Proyecto Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Estudio de la gestión de configuración a través de NETCONF

Autor: Miguel Merelo Hernández

Tutor: Dr. Antonio J. Estepa Alonso

Dep. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

Proyecto Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Estudio de la gestión de configuración a través de NETCONF

Autor:
Miguel Merelo Hernández

Tutor:
Dr. Antonio J. Estepa Alonso
Profesor Titular

Dep. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

Resumen

En este trabajo se realizará un estudio sobre el protocolo NETCONF indicando las opciones que ofrece. Se analizará la inclusión del protocolo en el mercado, las herramientas existentes y se realizará una prueba de concepto utilizando el agente ConfD Basic.

Abstract

In this work you will find a NETCONF protocol study emphasizing the benefits it can provide. The implementation of the protocol in the network devices will be analized as the tools to work with NETCONF. A proof of concept will be made using the ConfD Basic server.

Índice

Índice de Figuras

Índice de Tablas

1 Introducción

Vivimos en un mundo hiperconectado en el que la interacción entre personas, empresas o gobiernos se realiza a través de la comunicación. Esta comunicación se sustenta sobre redes de datos.

Una mayor interacción genera la necesidad de aumentar el tamaño de las redes para satisfacer la demanda. Esto provoca la existencia de un mayor número de nodos en la red, los cuales deben configurarse para trabajar como se desea. Los procesos realizados para configurar los nodos se denomina gestión de la configuración.

La gestión de la configuración se encarga de indicar al equipo cómo debe trabajar con la información que maneja, ya sea la generada por el propio equipo o la recibida de otro dispositivo.

Actualmente, la gestión de configuración se realiza mediante CLI, interfaz web y SNMP[1]. CLI (Command-Line Interface) es el método más potente de acceso al equipo pero implica la necesidad de conocer el sistema del dispositivo de red que estemos gestionando. Cada fabricante o gama de productos puede implementar unos comandos distintos para realizar la misma configuración, lo que requiere un alto conocimiento del dispositivo. La interfaz web ofrece un acceso a la configuración mucho más visual a cambio de un menor acceso a las opciones de configuración. Por último, SNMP es un protocolo que responde a la necesidad de crear un modelo unificado de gestión de configuración y permite que la misma configuración, con los mismos datos, funcione en todos los dispositivos de red simplificando esta tarea. A pesar de que, también, se utilice para la gestión de la configuración, SNMP es generalista, no está enfocado a este fin.

NETCONF[2], a diferencia de SNMP, es un protocolo para realizar la gestión de la configuración exclusivamente. Una de las particularidades de NETCONF es que define un método de configuración de múltiples equipos con capacidad para probar la nueva configuración y decidir si mantener los cambios o revertirlos, esta característica es idónea para la gestión de las redes grandes y para permitir su escalado.

El objeto de este trabajo será el estudio del protocolo de gestión de configuración NETCONF, de su filosofía, modelos de datos y estructura así como de las herramientas, tanto privativas como libres, para su uso y de un análisis de la implementación del protocolo en los equipos de los principales fabricantes, comparándolo con otros protocolos de gestión de configuración existentes. Por último, se realizará una prueba de concepto con alguna de las herramientas estudiadas.

2 NETCONF

La definición que nos proporciona la RFC-6241[2] sobre el protocolo es que "NETCONF define un mecanismo simple a través del cual un dispositivo de red puede ser gestionado". En otras palabras, NETCONF es un protocolo de gestión de configuración. A diferencia de otros protocolos generalistas, como puede ser SNMP, NETCONF se centra en exclusiva en la gestión de configuración.

NETCONF nace con la función de suplir ciertos aspectos que no se tratan en otros protocolos de gestión como puede ser la posibilidad de aplicar cambios de configuración a varios elementos de la red simultáneamente o definir un método de recuperación en caso de un fallo en la configuración, pérdida de conexión o similar. Estos aspectos y varios más serán tratados en este capítulo así como la arquitectura del mismo, el sistema de filtrado de datos y la implementación del protocolo en los dispositivos de red disponibles en el mercado.

2.1 Objetivo

Durante el año 2001, se dieron una serie de reuniones, entre operadores de red y desarrolladores de protocolos, en las que se abordaba los problemas que tenían los protocolos de gestión y las necesidades que no suplían. Estas reuniones desembocaron en un taller de trabajo organizado por IAB(Internet Architecture Board) en junio de 2002.

Tomando como referencia las conclusiones recogidas en RFC-3535[3], correspondientes a dicho taller de trabajo, podemos destacar ciertas funciones correspondientes a la gestión de configuración que deberían implementarse en un protocolo de esta índole.

Un protocolo de gestión de configuración debe hacer una clara distinción entre los datos de configuración, los de operación y las estadísticas que almacena el dispositivo. NETCONF define los datos de configuración como aquella información necesaria para que el dispositivo pase de su estado inicial al estado actual mientras que los datos de operación y estadística los define como aquellos datos que se generan mientras que el dispositivo está trabajando. Para ayudar a esta separación define dos métodos de lectura de los datos, el método <get-config> exclusivo para los datos de configuración y el método <get> que puede ser utilizado para obtener cualquier dato del dispositivo, incluidos los de configuración.

Otro de los puntos importantes de la gestión de configuración es la robustez en el control de acceso. Se pueden producir modificaciones simultáneas en la configuración de un equipo en dos aspectos distintos y que estos interfieran entre sí y debe existir un mecanismo para evitar este problema. Para ello, NETCONF, define que las peticiones que llegan al dispositivo (mensajes de modificación o lectura de parámetros) deben procesarse y responderse en orden de llegada siguiendo un mecanismo FIFO (First In, First Out). NETCONF permite que existan varias sesiones abiertas simultáneamente y que una de las sesiones bloquee el cambio de la configuración de las otras sesiones. Además, permite que, desde una sesión, se bloquee la modificación de algunos campos de configuración. [4]

En la gestión de configuración se debe hacer una diferenciación clara entre la distribución de la configuración y la aplicación de la misma. Esta función permite modificar el comportamiento de un conjunto de nodos simultáneamente logrando así disminuir el tiempo en el cual la red no se encuentra operativa. Adicionalmente a esto se debe considerar la posibilidad de incorporar un sistema de back-up para que, en caso de la perdida de conectividad por la nueva configuración, se pueda regresar a la situación anterior y volver a tener acceso a los nodos de la red. Con este objetivo, NETCONF define un mecanismo en el cual se genera un punto de recuperación antes de implementar la nueva configuración, tras ello se implementa esta configuración y se prueba. Si la configuración es correcta se puede fijar como permanente, en caso contrario se vuelve al punto de recuperación. Para poder realizar todo esto NETCONF define la existencia de múltiples almacenes de datos de configuración y una serie de primitivas.

Otras funciones útiles en la gestión de configuración es la posibilidad de monitorizar el acceso y los cambios que se producen en un dispositivo. Para ello, se define un modelo de datos [5] que permite realizarlo al igual que otros dos modelo de datos RFC-5277 [6] RFC-6470 [7] para las notificaciones de eventos, siguiendo un sistema publicador-suscriptor. La diferencia entre ambos es que el primero permite monitorizar apartados del protocolo y de su implementación, como los almacenes de datos o las sesiones y los otros dos se centran en monitorizar aspectos como cambios en la configuración y en estadísticos, contadores.

Como característica extra que podemos destacar de NETCONF es su sistema de filtrado. Ambos aspectos están directamente relacionados, el sistema de filtrado de NETCONF hace que podamos definir todo el conjunto de datos que queremos recuperar y que sigan un patrón determinado, por ejemplo, podemos recuperar toda la información de las tarjetas de red que tenga la máscara de red 255.255./16. Este filtrado lo procesa el servidor NETCONF y responde en consecuencia. Utilizando este sistema somos capaces de recuperar toda la información de configuración en una sola transacción, lo que se conoce como bulk transaction, necesario en caso de que queramos analizar toda la configuración de un dispositivo.

A continuación se muestra una tabla comparativa entre SNMP y NETCONF.

Tabla 2.1 Comparación de las necesidades cubiertas por SNMP y NETCONF.

	SNMP	NETCONF
Distingue entre datos de configuración y operación	NO	SI
Protección ante cambio de configuraciones simultáneos	NO	SI
Diferencia entre despliegue y activación de la configuración	NO	SI
Almacén de múltiples configuraciones	NO	SI
Monitorización de estadísticos	SI	SI
Monitorización de los cambios de configuración	NO	SI
Sistema de filtrado	NO	SI
Bulk transactions	SI	SI

SNMP es un protocolo de uso general, es decir, no está pensado específicamente para la gestión de configuración por lo que una comparación con NETCONF puede ser dudosa, aún así, SNMP se utiliza a día de hoy para la gestión de configuración con lo cual queda justificada la comparación entre ambos.

2.2 Arquitectura

NETCONF es un protocolo de la capa de aplicación, capa 7 (según modelo OSI) que debe trabajar sobre una capa de transporte que tiene que cumplir ciertos requisitos.

2.2.1 Capa de transporte

NETCONF delega el transporte y la seguridad a las capas inferiores. Estas capas deben encargarse de mantener las sesiones entre cliente y servidor y encargarse de la integridad y confidencialidad de la misma. La capa de transporte debe ofrecer un mecanismo que permita indicar el tipo de la sesión, cliente o servidor.

