol', and the error-tag is 'operation-failed'.

3.3. Synchronization Aspects

The time capability defined in this document requires clients and servers to maintain clocks. It is assumed that clocks are synchronized by a method that is outside the scope of this document, e.g., [RFC5905] or [IEEE1588].

This document does not define any requirements pertaining to the degree of accuracy of performing scheduled RPCs. Note that two factors affect how accurately the server can perform a scheduled RPC: one factor is the accuracy of the clock synchronization method used to synchronize the clients and servers and the second factor is the server's ability to execute real-time configuration changes, which greatly depends on how it is implemented. Typical networking devices are implemented by a combination of hardware and software. While the execution time of a hardware module can typically be predicted with a high level of accuracy, the execution time of a software module may be variable and hard to predict. A configuration update would typically require the server's software to be involved, thus affecting how accurately the RPC can be scheduled.

Another important aspect of synchronization is monitoring; a client should be able to check whether a server is synchronized to a reference time source. Typical synchronization protocols, such as the Network Time Protocol [RFC5905], provide the means ([RFC5907], [RFC7317]) to verify that a clock is synchronized to a time reference by querying its Management Information Base (MIB). The get-time

feature defined in this document (see Figure 2) allows a client to obtain a rough estimate of the time offset between the client's clock and the server's clock.

Since servers do not perform configuration changes instantaneously, the processing time of an RPC should not be overlooked. The scheduled time always refers to the start time of the RPC, and the execution time always refers to its completion time.

3.4. Scheduled Time Format

The scheduled time and execution time fields in <rpc> messages use a common time format field.

The time format used in this document is the date-and-time format, defined in Section 5.6 of [RFC3339] and Section 3 of [RFC6991].

```
leaf scheduled-time {
  type yang:date-and-time;
  description
  "The time at which the RPC is scheduled to be performed.";
}

leaf execution-time {
  type yang:date-and-time;
  description
  "The time at which the RPC was executed.";
}
```

3.5. Scheduling Tolerance

When a client sends an RPC that is scheduled to Ts, the server MUST verify that the value Ts is not too far in the past or in the future. As illustrated in Figure 6, the server verifies that Ts is within the scheduling-tolerance range.

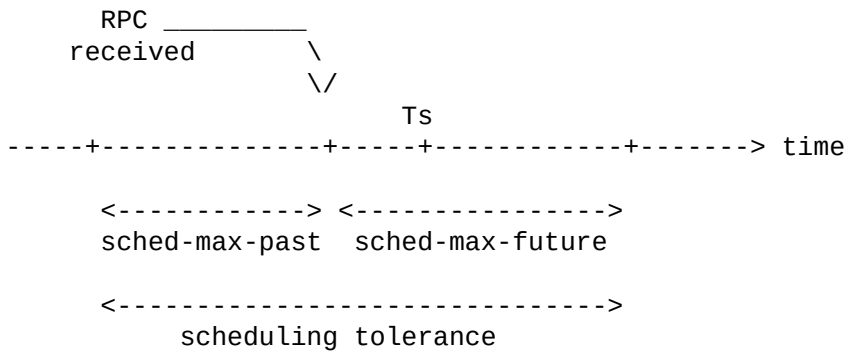

Figure 6: Scheduling Tolerance

The scheduling tolerance is determined by two parameters: sched-max-future and sched-max-past. These two parameters use the time-interval format (Section 3.7.), and their default value is 15 seconds.

If the scheduled time, Ts, is within the scheduling-tolerance range, the scheduled RPC is performed; if Ts occurs in the past and within the scheduling tolerance, the server performs the RPC as soon as possible; whereas if Ts is a future time, the server performs the RPC at Ts.

If Ts is not within the scheduling-tolerance range, the scheduled RPC is discarded, and the server responds with an error message [RFC6241] including a bad-element error-tag. An example is provided in Section 5.3.

3.6. Scheduling the Near vs. Far Future

The scheduling bound defined by sched-max-future guarantees that every scheduled RPC is restricted to a scheduling time in the near future.

The scheduling mechanism defined in this document is intended for near-future scheduling, on the order of seconds. Far-future scheduling is outside the scope of this document.

Example 1 is a typical example of using near-future scheduling; the goal in the example is to perform the RPC at multiple servers at the same time; therefore, it is best to schedule the RPC to be performed a few seconds in the future.

The Challenges of Far-Future Scheduling

When an RPC is scheduled to be performed at a far-future time, during the long period between the time at which the RPC is sent and the time at which it is scheduled to be executed, the following erroneous events may occur:

- The server may restart.
- The client's authorization level may be changed.
- The client may restart and send a conflicting RPC.
- A different client may send a conflicting RPC.

In these cases, if the server performs the scheduled operation, it may perform an action that is inconsistent with the current network policy or inconsistent with the currently active clients.

Near-future scheduling guarantees that external events, such as the examples above, have a low probability of occurring during the sched-max-future period, and even when they do, the period of inconsistency is limited to sched-max-future, which is a short period of time.

The Trade-off in Setting the sched-max-future Value

The sched-max-future parameter should be configured to a value that is high enough to allow the client to:

1. Send the scheduled RPC, potentially to multiple servers.
2. Receive notifications or <rpc-error> messages from the server(s) or wait for a timeout and decide that if no response has arrived then something is wrong.
3. If necessary, send a cancellation message, potentially to multiple servers.

On the other hand, sched-max-future should be configured to a value that is low enough to allow a low probability of the erroneous events above, typically on the order of a few seconds. Note that, even if sched-max-future is configured to a low value, it is still possible (with a low probability) that an erroneous event will occur. However, this short, potentially hazardous period is not significantly worse than in conventional (unscheduled) RPCs, as even a conventional RPC may in some cases be executed a few seconds after it was sent by the client.

The Default Value of sched-max-future

The default value of sched-max-future is defined to be 15 seconds. This duration is long enough to allow the scheduled RPC to be sent by the client, potentially to multiple servers, and in some cases to send a cancellation message, as described in Section 3.2. On the other hand, the 15-second duration yields a very low probability of a reboot or a permission change.

3.7. Time-Interval Format

The time-interval format is used for representing the length of a time interval and is based on the date-and-time format. It is used for representing the scheduling tolerance parameters, as described in the previous section.

While the date-and-time type uniquely represents a specific point in time, the time-interval type defined below can be used to represent the length of a time interval without specifying a specific date.

The time-interval type is defined as follows:

```
typedef time-interval {
  type string {
    pattern '\d{2}:\d{2}:\d{2}(\.\d+)?';
  }
  description
    "Defines a time interval, up to 24 hours.
     The format is specified as HH:mm:ss.f,
     consisting of two digits for hours,
     two digits for minutes, two digits
     for seconds, and zero or more digits
     representing second fractions.";
}
```

Example

The sched-max-future parameter is defined (Appendix A) as a time-interval, as follows:

```
leaf sched-max-future {
  type time-interval;
  default 00:00:15.0;
}
```

The default value specified for sched-max-future is 0 hours, 0 minutes, and 15 seconds.

4. Time Capability

The structure of this section is as defined in Appendix D of [RFC6241].

4.1. Overview

A server that supports the time capability can perform time-triggered operations as defined in this document.

A server implementing the :time capability:

- MUST support the ability to receive <rpc> messages that include a time element and perform a time-triggered operation accordingly.
- MUST support the ability to include a time element in the <rpc-reply> messages that it transmits.

4.2. Dependencies

With-defaults Capability

The time-capability YANG module (Appendix A) uses default values; thus, it is assumed that the with-defaults capability [RFC6243] is supported.

4.3. Capability Identifier

The :time capability is identified by the following capability string:

urn:ietf:params:netconf:capability:time:1.0

4.4. New Operations

<cancel-schedule>

The <cancel-schedule> RPC is used for cancelling an RPC that was previously scheduled.

A <cancel-schedule> RPC MUST include the <cancelled-message-id> element, which specifies the message ID of the scheduled RPC that needs to be cancelled.

A <cancel-schedule> RPC MAY include the <get-time> element. In this case, the <rpc-reply> includes the <execution-time> element, specifying the time at which the scheduled RPC was cancelled.

4.5. Modifications to Existing Operations

4.5.1. Affected Operations

The :time capability defined in this memo can be applied to any of the following operations:

- get-config
- get
- copy-config
- edit-config
- delete-config
- lock
- unlock
- commit

Three new elements are added to each of these operations:

- <scheduled-time> This element is added to the input of each operation, indicating the time at which the server is scheduled to invoke the operation. Every <rpc> message MAY include the <scheduled-time> element. A server that supports the :time capability and receives an <rpc> message with a <scheduled-time> element MUST perform the operation as close as possible to the scheduled time.

  The <scheduled-time> element uses the date-and-time format (Section 3.4.).