Un requisito para una implementación de NETCONF es que debe soportar el protocolo SSH. Esto queda definido en RFC-6242.

2.2.2 RPC y XML

NETCONF realiza el intercambio de información utilizando XML, crea una estructura de datos jerarquizada en la cual se encuentran todas las operaciones que deseamos realizar. La estructura del mensaje está basada en el mecanismo RPC. NETCONF utiliza RPC para realizar la comunicación entre cliente y servidor. Los mensajes RPC indican el tipo de mensaje que se está enviando, existen tres tipos, <rpc>, <rpc-reply> y <rpc-error>. De estos elementos se hablará más adelante en el punto 3 de este documento.
Los mensajes RPC contienen etiquetas de operación que van a definir lo que deseamos realizar, ya sea la modificación de la configuración, leer algún estadístico o recuperar algún dato. Por último, dentro de estas etiquetas, encontramos el contenido de estas operaciones.

2.2.3 Resumen

Hemos visto que NETCONF es un protocolo de aplicación entre servidor y cliente que solo define este intercambio de mensajes y delega la parte de transporte a capas inferiores como pueden ser SSH o TLS.

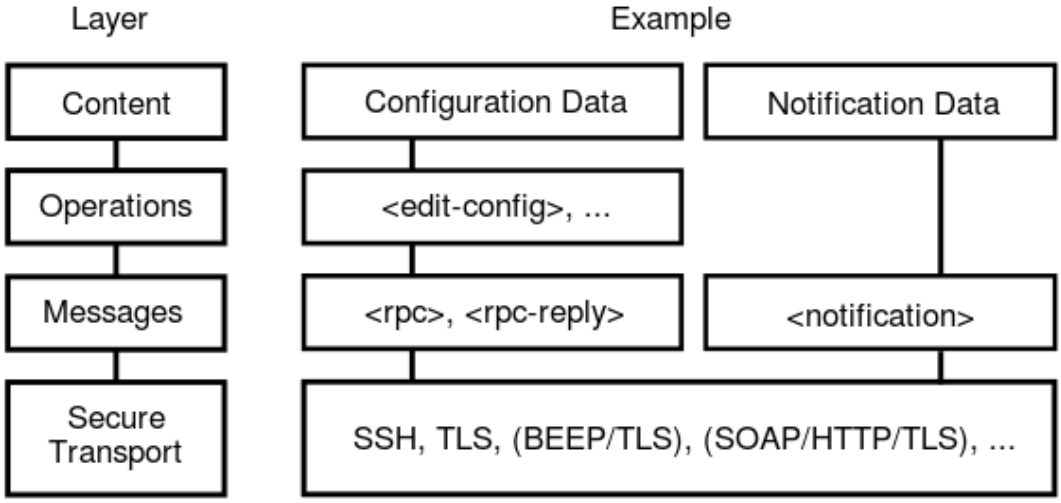

Figura 2.1 Arquitectura de NETCONF.

2.3 NETCONF en el mercado

2.3.1 Muestras

Con el objetivo de definir la situación actual en el que se encuentra NETCONF en el ámbito empresarial se han analizado los productos de varios de los principales fabricantes de enrutadores y conmutadores para ver la implementación de NETCONF en el mercado.
Para realizar esto se ha analizado la información que ofrecen los fabricantes de sus productos. La búsqueda se ha centrado en exclusiva en comprobar qué productos implementan NETCONF y qué productos implementan SNMP. Los fabricantes analizados son Alcatel-Lucent, Aruba, Cisco, Hewlett-Packard(HP), Juniper y Nokia. La información específica de cada fabricante se encuentra en el Apendice A de este documento.

2.3.2 Situación actual

De los fabricantes analizados solo encontramos NETCONF en algunos productos de Cisco y, en mayor medida, en los productos de Juniper y de Nokia mientras que SNMP lo encontramos en todos los productos

analizados de todos los fabricantes.

Una posible explicación de este hecho consiste en que la implementación de un nuevo protocolo requiere el desarrollo de un producto nuevo y que su incorporación al sistema operativo implica, para el equipo, un consumo de recursos mayor.

En el caso de Juniper, su sistema operativo, JunOS, ya incorporaba un sistema de configuración basado en XML y RPC lo cual puede explicar que la implementación de NETCONF, también basado en XML y RPC, esté más extendido en los productos de este fabricante. Todos los enrutadores de Juniper incorporan NETCONF o su sistema de gestión basado en XML. Respecto a los conmutadores, no encontramos NETCONF en ningún producto.

Respecto a Cisco, podemos encontrar una implementación mayor de NETCONF en su gama de conmutadores para data center y en los productos más recientes.

Nokia implementa NETCONF en cuatro de los seis productos de los que dispone.

2.3.3 Evolución

Juniper ha hecho una fuerte apuesta por NETCONF y parece que seguirá incorporando este protocolo en sus nuevos productos.

Cisco ha mostrado su interés en seguir incluyendo el protocolo en sus nuevos productos, como ha indicado Sachin Gupta, Vice President, Product Management for Cisco Switching, en el blog de cisco[8], consultado el 17/05/2017. En dicho artículo se muestra la necesidad de incorporar nuevos mecanismos de gestión que faciliten la tarea más allá de CLI.

Entre los que no implementan NETCONF podemos destacar a Alcatel-Lucent que sí tiene un sistema de gestión propio basado en XML y que puede ser la puerta a la futura implementación de NETCONF en sus productos.

3 Modelo de datos

Un agente y un cliente NETCONF se comunican enviando información codificada en XML. El modelo de comunicación que utilizan se basa en un sistema RPC (Remote Procedural Call).

Aunque en este capítulo vamos a hablar de YANG[9] como modelo de datos para NETCONF debido a su creación específica para ello y aceptación como estándar. NETCONF no define ningún modelo de datos para su uso.

3.1 XML

Los mensajes NETCONF enviados entre cliente y servidor se estructuran, como ya hemos dicho, en formato XML siguiendo la estructura (schema) definida en el Apéndice B de la RFC-6241[2]. Estos mensajes utilizan una codificación UTF-8.

Los mensajes recibidos en el servidor que no sigan el schema o codificación previstos serán rechazados, por el servidor, y se devolverá un mensaje de error al cliente.

3.2 Inicio de sesión

Una vez iniciada la conexión, a través de la capa de enlace, cliente y servidor se intercambian mensajes utilizando el elemento <hello>. Estos mensajes indican las Capabilities que implementa cada uno, (éstas serán comentadas en el punto 3.5). En el caso del servidor también envía el id de la sesión.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>
      urn:ietf:params:netconf:base:1.1
    </capability>
    <capability>
      urn:ietf:params:netconf:capability:startup:1.0
    </capability>
  </capabilities>
</hello>

<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>
      urn:ietf:params:netconf:base:1.1
    </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>
```

3.3 RPC

NETCONF utiliza un modelo de comunicación basado en RPC. La comunicación normal, correcta y sin errores, durante una sesión utilizará los elementos <rpc> y <rpc-reply>.

3.3.1 <rpc> y <rpc-reply>

El elemento <rpc> se utiliza para realizar acciones sobre el servidor que serán respondidas con un mensaje con el elemento <rpc-reply>.

Cada elemento <rpc> o <rpc-reply> contiene un atributo <message-id> que identifica el mensaje en esa conexión. Una petición <rpc> con un determinado <message-id> recibirá un <rpc-reply> con ese mismo valor.

Este ejemplo invoca la operación <get> sin parámetros.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:ex="http://example.net/content/1.0"
     ex:user-id="fred">
   <get/>
</rpc>

<rpc-reply message-id="101"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:ex="http://example.net/content/1.0"
    ex:user-id="fred">
  <data>
    <!-- contents here... -->
  </data>
</rpc-reply>
```

3.3.2 <rpc-error>

En caso de que ocurra algún error durante la ejecución en el servidor de una petición <rpc>, el elemento <rpc-reply> contendrá uno o varios elementos <rpc-error>.

Este mensaje <rpc-reply> puede contener varios elementos, <rpc-error>, describiendo cada uno de los errores que se hayan producido, mas solo es obligatorio que contenga uno de ellos.

El mensaje de error (rpc-error) contiene, a través de etiquetas, información de la capa en la que se ha producido el error (error-type), de la causa del error(error-tag) y de la importancia (error-severity). Adicionalmente existen etiquetas, error-app-tag, error-path, error-message y error-info que añaden información extra.

error-severity tiene dos posibles valores, error y warning, de momento solo se utiliza error y warning se reserva para un uso futuro.

En este ejemplo, el mensaje rpc se envía sin atributo message-id. La respuesta contiene información sobre este error.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
  </get-config>
</rpc>

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

La lista con los distintos errores existentes se puede encontrar en el Apéndice A de la RFC-6241[2].

3.3.3 <ok>

El elemento <ok> se incluye dentro de <rpc-reply> cuando el mensaje no tiene que enviar ningún dato de respuesta.

3.4 Operaciones

NETCONF define una serie de operaciones para la comunicación. Estas operaciones son un elemento incluidos dentro de un mensaje <rpc>.
Se definen nueve operaciones básicas para la gestión de la configuración que son get, get-config, edit-config, copy-config, delete-config, lock, unlock, close-session, kill-session.

3.4.1 <lock> y <unlock>

En el punto 2.1 de este documento se hizo mención a la posibilidad que existe en NETCONF de bloquear el acceso a la modificación de la configuraciones de otras sesiones; <lock> y <unlock> permiten realizar esta operación.

Para esta operación, el elemento <lock> define un parámetro obligatorio, <target>, que sirve para identificar el almacén de datos de configuración (datastore) que queremos bloquear. Este parámetro existe de la misma forma para el elemento <unlock> y su uso es similar.

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

```
</rpc>

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

3.4.2 <get-config>

Esta operación se utiliza para obtener toda o parte de la configuración almacenada.
Contiene dos elementos, el primero, <source>, hace referencia al almacén de datos de configuración al que nos referimos y el segundo, <filter>, se utiliza para indicar qué parte de la configuración queremos obtener. Si no existe el elemento <filter>, obtendremos toda la información del fichero.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
  </get-config>
</rpc>

<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
  <!-- datos del fichero -->
  </data>
</rpc-reply>
```

3.4.3 <edit-config>

Esta operación sirve para tratar con los parámetros de configuración de un determinado fichero de configuración.
Podemos realizar cinco acciones, indicadas dentro del atributo "operation". Las acciones son merge, replace, create, delete y remove. Delete y remove eliminan un parámetro del fichero de configuración, la diferencia es que si este parámetro no existe en el fichero, la respuesta contendrá un <error-tag> en el caso de delete. En el caso de remove, el servidor ignora esa parte y no genera un error. Si este atributo no se indica, se tomará la acción merge por defecto.
La acción merge reemplaza el elemento en caso de que exista en el fichero de configuración o lo añade al fichero de configuración si no existe. Las acciones antes previstas se realizan por cada elemento de manera individual.

Tabla 3.1 Resultado de la acción merge según el estado del almacén de datos.

OPERACIÓN	ALMACÉN DE DATOS (datastore)	RESULTADO
<interface> <name>Ethernet0/0</name> <mtu>1500</mtu> </interface>	<interface> </interface>	<interface> <name>Ethernet0/0</name> <mtu>1500</mtu> </interface>
<interface> <name>Ethernet0/0</name> <mtu>1500</mtu> </interface>	<interface> <name>Wifi0/0</name> <mtu>1000</mtu> </interface>	<interface> <name>Ethernet0/0</name> <mtu>1500</mtu> </interface>
<interface> <name>Ethernet0/0</name> <mtu>1500</mtu> </interface>	<interface> <name>Wifi0/0</name> </interface>	<interface> <name>Ethernet0/0</name> <mtu>1500</mtu> </interface>
<interface> <name>Ethernet0/0</name> </interface>	<interface> <name>Wifi0/0</name> <mtu>1500</mut> </interface>	<interface> <name>Ethernet0/0</name> <mtu>1500</mut> </interface>

En el siguiente caso realizamos la acción merge sobre el elemento interface. Modificamos name y la mtu.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <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>
```

3.4.4 <copy-config>

Crea o reemplaza un almacén de datos de configuración con el contenido de otro.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <copy-config>
    <target>
```

```
      <running/>
    </target>
    <source>
      <url>https://user:password@example.com/cfg/new.txt</url>
    </source>
  </copy-config>
</rpc>

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

3.4.5 <delete-config>

Elimina un almacén de datos de configuración.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <delete-config>
    <target>
      <startup/>
    </target>
  </delete-config>
</rpc>

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

3.4.6 <get>

Esta operación se utiliza para obtener información de estado del servidor y los datos de configuración actual del servidor. A diferencia de <get-config>, con <get> no podemos obtener datos de configuración de otros almacenes de datos de configuración.

En este ejemplo obtenemos los datos almacenados para la interfaz eth0.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <top xmlns="http://example.com/schema/1.2/stats">
        <interfaces>
          <interface>
            <ifName>eth0</ifName>
          </interface>
        </interfaces>
      </top>
    </filter>
  </get>
</rpc>
```

```
<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <top xmlns="http://example.com/schema/1.2/stats">
      <interfaces>
        <interface>
          <ifName>eth0</ifName>
          <ifInOctets>45621</ifInOctets>
          <ifOutOctets>774344</ifOutOctets>
        </interface>
      </interfaces>
    </top>
  </data>
</rpc-reply>
```

3.4.7 <close-session>

Operación para finalizar la sesión actual.

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

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

3.4.8 <kill-session>

Operación que fuerza la detención de una sesión. Esta sesión, indicada en el elemento <session-id>, no puede ser la actual o la respuesta será un error.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <kill-session>
    <session-id>4</session-id>
  </kill-session>
</rpc>

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

3.5 Capabilities

Las capabilities son un complemento extra que proporcionan mecanismos y características nuevas a los definidos por el protocolo. Deben ser anunciadas, por el servidor, al iniciar la sesión, durante la fase de intercambio de los mensajes <hello>.

NETCONF define unas capabilities que añaden nuevas operaciones. Se pueden destacar writeable-running capability, que permite modificar directamente el almacén de datos <running>, o Confirmed Commit Capability, que permite hacer cambios en la configuración para luego decidir si queremos mantenerlos o revertirlos.

Confirmed Commit Capability requiere de la existencia, y anuncio, de la Candidate Configuration Capability ya que toma de esta la operación <commit>.

Candidate Configuration Capability obliga a la existencia de un almacén de datos <candidate> en el cual podemos crear una configuración paralela a la que está actualmente en funcionamiento <running>. Al utilizar la operación <commit>, el dispositivo tomará la configuración del almacén <candidate> y la pondrá en funcionamiento. Confirmed Commit Capability añade parámetros a la operación <commit> para confirmar el cambio, también define una nueva operación <cancel-commit> para revertir los cambios y tomar la configuración inicial. Si tras hacer <commit>, con el parámetro <confirmed> no se envía otro mensaje <commit> en un tiempo preestablecido, el dispositivo volverá a la configuración previa.

El siguiente es un ejemplo que realiza un <commit> y se mantiene a la espera, durante 120 segundos, de un nuevo <commit> para mantener la configuración del almacén de datos <candidate>.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <commit>
    <confirmed/>
    <confirm-timeout>120</confirm-timeout>
  </commit>
</rpc>

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

Mensaje de confirmación de la configuración

```
<rpc message-id="102"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <commit/>
</rpc>

<rpc-reply message-id="102"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

En este caso se envía un mensaje que cancela los cambios y vuelve a la configuración previa.

```
<rpc message-id="102"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <cancel-commit/>
</rpc>

<rpc-reply message-id="102"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

3.6 YANG

YANG es un lenguaje de modelado de datos creado específicamente para la gestión de la configuración y los datos de estado de NETCONF definido en la RFC-7950[9].

3.6.1 Módulos

Los modelos de datos en YANG se definen en módulos. El módulo incorpora información sobre la versión de YANG utilizada, los datos del creador, información de la versión actual del módulo, descripción del módulo y el namespace que será el identificador del módulo dentro de una comunicación XML.

3.6.2 Importar módulos

YANG permite incorporar un módulo ya existente al nuestro para hacer uso del mismo, podremos decidir qué revisión del módulo querremos usar. Para ello haremos uso de la sentencia import, en la que se indicará el nombre del módulo, el prefijo con el que nos referimos a él en nuestro módulo y la fecha de revisión que vamos a utilizar.

```
import ejemplo{
  prefix "ej";
  revision-date 2017-05-29;
}
```

También podremos añadir nodos a un modulo importado a través de la sentencia augment.

3.6.3 Tipos de nodos

YANG define cuatro tipos de nodos, Leaf Node, Leaf-List Node, Container Node y List Node.

Leaf Node es un tipo de nodo que solo puede contener un dato, no pueden existir más de una instancia de este valor a su mismo nivel dentro del XML.

Leaf-List Node es equivalente a Leaf Node pero elimina la restricción de una instancia a su mismo nivel en el XML.

Container Node se utiliza para agrupar nodos relacionados. Puede contener tipo de nodo.

List Node permite definir una estructura de nodos y permite la existencia de n List Node similares.