- <get-time> This element is added to the input of each operation. An <rpc> message MAY include a <get-time> element, indicating that the server MUST include an <execution-time> element in its corresponding <rpc-reply>.

- <execution-time> This element is added to the output of each operation, indicating the time at which the server completed the operation. An <rpc-reply> MAY include the <execution-time> element. A server that supports the :time capability and receives an operation with the <get-time> element MUST include the execution time in its response.

The <execution-time> element uses the date-and-time format (Section 3.4.).

4.5.2. Processing Scheduled Operations

A server that receives a scheduled RPC MUST start executing the RPC as close as possible to its scheduled execution time.

If a session between a client and a server is terminated, the server MUST cancel all pending scheduled RPCs that were received in this session.

Scheduled RPCs are processed serially, in an order that is defined by their scheduled times. Thus, the server sends <rpc-reply> messages to scheduled RPCs according to the order of their corresponding schedules. Note that this is a modification to the behavior defined in [RFC6241], which states that replies are sent in the order the requests were received. Interoperability with [RFC6241] is guaranteed by the NETCONF capability exchange; a server that does not support the :time capability responds to RPCs in the order the requests were received. A server that supports the :time capability replies to conventional (non-scheduled) RPCs in the order they were received and replies to scheduled RPCs in the order of their scheduled times.

If a server receives two or more RPCs that are scheduled to be performed at the same time, the server executes the RPCs serially in an arbitrary order.

4.6. Interactions with Other Capabilities

Confirmed Commit Capability

The confirmed commit capability is defined in Section 8.4 of [RFC6241]. According to that document, a confirmed <commit> operation MUST be reverted if a confirming commit is not issued within the timeout period (which is 600 seconds by default).

When the time capability is supported, and a confirmed <commit> operation is used with the <scheduled-time> element, the confirmation timeout MUST be counted from the scheduled time, i.e., the client begins the timeout measurement starting at the scheduled time.

5. Examples

5.1. <scheduled-time> Example

The following example extends the example presented in Section 7.2 of [RFC6241] by adding the time capability. In this example, the <scheduled-time> element is used to specify the scheduled execution time of the configuration update (as shown in Figure 1).

```
<rpc message-id="101"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <scheduled-time
       xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-time">
        2015-10-21T04:29:00.235Z
    </scheduled-time>
    <config>
      <top xmlns="http://example.com/schema/1.2/config">
        <interface>
          <name>Ethernet0/0</name>
          <mtu>1500</mtu>
        </interface>
      </top>
    </config>
  </edit-config>
</rpc>

<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

5.2. <get-time> Example

The following example is similar to the one presented in Section 5.1, except that, in this example, the client includes a <get-time> element in its RPC and the server consequently responds with an <execution-time> element (as shown in Figure 2).

```
<rpc message-id="101"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <get-time
     xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-time">
    </get-time>
    <config>
      <top xmlns="http://example.com/schema/1.2/config">
        <interface>
          <name>Ethernet0/0</name>
          <mtu>1500</mtu>
        </interface>
      </top>
    </config>
  </edit-config>
</rpc>

<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
  <execution-time>
      2015-10-21T04:29:00.235Z
  </execution-time>
</rpc-reply>
```

5.3. Error Example

The following example presents a scenario in which the scheduled-time is not within the scheduling tolerance, i.e., it is too far in the past; therefore, an <rpc-error> is returned.

```
<rpc message-id="101"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <scheduled-time
       xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-time">
        2010-10-21T04:29:00.235Z
    </scheduled-time>
    <config>
      <top xmlns="http://example.com/schema/1.2/config">
        <interface>
          <name>Ethernet0/0</name>
          <mtu>1500</mtu>
        </interface>
      </top>
    </config>
  </edit-config>
</rpc>