Ejemplo adaptado a una clase:

```
module ejemplo{
  yang-version 1.1;
  namespace "urn:ejemplo";
  prefix "ejem";

  organization "US";
  contact "miguel@example.com";
  description
    "Modulo de ejemplo de una clase";

  revision 2017-05-29{
    description "Original";
```

```
  }
  container clase{
    config true;
    status current;
      container profesor{
      leaf nombre{
        type string
        description
          "Nombre del profesor";
      }
    }
     list alumno{
        key "nombre";
        leaf nombre{
          type string;
        description
          "Nombre del alumno";
        }
      container examenes{
        leaf-list calificacion{
          type decimal64;
          description
            "Calificaciones obtenidas";
        }
      }
    }
  }
}
```

Unos posibles valores en XML, señalando cual es cada Node, serían:

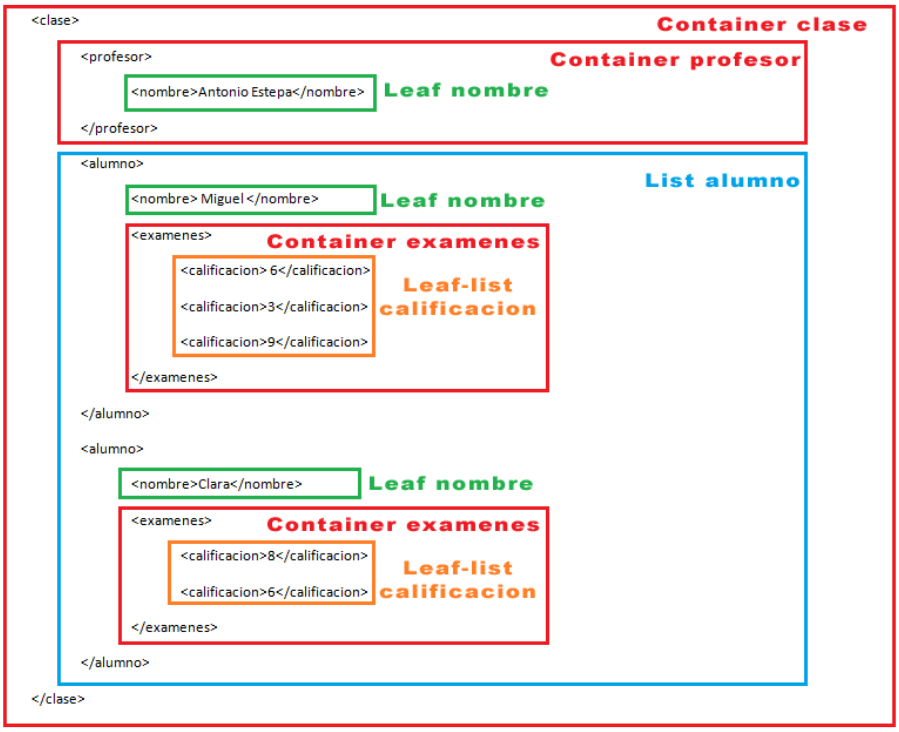

Figura 3.1 Ejemplo de valores en un XML a partir de un módulo de YANG.

A la hora de definir un nodo, podremos utilizar la sentencia config seguida de true o false para indicar si ese nodo es configurable o no. En caso de que el nodo esté marcado como config false, ese elemento será interpretado como dato de estado.

Si un nodo no incluye la sentencia config, obtendrá el valor de este parámetro de su nodo padre. Si ningún nodo tiene esta sentencia, se tomará por defecto el valor true. Un nodo hijo de un padre con config false nunca podrá ser configurable.

La declaración status indica si el estado del nodo, puede tomar los valores current, deprecated y obsolete. Por defecto toma el valor current.

3.6.4 Tipos de datos

YANG implementa una serie de tipos de datos por defecto, como pueden ser int8 o string, pero permite definir nuevos tipos a través de typedef.

Para definir un nuevo tipo con typedef debemos partir de un tipo por defecto u otro tipo ya creado y, a continuación, podremos realizar modificaciones sobre estos como, por ejemplo, limitar el rango de números válidos o añadir un valor por defecto al tipo. Estas modificaciones se realizan a través de las sentencias range y default, respectivamente.

```
typedef nuevo-tipo{
  type int8{
    range "0..10";
  }
  default 10;
}
```

3.6.5 Operaciones

YANG permite definir nuevas operaciones utilizando la sentencia rpc. Podremos indicar que parámetros contiene la operación y que parámetros se recibirán como respuesta. Esto se conseguirá mediante el uso, dentro de la sentencia rpc, de las sentencias input y output.

```
rpc operacion {
  input {
       leaf image-name {
      type string;
         }
       }
       output {
        leaf status {
             type string;
          }
       }
}
```

Se verá reflejado en los mensajes en XML como:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <operacion xmlns="http://example.com/system">
    <image-name>example-fw-2.3</image-name>
```

```
  </operacion>
</rpc>

<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <status xmlns="http://example.com/system">
    The image example-fw-2.3 is being installed.
  </status>
</rpc-reply>
```

3.6.6 Notificaciones

Para definir una nueva notificación, utilizaremos la sentencia notification.

```
notification link-failure {
  description
          "A link failure has been detected.";
        leaf if-name {
          type leafref {
               path "/interface/name";
            }
        }
        leaf if-admin-status {
          type admin-status;
        }
        leaf if-oper-status {
           type oper-status;
        }
}
```

4 Herramientas

Existen diversas herramientas relacionadas con NETCONF, tanto de carácter de software libre como de software propietario. En este capítulo se tratarán algunas herramientas para el desarrollo de módulos a través de YANG y de implementaciones de NETCONF, tanto de la parte de cliente como de la parte de servidor. También se mencionarán algunas librerías existentes disponibles para trabajar con NETCONF.

4.1 YANG

Para validar módulos YANG se disponen de diferentes aplicaciones. En su versión web tenemos yangvalidator (*http://yangvalidator.org/*) y yangdump (*http://www.netconfcentral.org/run_yangdump*).

A la hora de generar nuevos módulos podemos encontrar el plugin para Eclipse YANG IDE (*http://docs.opendaylight.org/en/stable-boron/getting-started-guide/project-specific-guides/yangide.html*), la versión actual, 1.1.1,no soporta YANG 1.1.

Por último, hay herramientas para resaltar ciertas palabras claves de YANG en Eclipse (*https://github.com/novakmi/mini/tree/master/yang*), Notepad++ (*https://github.com/vkosuri/lang-yang*) y emacs (*https://github.com/mbj4668/yang-mode*).

4.2 Clientes

Son varias las implementaciones de clientes tanto de software propietario como de software libre. Referido a las implementaciones de clientes NETCONF de software propietario encontramos las aplicaciones que ofrecen los fabricantes de dispositivos de red como CISCO o JUNIPER o la implementación de yumaworks, yangcli-pro (*https://www.yumaworks.com/downloads/*).

Entre las aplicaciones con licencias de software libre podemos destacar Netopeer-GUI (*https://github.com/CESNET/Netopeer-GUI*), una herramienta web para la gestión de NETCONF. Adicionalmente, existen implementaciones como ConfD Basic (*http://www.tail-f.com/confd-basic/*), Netopeer2 (*https://github.com/CESNET/Netopeer2*) o EnSuite (*http://ensuite.sourceforge.net/*) que incorporan tanto la parte de cliente como la de servidor y serán tratadas en el siguiente apartado. En el caso de Netopeer2 tiene un cliente basado en línea de comandos mientras que con el cliente de EnSuite se trabaja a través de un navegador.

4.3 Servidores

Como se ha mencionado en el apartado anterior, tenemos implementaciones en la parte de servidor en el proyecto Netopeer2, ConfD Basic y EnSuite que implementan la parte del cliente y del servidor. Además de los mencionados podemos encontrar OpenYuma (*https://github.com/OpenClovis/OpenYuma*) con licencia de software libre.

Entre las aplicaciones de software privativo encontramos YumaPro (*https://www.yumaworks.com/netconfd-pro/*). Este software se desligó del proyecto OpenYuma y ambas han seguido desarrollándose por caminos distintos.

4.4 Librerías

Muchas aplicaciones de clientes y servidores NETCONF implementan alguna librería de validación de módulos. Entre éstas podemos encontrar libyang (*https://github.com/CESNET/libyang*) escrita en C o pyang (*https://github.com/mbj4668/pyang*) escrita en Python. Ambas con licencias de software libre, BSD-3 e ISC, respectivamente.

La mayoría de librerías existentes están enfocadas para la implementación de clientes NETCONF. Las hay en Java, JNC (*https://github.com/tail-f-systems/JNC*), en Python, ncclient (*https://pypi.python.org/pypi/ncclient*) o Junos PyEZ (*https://github.com/Juniper/py-junos-eznc*) y para Android, netconf4android (*https://code.google.com/archive/p/netconf4android/*).

También encontramos librerías para la implementación de servidores, y clientes, escritas en C, como libnetconf (*https://github.com/CESNET/libnetconf*).

Por último, también se debe mencionar la existencia de sysrepo (*https://github.com/sysrepo/sysrepo*), un gestor de almacenes de datos, escrito en C. Este gestor lo utiliza, por ejemplo, la implementación del servidor NETCONF Netopeer.

5 Prueba de concepto

En este capítulo se pondrá en práctica los conocimientos teóricos aportados en este trabajo. Se creará un nuevo módulo, se incorporará a un servidor NETCONF y realizaremos distintas consultas y modificaciones de su configuración. Para ello, haremos uso de distintas herramientas mencionadas en el capítulo anterior. Haremos uso de yangdump para la validación del módulo YANG, utilizaremos ConfD Basic para la implementación del agente NETCONF y se usará yangcli-pro y el cliente de ConfD Basic para hacer pruebas, netconf-console-tcp.

El cliente de ConfD Basic que vamos a utilizar tiene ciertas peculiaridades de las que nos podremos beneficiar. No es una implementación de NETCONF al uso ya que los mensajes que envía al servidor no van cifrados, por lo tanto, este cliente, no sería útil con otras implementaciones de NETCONF ni sería correcto su uso más allá de la realización de pruebas. Al no ir cifrado, nos permite capturar los mensajes que envía y analizarlos. Por último, otra peculiaridad, es que no mantiene la sesión abierta, es decir, cuando indicamos una consulta que debe realizar, el cliente iniciará la conexión con el servidor, a través de los mensajes <hello>, hará la petición y cerrará la conexión, con la operación <close-session>.

5.1 Módulo YANG

Crearemos un módulo de prueba, sin uso práctico. El módulo se llamará *agradecimientos* y tendrá un nodo container principal llamado *gracias*, el cual contendrá, a su vez, un nodo leaf *alumno*, un nodo container *tutor* y dos nodos list, el primero llamado *familia* y el segundo *amigos*.

Los nodos *tutor*, *familia* y *amigos* contendrá dos nodos leaf, *nombre* y *motivo*. En el caso de los nodos list, la llave será el nodo leaf *nombre*.

La definición del módulo se encuentra en el Apéndice B.1

Utilizamos la herramienta de validación de módulos yangdump que nos hace un resumen del contenido del módulo y, si éste, contiene errores.

Validation results for agradecimientos.yang

```
*** Generated by yangdump-pro 16.10-7
*** Copyright (c) 2008-2012, Andy Bierman, All Rights Reserved.
*** Copyright (c) 2012-2017, YumaWorks, Inc., All Rights Reserved.