<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>bad-element</error-tag>
    <error-severity>error</error-severity>
    <error-info>
      <bad-element>scheduled-time</bad-element>
    </error-info>
  </rpc-error>
</rpc-reply>
```

6. Security Considerations

6.1. General Security Considerations

The security considerations of the NETCONF protocol in general are discussed in [RFC6241].

The usage of the time capability defined in this document can assist an attacker in gathering information about the system, such as the exact time of future configuration changes. Moreover, the time elements can potentially allow an attacker to learn information about the system's performance. Furthermore, an attacker that sends malicious <rpc> messages can use the time capability to amplify her attack; for example, by sending multiple <rpc> messages with the same scheduled time. It is important to note that the security measures described in [RFC6241] can prevent these vulnerabilities.

The time capability relies on an underlying time synchronization protocol. Thus, by attacking the time protocol, an attack can potentially compromise NETCONF when using the time capability. A detailed discussion about the threats against time protocols and how to mitigate them is presented in [RFC7384].

The time capability can allow an attacker to attack a NETCONF server by sending malicious RPCs that are scheduled to take place in the future. For example, an attacker can send multiple scheduled RPCs that are scheduled to be performed at the same time. Another possible attack is to send a large number of scheduled RPCs to a NETCONF server, potentially causing the server's buffers to overflow. These attacks can be mitigated by a carefully designed NETCONF server; when a server receives a scheduled RPC that exceeds its currently available resources, it should reply with an <rpc-error> and discard the scheduled RPC.

Note that if an attacker has been detected and revoked, its future scheduled RPCs are not executed; as defined in Section 4.5.2, once the session with the attacker has been terminated, the corresponding scheduled RPCs are discarded.

6.2. YANG Module Security Considerations

This memo defines a new YANG module, as specified in Appendix A.

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is Secure SHell (SSH) [RFC6242]. The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF users to a preconfigured subset of all available NETCONF protocol operations and content.

This YANG module defines <sched-max-future> and <sched-max-past>, which are writable/creatable/deletable. These data nodes may be considered sensitive or vulnerable in some network environments. An attacker may attempt to maliciously configure these parameters to a

low value, thereby causing all scheduled RPCs to be discarded. For instance, if a client expects <sched-max-future> to be 15 seconds, but in practice it is maliciously configured to 1 second, then a legitimate scheduled RPC that is scheduled to be performed 5 seconds in the future will be discarded by the server.

This YANG module defines the <cancel-schedule> RPC. This RPC may be considered sensitive or vulnerable in some network environments. Since the value of the <schedule-id> is known to all the clients that are subscribed to notifications from the server, the <cancel-schedule> RPC may be used maliciously to attack servers by cancelling their pending RPCs. This attack is addressed in two layers: (i) security at the transport layer, limiting the attack only to clients that have successfully initiated a secure session with the server, and (ii) the authorization level required to cancel an RPC should be the same as the level required to schedule it, limiting the attack only to attackers with an authorization level that is equal to or higher than that of the client that initiated the scheduled RPC.

7. IANA Considerations

The following capability identifier URN has been registered in the "Network Configuration Protocol (NETCONF) Capability URNs" registry:

urn:ietf:params:netconf:capability:time:1.0

The following XML namespace URN has been registered in the "IETF XML Registry", following the format defined in [RFC3688]:

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

Registrant Contact: The IESG.

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

The following module name has been registered in the "YANG Module Names" registry, defined in [RFC6020].

name: ietf-netconf-time

prefix: nct

namespace: urn:ietf:params:xml:ns:yang:ietf-netconf-time

RFC: 7758

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <http://www.rfc-editor.org/info/rfc3339>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6470] Bierman, A., "Network Configuration Protocol (NETCONF) Base Notifications", RFC 6470, DOI 10.17487/RFC6470, February 2012, <http://www.rfc-editor.org/info/rfc6470>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://www.rfc-editor.org/info/rfc6991>.

8.2. Informative References

[Cond] Watsen, K., "Conditional Enablement of Configuration Nodes", draft-kwatsen-conditional-enablement-00, Work in Progress, February 2013.

[IEEE1588] IEEE, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems Version 2", IEEE Standard 1588.

[OneClock] Mizrahi, T. and Y. Moses, "OneClock to Rule Them All: Using Time in Networked Applications", IEEE/IFIP Network Operations and Management Symposium (NOMS), 2016.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <http://www.rfc-editor.org/info/rfc5905>.

[RFC5907] Gerstung, H., Elliott, C., and B. Haberman, Ed., "Definitions of Managed Objects for Network Time Protocol Version 4 (NTPv4)", RFC 5907, DOI 10.17487/RFC5907, June 2010, <http://www.rfc-editor.org/info/rfc5907>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <http://www.rfc-editor.org/info/rfc6242>.

[RFC6243] Bierman, A. and B. Lengyel, "With-defaults Capability for NETCONF", RFC 6243, DOI 10.17487/RFC6243, June 2011, <http://www.rfc-editor.org/info/rfc6243>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <http://www.rfc-editor.org/info/rfc7317>.

[RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", RFC 7384, DOI 10.17487/RFC7384, October 2014, <http://www.rfc-editor.org/info/rfc7384>.

[Time4] Mizrahi, T. and Y. Moses, "Software Defined Networks: It’s About Time", IEEE INFOCOM, 2016.

Appendix A. YANG Module for the Time Capability

This section is normative.

```
<CODE BEGINS> file "ietf-netconf-time@2016-01-26.yang"

module ietf-netconf-time {

   namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-time";

   prefix nct;
   import ietf-netconf { prefix nc; }

   import ietf-yang-types { prefix yang; }

   import ietf-netconf-monitoring { prefix ncm; }

   organization
     "IETF";

   contact
     "Editor: Tal Mizrahi
        <dew@tx.technion.ac.il>
      Editor: Yoram Moses
        <moses@ee.technion.ac.il>";

   description
     "This module defines a capability-based extension to the
      Network Configuration Protocol (NETCONF) that allows
      time-triggered configuration and management operations.
      This extension allows NETCONF clients to invoke configuration
      updates according to scheduled times and allows NETCONF
      servers to attach timestamps to the data they send to NETCONF
      clients.

      Copyright (c) 2016 IETF Trust and the persons identified as
      the 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).";

   revision 2016-01-26 {
     description
       "Initial version.";
```

```
    reference
      "RFC 7758:
       Time Capability in NETCONF";
  }

  typedef time-interval {
    type string {
      pattern '\d{2}:\d{2}:\d{2}(\.\d+)?';
    }
    description
      "Defines a time interval, up to 24 hours.
       The format is specified as HH:mm:ss.f,
       consisting of two digits for hours,
       two digits for minutes, two digits
       for seconds, and zero or more digits
       representing second fractions.";
  }

  grouping scheduling-tolerance-parameters {
    leaf sched-max-future {
      type time-interval;
      default 00:00:15.0;
      description
        "When the scheduled time is in the future, i.e., greater
         than the present time, this leaf defines the maximal
         difference between the scheduled time
         and the present time that the server is willing to
         accept.  If the difference exceeds this number, the
         server responds with an error.";
    }

    leaf sched-max-past {
      type time-interval;
      default 00:00:15.0;
      description
        "When the scheduled time is in the past, i.e., less
         than the present time, this leaf defines the maximal
         difference between the present time
         and the scheduled time that the server is willing to
         accept.  If the difference exceeds this number, the
         server responds with an error.";
    }

    description
      "Contains the parameters of the scheduling tolerance.";
  }
  // extending the get-config operation
  augment /nc:get-config/nc:input {
```

```
    leaf scheduled-time {
      type yang:date-and-time;
      description
        "The time at which the RPC is scheduled to be performed.";
    }

    leaf get-time {
      type empty;
      description
        "Indicates that the rpc-reply should include the
         execution-time.";
    }

    description
      "Adds the time element to <get-config>.";
  }

  augment /nc:get-config/nc:output {
    leaf execution-time {
      type yang:date-and-time;
      description
        "The time at which the RPC was executed.";
    }

    description
      "Adds the time element to <get-config>.";
  }

  augment /nc:get/nc:input {
    leaf scheduled-time {
      type yang:date-and-time;
      description
        "The time at which the RPC is scheduled to be performed.";
    }

    leaf get-time {
      type empty;
      description
        "Indicates that the rpc-reply should include the
         execution-time.";
    }

    description
      "Adds the time element to <get>.";
  }

  augment /nc:get/nc:output {
    leaf execution-time {
```

```
      type yang:date-and-time;
      description
        "The time at which the RPC was executed.";
    }

    description
      "Adds the time element to <get>.";
  }

  augment /nc:copy-config/nc:input {
    leaf scheduled-time {
      type yang:date-and-time;
      description
        "The time at which the RPC is scheduled to be performed.";
    }

    leaf get-time {
      type empty;
      description
        "Indicates that the rpc-reply should include the
         execution-time.";
    }

    description
      "Adds the time element to <copy-config>.";
  }

  augment /nc:copy-config/nc:output {
    leaf execution-time {
      type yang:date-and-time;
      description
        "The time at which the RPC was executed.";
    }

    description
      "Adds the time element to <copy-config>.";
  }

  augment /nc:edit-config/nc:input {
    leaf scheduled-time {
      type yang:date-and-time;
      description
        "The time at which the RPC is scheduled to be performed.";
    }

    leaf get-time {
      type empty;
      description
```

```
        "Indicates that the rpc-reply should include the
         execution-time.";
    }