*** /home/andy/swdev/ncorg/ncorg/ncorg/workdir/agradecimientos.yang
*** 0 Errors, 0 Warnings
```

Reports for agradecimientos.yang

```
modversion:
module agradecimientos@2017-06-08

exports:
namespace http://miguelmerelo.es/NETCONFmoduleTFG
prefix agr
container gracias

dependencies:
identifiers:
container /gracias
leaf /gracias/alumno
container /gracias/tutor
leaf /gracias/tutor/nombre
leaf /gracias/tutor/motivo
list /gracias/familia
leaf /gracias/familia/nombre
leaf /gracias/familia/motivo
list /gracias/amigos
leaf /gracias/amigos/nombre
leaf /gracias/amigos/motivo
```

Figura 5.1 Resultado de la comprobación de un módulo YANG con Yangdump.

5.2 Introducción de datos en el módulo

Existen dos formas de trabajar con el cliente netconf-console-tcp, podemos utilizar parámetros específicos para una acción concreta, como <get> o <get-config>, o podemos utilizar el parámetro rpc el cual nos permite introducir el mensaje rpc al completo. En nuestro caso utilizaremos el parámetro rpc.

Introducimos los datos contenidos en el fichero editAgradecimientos.xml (Apéndice B.2) al almacén de datos running.

```
. /bin/netconf-console-tcp -u admin -p admin --port 2023 --rpc=
    editAgradecimientos.xml
```

Si el envío es correcto visualizaremos la respuesta del servidor.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <ok/>
</rpc-reply>
```

Figura 5.2 Mensaje <ok> de respuesta del servidor..

Si pasamos a analizar los mensajes intercambiados, podemos observar cómo se está produciendo la conexión, primero el envío del mensaje <hello> con todas las capabilities disponibles.

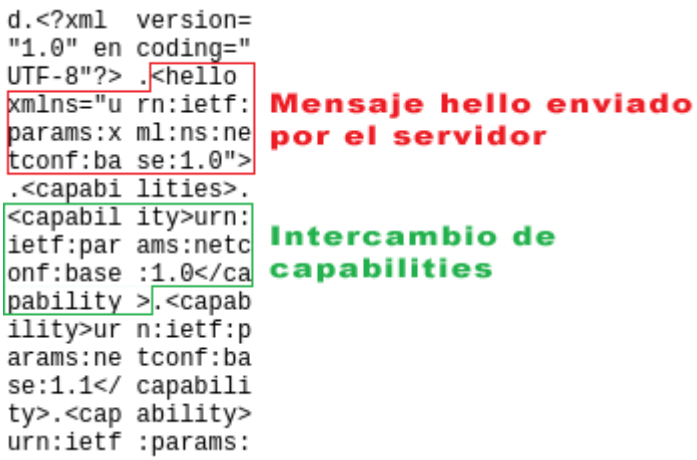

Figura 5.3 Análisis mensaje hello del servidor..

El cliente envía un mensaje similar.
A continuación se envía el mensaje rpc (Apéndice B.2) con los datos a modificar.

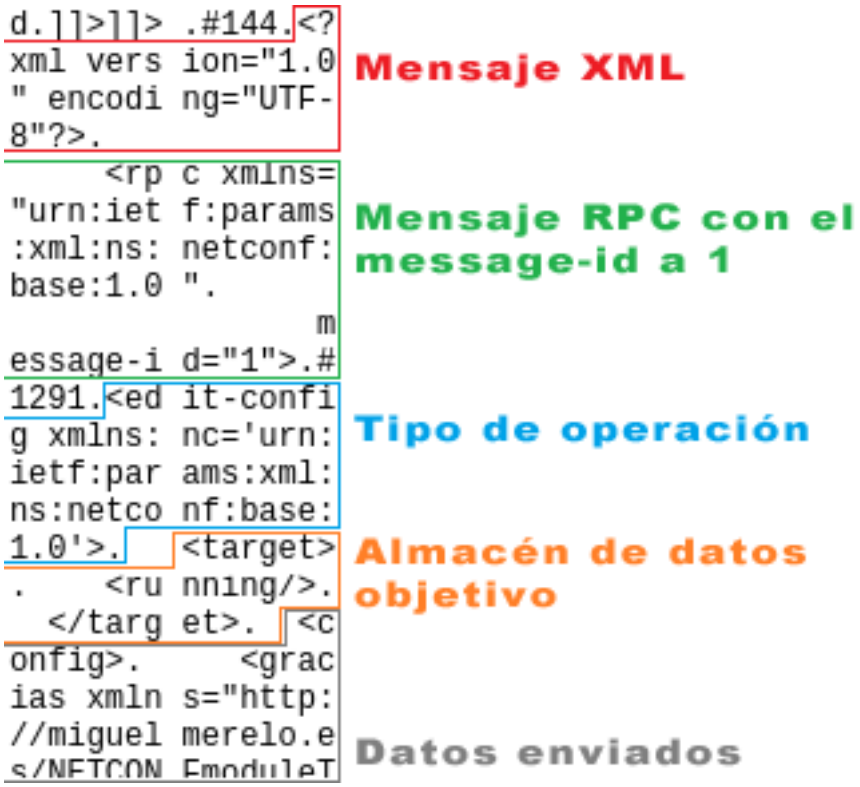

Figura 5.4 Análisis del mensaje <edit-config>..

Se recibe el mensaje ok.

```
d..#130. <?xml ve
rsion="1 .0" enco
ding="UT F-8"?>.<
rpc-repl y xmlns=
"urn:iet f:params
:xml:ns: netconf:
base:1.0 " messag
e-id="1" ><ok/></
rpc-repl y>.##.
```

Figura 5.5 Análisis mensaje OK del servidor..

Vemos que el message-id de respuesta, 1, coincide con el de la consulta.

En una implementación de un cliente NETCONF se mantendría abierta la conexión para realizar otras operaciones, en el cliente que estamos utilizando, como dijimos previamente, tras realizar una operación se cierra la conexión.

El cliente, izquierda, realiza la operación <close-session> y el servidor, derecha, responde con un mensaje ok.

```
e..#147. <?xml ve
rsion="1 .0" enco
ding="UT F-8"?>.
   <rpc  xmlns="u
rn:ietf: params:x
ml:ns:ne tconf:ba
se:1.0"  message-
id="0">.
<close-s ession/>
.    </r pc>
```

```
e..#130. <?xml ve
rsion="1 .0" enco
ding="UT F-8"?>.<
rpc-repl y xmlns=
"urn:iet f:params
:xml:ns: netconf:
base:1.0 " messag
e-id="0" ><ok/></
rpc-repl y>.##.
```

Figura 5.6 Mensajes <close-session> de cierre de sesión..

5.3 Consulta de la configuración

En este apartado vamos a realizar consultas de los datos de configuración.

Vamos a utilizar yangcli-pro para verificar que el servidor ejecuta correctamente NETCONF. Yangcli-pro hará la conexión utilizando ssh por lo que no podremos ver el contenido de los mensajes. Ejecutaremos la operación <get-config>, sin parámetros, con <target> running </target> que nos devolverá todos los datos del almacén de datos running.

```
          ssh_keydir /var/confd/homes/public/.ssh
          uid 9000
        }
      }
    }
  }
  gracias {
    alumno 'Miguel Merelo Hernández'
    amigos {
      motivo 'Sé que siempre podré contar contigo'
      nombre Alicia
    }
    amigos {
      motivo 'Avec l'ICIT tout a commencé'
      nombre Clara
    }
    amigos {
      motivo 'Primero tortuga, después GC, te gusta lo verde'
      nombre David
    }
    amigos {
      motivo '¿En tu pueblo o en el mío?'
      nombre Ignacio
    }
    amigos {
      motivo 'Gracias por tu incondicional apoyo, al MIT no voy, pero porque no quiero'
      nombre Juan
    }
    familia {
      motivo 'Por dejarme hacer lo que más me gusta'
      nombre 'A mis padres'
    }
    familia {
      motivo 'La distancia nunca nos separará'
      nombre Alberto
    }
    tutor {
      motivo 'Por ofrecerme este trabajo y por su guía durante el mismo'
      nombre 'Antonio Estepa Alonso'
    }
  }
  nacm {
    groups {
      group  admin {
        name admin
        user-name admin
        user-name private
      }
      group  oper {
        name oper
        user-name oper
        user-name public
      }
    }
    rule-list  admin {
      name admin
```

Figura 5.7 Respuesta a <get-config> sin filtro..

Recibimos todo el contenido del almacén de datos.

Si vemos el contenido de los mensajes.

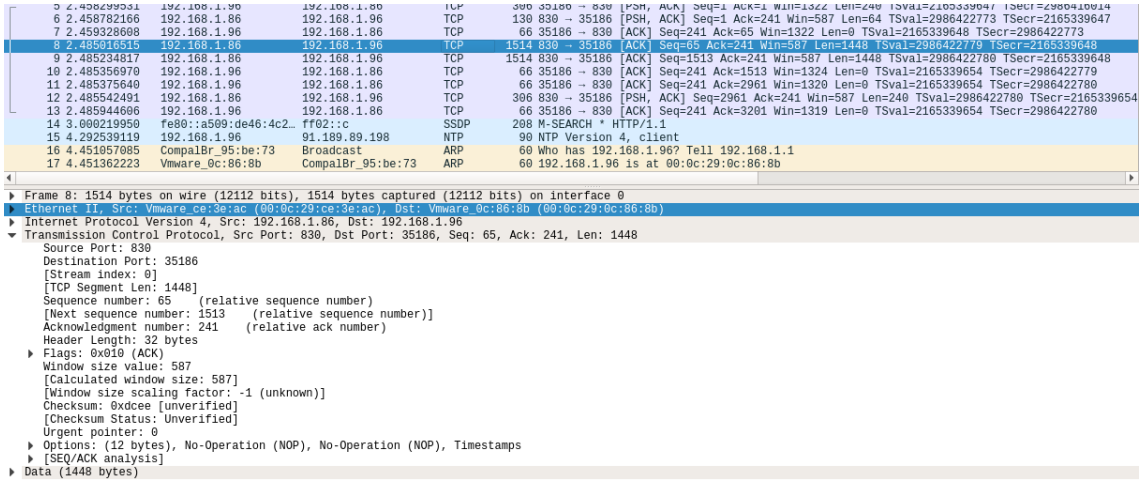

Figura 5.8 Captura de <get-config> cifrado. Con ssh..

Podemos suponer que el mensaje resaltado y el siguiente son los mensajes en los que se produce la respuesta

del servidor a la petición del cliente. Vemos como el puerto origen es el 830, puerto generalmente utilizado por NETCONF sobre ssh.

Existe la opción de realizar un filtrado de los resultado. En el siguiente ejemplo (Apéndice B.3), solo queremos que se nos muestre el árbol del módulo que hemos creado.

Recibimos la siguiente respuesta.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <data>
    <gracias xmlns="http://miguelmerelo.es/NETCONFmoduleTFG">
      <alumno>Miguel Merelo Hernández</alumno>
      <tutor>
        <nombre>Antonio Estepa Alonso</nombre>
        <motivo>Por ofrecerme este trabajo y por su guía durante el mismo</motivo>
      </tutor>
      <familia>
        <nombre>A mis padres</nombre>
        <motivo>Por dejarme hacer lo que más me gusta</motivo>
      </familia>
      <familia>
        <nombre>Alberto</nombre>
        <motivo>La distancia nunca nos separará</motivo>
      </familia>
      <amigos>
        <nombre>Alicia</nombre>
        <motivo>Sé que siempre podré contar contigo</motivo>
      </amigos>
      <amigos>
        <nombre>Clara</nombre>
        <motivo>Avec l'ICIT tout a commencé</motivo>
      </amigos>
      <amigos>
        <nombre>David</nombre>
        <motivo>Primero tortuga, después GC, te gusta lo verde</motivo>
      </amigos>
      <amigos>
        <nombre>Ignacio</nombre>
        <motivo>¿En tu pueblo o en el mío?</motivo>
      </amigos>
      <amigos>
        <nombre>Juan</nombre>
        <motivo>Gracias por tu incondicional apoyo, al MIT no voy, pero porque no quiero</motivo>
      </amigos>
    </gracias>
  </data>
</rpc-reply>
```

Figura 5.9 Mensaje recibido tras usar <get-config> con filtro..

5.4 Lectura de estadísticos

Con la operación <get> podemos realizar consultas de estadísticos tales como los módulos que implementa, la versión, las sesiones abiertas...

En el siguiente ejemplo (Apéndice B.4) hacemos una consulta de todos los estadísticos.

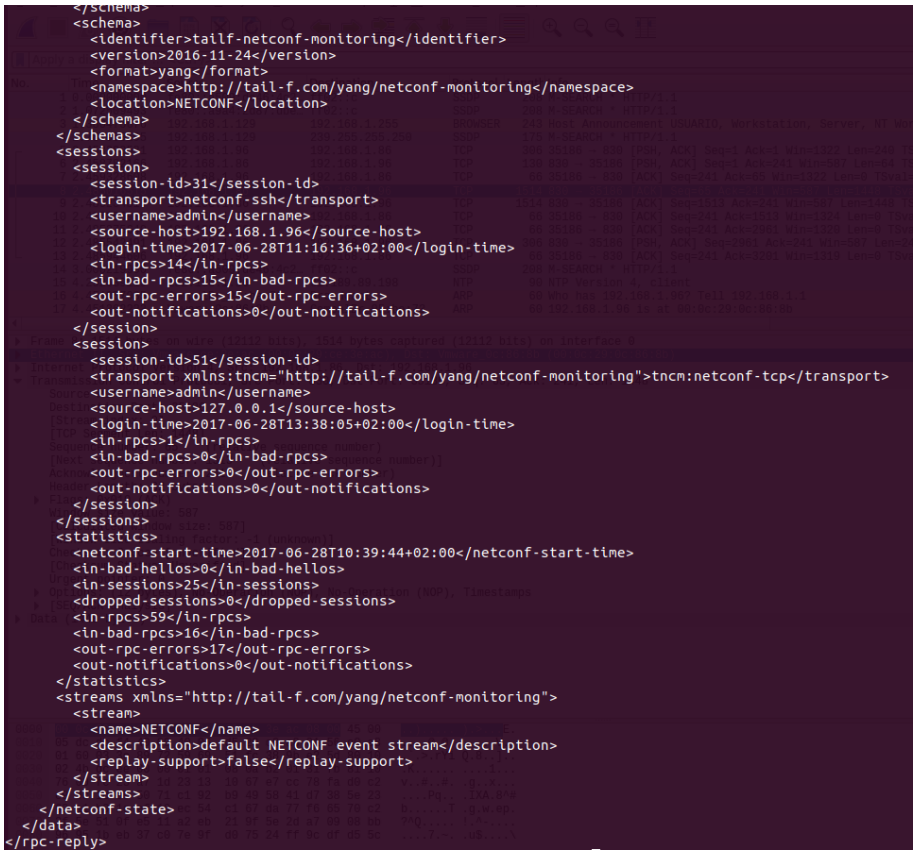

Figura 5.10 Operación <get> de estadísticos..

No se muestra la información completa de lo que devuelve.

También podemos aplicar filtrado a esta consulta. En concreto vamos a pedir solo los datos de las sesiones abiertas (Apéndice B.5).

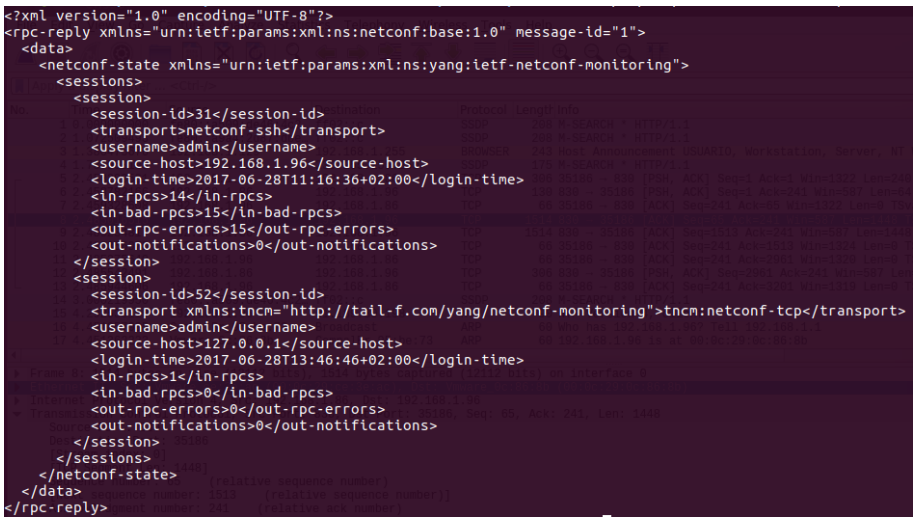

Figura 5.11 Operación <get> con filtro para las sesiones..

5.5 Commit

Una de las funciones que se destacan de NETCONF es la posibilidad de revertir los cambios de manera sencilla. Una vez hayamos modificado el almacén de datos candidato, podremos ejecutar la operación commit. Haciendo uso de la etiqueta <confirmed> y <confirm-timeout> podremos definir el tiempo máximo en el que quedará a la espera de un commit. Si no recibiera esta segunda operación, en el plazo fijado, se anularían los cambios.

Realizamos commit para realizar el cambio de configuración manteniéndonos a la espera de su confirmación. Fijamos el tiempo máximo de espera en 120 segundos. La etiqueta persist permite que se pueda realizar el commit o cancel-commit para este cambio desde otra sesión.