    description
      "Adds the time element to <edit-config>.";
  }

  augment /nc:edit-config/nc:output {
    leaf execution-time {
      type yang:date-and-time;
      description
        "The time at which the RPC was executed.";
    }

    description
      "Adds the time element to <edit-config>.";
  }

  augment /nc:delete-config/nc:input {
    leaf scheduled-time {
      type yang:date-and-time;
      description
        "The time at which the RPC is scheduled to be performed.";
    }

    leaf get-time {
      type empty;
      description
        "Indicates that the rpc-reply should include the
         execution-time.";
    }

    description
     "Adds the time element to <delete-config>.";
  }

  augment /nc:delete-config/nc:output {
    leaf execution-time {
      type yang:date-and-time;
      description
        "The time at which the RPC was executed.";
    }
    description
      "Adds the time element to <delete-config>.";
  }

  augment /nc:lock/nc:input {
```

```
    leaf scheduled-time {
      type yang:date-and-time;
      description
        "The time at which the RPC is scheduled to be performed.";
    }

    leaf get-time {
      type empty;
      description
        "Indicates that the rpc-reply should include the
         execution-time.";
    }

    description
      "Adds the time element to <lock>.";
  }
  augment /nc:lock/nc:output {
    leaf execution-time {
      type yang:date-and-time;
      description
        "The time at which the RPC was executed.";
    }

    description
      "Adds the time element to <lock>.";
  }

  augment /nc:unlock/nc:input {
    leaf scheduled-time {
      type yang:date-and-time;
      description
        "The time at which the RPC is scheduled to be performed.";
    }

    leaf get-time {
      type empty;
      description
        "Indicates that the rpc-reply should include the
         execution-time.";
    }

    description
      "Adds the time element to <unlock>.";
  }

  augment /nc:unlock/nc:output {
    leaf execution-time {
      type yang:date-and-time;
```

```
      description
        "The time at which the RPC was executed.";
    }

    description
      "Adds the time element to <unlock>.";
  }
  augment /nc:commit/nc:input {
    leaf scheduled-time {
      type yang:date-and-time;
      description
        "The time at which the RPC is scheduled to be performed.";
    }

    leaf get-time {
      type empty;
      description
        "Indicates that the rpc-reply should include the
         execution-time.";
    }

    description
      "Adds the time element to <commit>.";
  }

  augment /nc:commit/nc:output {
    leaf execution-time {
      type yang:date-and-time;
      description
        "The time at which the RPC was executed.";
    }

    description
      "Adds the time element to <commit>.";
  }

  augment /ncm:netconf-state {
    container scheduling-tolerance {
      uses scheduling-tolerance-parameters;
      description
        "The scheduling tolerance when the time capability
         is enabled.";
    }
    description
      "The scheduling tolerance of the server.";
  }

  rpc cancel-schedule {
```

```
     description
       "Cancels a scheduled message.";
     reference
       "RFC 7758:
        Time Capability in NETCONF";

     input {
       leaf cancelled-message-id {
         type string;
         description
           "The ID of the message to be cancelled.";
       }
       leaf get-time {
         type empty;
         description
           "Indicates that the rpc-reply should include
            the execution-time.";
       }
     }
     output {
       leaf execution-time {
         type yang:date-and-time;
         description
           "The time at which the RPC was executed.";
       }
     }
   }

   notification netconf-scheduled-message {
     leaf schedule-id {
       type string;
       description
         "The ID of the scheduled message.";
     }

     leaf scheduled-time {
       type yang:date-and-time;
       description
         "The time at which the RPC is scheduled to be performed.";
     }
     description
       "Indicates that a scheduled message was received.";
     reference
       "RFC 7758:
        Time Capability in NETCONF";
   }

}
```

<CODE ENDS>

Acknowledgments

The authors gratefully acknowledge Joe Marcus Clarke, Andy Bierman, Balazs Lengyel, Jonathan Hansford, John Heasley, Robert Sparks, Al Morton, Olafur Gudmundsson, Juergen Schoenwaelder, Joel Jaeggli, Alon Schneider, and Eylon Egozi for their insightful comments.

This work was supported in part by Israel Science Foundation grant ISF 1520/11.

Authors' Addresses

Tal Mizrahi
Department of Electrical Engineering
Technion - Israel Institute of Technology
Technion City, Haifa, 32000
Israel

Email: dew@tx.technion.ac.il

Yoram Moses
Department of Electrical Engineering
Technion - Israel Institute of Technology
Technion City, Haifa, 32000
Israel

Email: moses@ee.technion.ac.il