```
<commit>
    <confirmed/>
    <confirm-timeout>120</confirm-timeout>
    <persist>persiste</persist>
</commit>
```

A continuación se encuentran las operaciones para confirmar el cambio, commit, y para cancelarlo, cancel-commit. Nótese que incluyen la etiqueta persist con el mismo valor que en el primer commit.

```
<commit>
  <persist-id>persiste</persist-id>
</commit>
```

```
<cancel-commit>
  <persist-id>persiste</persist-id>
</cancel-commit>
```

5.6 Notificaciones

Tenemos la posibilidad de suscribirnos a notificaciones que implemente el agente. Por ejemplo, podemos suscribirnos para que nos avise cada vez que se realiza una nueva conexión. Haremos uso de la operación <create-subscription>. Nos suscribimos al stream NETCONF.

```
<create-subscription xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <stream>NETCONF</stream>
</create-subscription>
```

La lista de stream disponibles la podemos obtener realizando <get> a los estadísticos.

Las siguientes respuestas se corresponden a un usuario que inicia una sesión con el servidor y la cierra inmediatamente después.

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2017-06-28T14:15:03.965253+02:00</eventTime>
  <netconf-session-start xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-notifications">
    <username>admin</username>
    <session-id>60</session-id>
    <source-host>192.168.1.96</source-host>
  </netconf-session-start>
</notification>
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2017-06-28T14:15:12.251967+02:00</eventTime>
  <netconf-session-end xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-notifications">
    <username>admin</username>
    <session-id>60</session-id>
    <source-host>192.168.1.96</source-host>
    <termination-reason>dropped</termination-reason>
  </netconf-session-end>
</notification>
```

Figura 5.12 Captura de las notificaciones recibidas..

6 Conclusiones

A lo largo de este trabajo se ha intentado dar una visión de conjunto del protocolo NETCONF y definir sus características y funcionamiento. Se han estudiado las herramientas existentes para su utilización, el modelo de datos YANG y la implementación del mismo en el mercado.

En el año 2002 se definen una serie de funciones que deberían existir en un protocolo de gestión de configuración. NETCONF toma estas ideas y las lleva a cabo resolviéndose en un protocolo intuitivo a la par que eficaz. Un protocolo que define claramente los datos de configuración y de operación, que diferencia entre despliegue y activación de la configuración, que permite almacenar múltiples configuraciones y que es capaz de realizar una monitorización de los cambios que se producen en el dispositivo.

NETCONF ha alcanzado una implementación en los productos a la venta de los fabricantes de alrededor del 30 % y se prevé un aumento del mismo, teniendo en cuenta las declaraciones de uno de los mayores suministradores de dispositivos de red, CISCO.

NETCONF es un protocolo que mejora los sistemas actuales de gestión de la configuración como puedan ser cli y snmp ofreciendo muchas ventajas. La contrapartida es que es un protocolo poco conocido y, por tanto, poco demandado. La extensión del mismo dependerá de las facilidades que los fabricantes ofrezcan a sus clientes para trabajar con ello. A pesar de todo ello, NETCONF es una puesta de futuro para facilitar el manejo de grandes redes hacia las que estamos convergiendo de modo irreversible.

Apéndice A

Análisis fabricantes

En este apéndice se encuentran los datos relativos al análisis sobre la incorporación de NETCONF y de SNMP en los conmutadores y enrutadores de los fabricantes Cisco, Juniper, HP, Aruba, Alcatel-Lucent y Nokia.
Los datos aquí presentes se han obtenido de los datasheets de los productos que los fabricantes tienen a la venta. Han sido tomados en el mes de Mayo del año 2017.

A.1 Cisco

A.1.1 Conmutadores

Modelo	NETCONF	SNMP
Cisco Industrial Ethernet 5000 Series Switches	NO	SI
Cisco Industrial Ethernet 4010 Series Switches	NO	SI
Cisco Industrial Ethernet 4000 Series Switches	NO	SI
Cisco Industrial Ethernet 3010 Series Switches	NO	SI
Cisco Industrial Ethernet 3000 Series Switches	NO	SI
Cisco Industrial Ethernet 2500 Series Switches	NO	SI
Cisco Industrial Ethernet 2020 Series Switches	NO	SI
Cisco Industrial Ethernet 2000 Series Switches	NO	SI
Cisco Industrial Ethernet 2000U Series Switches	NO	SI
Cisco Industrial Ethernet 1000 Series Switches	NO	SI
CiscoCatalyst 7000 Series	NO	SI
Cisco Catalyst 6800 Series	NO	SI
Cisco Catalyst 6500 Series	NO	SI
Cisco Catalyst 4900 Series	NO	SI
Cisco Catalyst 4500 Series	NO	SI
Cisco Catalyst 4500X Series	NO	SI
Cisco Catalyst 3850 Series	NO	SI
Cisco Catalyst 3750 Series	NO	SI
Cisco Catalyst 3650 Series	NO	SI
Cisco Catalyst 3560 Series	NO	SI
Cisco Catalyst 3560-C Series	NO	SI
Cisco Catalyst 3560-CX Series	NO	SI
Cisco Catalyst 2960-C Series	NO	SI
CiscoCatalyst 2960-CX Series	NO	SI
Cisco Catalyst 2960-Plus Series	NO	SI
Cisco Catalyst 2960-X Series	NO	SI
Cisco Blade Switches for Dell	NO	SI
Cisco Blade Switches for FSC	NO	SI
Cisco Blade Switches for HP	NO	SI
Cisco Blade Switches for IBM	NO	SI
Cisco Nexus 4000 Series Switches	SI	SI
Cisco Nexus 7000 Series Switches	NO	SI
Cisco Nexus 6000 Series Switches	SI	SI
Cisco Nexus 5000 Series Switches	SI	SI
Cisco Nexus 4000 Series Switches	SI	SI
Cisco Nexus 3000 Series Switches	SI	SI
Cisco Nexus 2000 Series Fabric Extenders	SI	SI
Cisco ME 4900 Series Ethernet Switches	NO	SI
Cisco ME 3800X Series Carrier Ethernet Switch Routers	NO	SI
Cisco ME 3600X Series Ethernet Access Switches	NO	SI
Cisco ME 3400 Series Ethernet Access Switches	NO	SI
Cisco SFS 7000 Series InfiniBand Server Switches	NO	SI

A.1.2 Enrutadores

Modelo	NETCONF	SNMP
Cisco 4000 Series Integrated Services Routers	SI	SI
Cisco 3900 Series Integrated Services Routers	NO	SI
Cisco 3800 Series Integrated Services Routers	NO	SI
Cisco 2900 Series Integrated Services Routers	NO	SI
Cisco 2800 Series Integrated Services Routers	NO	SI
Cisco 1900 Series Integrated Services Routers	NO	SI
Cisco 1800 Series Integrated Services Routers	NO	SI
Cisco 800 Series Integrated Services Routers	NO	SI
Cisco 890 Series Routers	NO	SI
Cisco 880 Series Routers	NO	SI
Cisco 870 Series Routers	NO	SI
Cisco 860 Series Routers	NO	SI
Cisco 850 Series Routers	NO	SI
Cisco 800m Series Routers	NO	SI
Cisco Network Convergence System 6000 Series Routers	SI	SI
Cisco Network Convergence System 5011 Series Routers	SI	SI
Cisco Network Convergence System 5000 Series Routers	SI	SI
Cisco ASR 1000 Series Aggregation Services Routers	NO	SI
Cisco 7300 Series Routers	NO	SI
Cisco 800 Series Industrial Integrated Services Routers	NO	SI
Cisco 2000 Series Connected Grid Routers	NO	SI
Cisco 1000 Series Connected Grid Routers	SI	SI

A.2 Juniper

En el caso de Juniper, algunos de sus dispositivos tienen un protocolo de gestión basado en XML y RCP similar a NETCONF, están indicados con un asterisco*.

A.2.1 Conmutadores

Modelo	NETCONF	SNMP
EX9200	NO	SI
EX4600	NO	SI
EX4550	NO	SI
EX4300	NO	SI
EX4200	NO	SI
EX3400	NO	SI
EX2300	NO	SI
EX2200	NO	SI
QFX10000	NO*	SI
QFX5200	NO*	SI
QFX5110	NO*	SI
QFX5100	NO*	SI
OCX1100	NO*	SI

A.2.2 Enrutadores

Modelo	NETCONF	SNMP
MX2020	SI	SI
MX2010	SI	SI
MX2008	SI	SI
MX960	SI	SI
MX480	SI	SI
MX240	SI	SI
MX104	SI	SI
MX80	SI	SI
MX40	SI	SI
MX10	SI	SI
MX5	SI	SI
PTX5000	NO*	SI
PTX3000	NO*	SI
PTX1000	NO*	SI
ACX5000	SI	SI
ACX4000	SI	SI
ACX2200	SI	SI
ACX2100	SI	SI
ACX1100	SI	SI
ACX1000	SI	SI
ACX500	SI	SI
T4000	SI	SI
T1600	SI	SI
T640	SI	SI
TX Matrix	SI	SI

A.3 Hewlett Packard

A.3.1 Conmutadores

Modelo	NETCONF	SNMP
Arista 7280 Series	NO	SI
Arista 7260 Series	NO	SI
Arista 7250 Series	NO	SI
Arista 7160 Series	NO	SI
Arista 7150 Series	NO	SI
Arista 7060X Series	NO	SI
Arista 7050X Series	NO	SI
Arista 7010T Gigabit Ethernet Data Center Switch Series	NO	SI
HPE Altoline 6940 Series	NO	SI
HPE FlexFabric 5950 Switch	SI	SI
HPE FlexFabric 5940 Series	SI	SI
HPE FlexFabric 5930 Series	SI	SI
HPE FlexFabric 5920 Series	SI	SI
HPE FlexFabric 5900 Series	SI	SI
HPE FlexFabric 5700 Series	SI	SI
Aruba 3810 Switch Series	NO	SI
Aruba 3800 Series	NO	SI
Aruba 2930M Switch Series	NO	SI
Aruba 2920 Switch Series	NO	SI
Aruba 2915 8G PoE Switch	NO	SI
Aruba 2540 Switch Series	NO	SI
Aruba 2530 Series	NO	SI
Aruba 2620 Series	NO	SI
Aruba 2615 8 PoE Switch	NO	SI
HPE FlexNetwork 5820 Series	NO	SI
HPE FlexNetwork 5510 HI Series	SI	SI
HPE FlexNetwork 5500 HI Series	NO	SI
HPE FlexNetwork 5130 HI Series	SI	SI
HPE FlexNetwork 5120 SI Series	NO	SI
HPE FlexNetwork 3600 SI Series	NO	SI
HPE OfficeConnect 1950 Series	NO	SI
HPE OfficeConnect 1920S Switch Series	NO	SI
HPE OfficeConnect 1910 Series	NO	SI
HPE OfficeConnect 1820 Series	NO	SI
HPE OfficeConnect PS1810 Switch Series	NO	SI
HPE OfficeConnect 1410 Series	NO	SI
HPE OfficeConnect 1620 Series	NO	SI
Arista 7500R series	SI	SI
Arista 7320 X-Series	NO	SI
Arista 7300X Series	NO	SI
HPE FlexFabric 12900E Series	SI	SI
HPE FlexFabric 7900 Series	SI	SI
Aruba 5400R zl2 Switch Series	NO	SI
HPE FlexNetwork 10500 Series	SI	SI
HPE FlexNetwork 7500 Series	SI	SI

A.3.2 Enrutadores

Modelo	NETCONF	SNMP
HPE FLexNetwork MSR95x Router Series	NO	SI
HPE FlexNetwork HSR6800 Router Series	NO	SI
Alcatel-Lucent 7750 Service Router Series	NO	SI
HPE FlexNetwork MSR1000 Router Series	NO	SI
HPE FlexNetwork MSR4000 Router Series	NO	SI
HPE FlexNetwork MSR2000 Router Series	NO	SI
HPE FlexNetwork MSR3000 Router Series	NO	SI
HPE FlexNetwork MSR93x Router Series	NO	SI
HPE FlexNetwork HSR6600 Router Series	NO	SI
HPE MSR20-1x Series	NO	SI

A.4 Aruba

A.4.1 Conmutadores

Modelo	NETCONF	SNMP
Serie 5400R	NO	SI
Serie 3810	NO	SI
Serie 2930F	NO	SI
Serie 2920	NO	SI
Serie 2540	NO	SI
Serie 2530	NO	SI

A.5 Alcatel-Lucent

A.5.1 Conmutadores

Modelo	NETCONF	SNMP
OmniSwitch 6250	NO	SI
OmniSwitch 6350	NO	SI
OmniSwitch 6450	NO	SI
OmniSwitch 6855	NO	SI
OmniSwitch 6860	NO	SI
OmniSwitch 6865	NO	SI
OmniSwitch 10K	NO	SI
OmniSwitch 9900	NO	SI
OmniSwitch 6900	NO	SI

A.5.2 Enrutadores

Modelo	NETCONF	SNMP
OmniAccess ESR 5720	NO	SI
OmniAccess ESR 5800	NO	SI
OmniAccess ESR 7500	NO	SI

A.6 Nokia

A.6.1 Conmutadores

Modelo	NETCONF	SNMP
7210 Service Access Switch	SI	SI
7450 Ethernet Service Switch	SI	SI
1850 Transport Service Switch	NO	SI

A.6.2 Enrutadores

Modelo	NETCONF	SNMP
7950 Extensible Routing System	SI	SI
7750 Service Router	SI	SI
7705 Service Aggregation Router	NO	SI

Apéndice B
Prueba de concepto

B.1 Módulo YANG

```
module agradecimientos{
  yang-version 1.1;
  namespace "http://miguelmerelo.es/NETCONFmoduleTFG";
  prefix "agr";

  organization
    "US TFG";

  contact
    "Miguel Merelo: <mailto:merelo.miguel@ieee.org>";

  description
    "Testing data model for gratitude in my TFG";

  revision "2017-06-08" {
    description "Initial revision.";
    reference
      "http://miguelmerelo.es/NETCONFmoduleTFG";
  }

  container gracias {
    description
      "Agradecimientos";

    leaf alumno{
      type string;
      description
        "Nombre del alumno que desarrolla el TFG";
    }

    container tutor{
      description
        "Información del tutor del TFG";

      leaf nombre{
        type string;
        description
          "Nombre del tutor del TFG";
```

```
      }
      leaf motivo{
        type string;
        description
          "Agradecimientos";
      }
    } //tutor

    list familia{
      key "nombre";
      description
        "Familiares a los que agradecer";

      leaf nombre{
        type string;
        description
          "Nombre del familiar";
      }
      leaf motivo{
        type string;
        description
          "Agradecimientos";
      }
    } //familia

    list amigos{
      key "nombre";
      description
        "Amigos a los que agradecer";

      leaf nombre{
        type string;
        description
          "Nombre del amigo";
      }
      leaf motivo{
        type string;
        description
          "Agradecimientos";
      }
    } //amigos
  } //gracias
} //agradecimientos
```

B.2 Editar Agradecimientos

```
<edit-config xmlns:nc='urn:ietf:params:xml:ns:netconf:base:1.0'>
  <target>
    <running/>
  </target>
  <config>
    <gracias xmlns="http://miguelmerelo.es/NETCONFmoduleTFG">
      <alumno>Miguel Merelo Hernández</alumno>
      <tutor>
```

```
        <nombre>Antonio Estepa Alonso</nombre>
        <motivo>Por ofrecerme este trabajo y por su guía durante el mismo</
            motivo>
      </tutor>
    <familia>
    <nombre>A mis padres</nombre>
    <motivo>Por dejarme hacer lo que más me gusta</motivo>
    </familia>
    <familia>
    <nombre>Alberto</nombre>
    <motivo>La distancia nunca nos separará</motivo>
    </familia>
    <amigos>
        <nombre>Clara</nombre>
        <motivo>Avec l'ICIT tout a commencé</motivo>
      </amigos>
    <amigos>
        <nombre>Juan</nombre>
        <motivo>Gracias por tu incondicional apoyo, al MIT no voy, pero porque
            no quiero</motivo>
      </amigos>
    <amigos>
        <nombre>Ignacio</nombre>
        <motivo>¿En tu pueblo o en el mío?</motivo>
      </amigos>
    <amigos>
        <nombre>David</nombre>
        <motivo>Primero tortuga, después GC, te gusta lo verde</motivo>
      </amigos>
      <amigos>
        <nombre>Alicia</nombre>
        <motivo>Sé que siempre podré contar contigo</motivo>
      </amigos>
    </gracias>
  </config>
</edit-config>
```

B.3 Filtrado Agradecimientos

```
<get-config>
  <source>
    <running/>
  </source>
  <filter>
    <gracias xmlns="http://miguelmerelo.es/NETCONFmoduleTFG"/>
  </filter>
</get-config>
```

B.4 Get Estadísticos

```
<get>
  <filter>
```

```
    <netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring
        "/>
  </filter>
</get>
```

B.5 Get Estadísticos con filtro

```
<get>
  <filter>
    <netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
      <sessions/>
    </netconf-state>
  </filter>
</get>
```

Bibliografía

[1] IETF. (1990, May) Rfc-1157. [Online]. Available: https://tools.ietf.org/html/rfc1157

[2] ——. (2011, June) Rfc-6241. [Online]. Available: https://tools.ietf.org/html/rfc6421

[3] ——. (2003, May) Rfc-3535. [Online]. Available: https://tools.ietf.org/html/rfc3535

[4] ——. (2009, December) Rfc-5717. [Online]. Available: https://tools.ietf.org/html/rfc5717

[5] ——. (2010, October) Rfc-6022. [Online]. Available: https://tools.ietf.org/html/rfc6022

[6] ——. (2008, July) Rfc-5277. [Online]. Available: https://tools.ietf.org/html/rfc5277

[7] ——. (2012, February) Rfc-6470. [Online]. Available: https://tools.ietf.org/html/rfc6470

[8] S. Gupta. (2017, April) Network programmability – bringing the digital revolution to network programming. [Online]. Available: https://blogs.cisco.com/enterprise/network-programmability-digital-revolution

[9] IETF. (2016, August) Rfc-7950. [Online]. Available: https://tools.ietf.org/html/rfc7950